BAOU
Education for All

Dr. Babasaheb Ambedkar
Open University
(Established by Government of Gujarat)

# Introduction to Web Designing

BSCIT - 305

Bachelor of Science (Hons.)
Information Technology

2022

# Introduction to Web Designing

Dr. Babasaheb Ambedkar Open University

# Introduction to Web Designing

**Expert Committee**

| | |
|---|---|
| Prof. (Dr.) Nilesh K. Modi<br>Professor and Director, School of Computer Science,<br>Dr. Babasaheb Ambedkar Open University, Ahmedabad | (Chairman) |
| Prof. (Dr.) Ajay Parikh<br>Professor and Head, Department of Computer Science<br>Gujarat Vidyapith, Ahmedabad | (Member) |
| Prof. (Dr.) Satyen Parikh<br>Dean, School of Computer Science and Application<br>Ganpat University, Kherva, Mahesana | (Member) |
| M. T. Savaliya<br>Associate Professor and Head<br>Computer Engineering Department<br>Vishwakarma Engineering College, Ahmedabad | (Member) |
| Mr. Nilesh Bokhani<br>Assistant Professor, School of Computer Science,<br>Dr. Babasaheb Ambedkar Open University, Ahmedabad | (Member) |
| Dr. Himanshu Patel<br>Assistant Professor, School of Computer Science,<br>Dr. Babasaheb Ambedkar Open University, Ahmedabad | (Member Secretary) |

**Course Writer**

Ms. Khyati Shah — Som-Lalit Institute of Computer Application, Ahmedabad

**Content Editor**

Mr. Nilesh N. Bokhani — Assistant Professor, School of Computer Science,
Dr. Babasaheb Ambedkar Open University, Ahmedabad

**Content Reviewer**

Prof. (Dr.) Nilesh K. Modi — Professor and Director, School of Computer Science,
Dr. Babasaheb Ambedkar Open University, Ahmedabad

Copyright © Dr. Babasaheb Ambedkar Open University – Ahmedabad. June 2022

**ISBN** -

**Printed and published by**: Dr. Babasaheb Ambedkar Open University, Ahmedabad While all efforts have been made by editors to check accuracy of the content, the representation of facts, principles, descriptions and methods are that of the respective module writers. Views expressed in the publication are that of the authors, and do not necessarily reflect the views of Dr. Babasaheb Ambedkar Open University. All products and services mentioned are owned by their respective copyrights holders, and mere presentation in the publication does not mean endorsement by Dr. Babasaheb Ambedkar Open University. Every effort has been made to acknowledge and attribute all sources of information used in preparation of this learning material. Readers are requested to kindly notify missing attribution, if any.

Dr. Babasaheb Ambedkar Open University

BSCIT-305

# Introduction to Web Designing

## BLOCK1: INTRODUCTION TO WEB AND HTML


**UNIT-1**
INTRODUCTION TO WEB 06

**UNIT-2**
HYPERTEXT MARKUP LANGUAGE 21

**UNIT-3**
WORKING WITH TEXT 32

**UNIT-4**
WORKING WITH LISTS, TABLES AND FRAMES 40


## BLOCK-2: ADVANCED HTML AND HTML5


**UNIT-5**
WORKING WITH HYPERLINKS, IMAGES, MULTIMEDIA AND FORMS 64

**UNIT-6**
INTRODUCTION TO HTML5 88

**UNIT-7**
INTRODUCTION TO NEW ELEMENTS IN HTML5 98

## BLOCK-3: DYNAMIC HTML CONCEPTS (CSS,JAVA SCRIPT)


**UNIT-8**
CASCADING STYLE SHEET 105

**UNIT-9**
CASCADING STYLE SHEET ATTRIBUTES AND PROPERTIES 119

**UNIT-10**
INTRODUCTION TO JAVA SCRIPT 130

**UNIT-11**
ARRAYS, FUNCTIONS, EVENTS AND DIALOG BOXES IN JAVA SCRIPT 150


## BLOCK-4: CONNECTIVITY DEVICES, NETWORK TOPOLOGIES AND ARCHITECTURE


**UNIT-12**
JAVA SCRIPT OBJECTS, METHODS AND PROPERTIES 159

**UNIT-13**
INTRODUCTION TO JQUERY 194

**UNIT-14**
JQUERY SELECTORS, FUNCTIONS, EFFECTS AND EVENTS 199

**UNIT-15**
INTRODUCTION TO XML 298

# Introduction to Web-Desisgning

## Contents

**BLOCK1: INTRODUCTION TO WEB AND HTML**

**UNIT 1 INTRODUCTION TO WEB**

Introduction of www, Web system architecture, Exploring HTTP, Uniform resource locator, Domain name and IP address, Web browsers, Web pages, Use of cookies, Summary

**UNIT 2 HYPERTEXT MARKUP LANGUAGE**

Introduction to HTML, HTML editors, Basic structure of HTML, Getting started with HTML, Displaying Web page in Web Browser, Modifying the Background of HTML page, Specifying Metadata of HTML page, Summary

**UNIT 3 WORKING WITH TEXT**

Adding plain Text to an HTML Web page, Adding Text in New Line, Creating Headings on a Web page, Creating a Paragraph, Creating a Horizontal Ruler Line, Inserting the <pre> tag, Formatting Tags, Aligning the Text, Grouping the Text, Indenting Quotations, Working with Character Entities, Commenting the Text, Summary

**UNIT 4 WORKING WITH LISTS,TABLES AND FRAMES**

Working with Lists, Working with Tables, Working with Frames, Summary

**BLOCK 2: ADVANCED HTML AND HTML5**

**UNIT 5 WORKING WITH HYPERLINKS, IMAGES, MULTIMEDIA AND FORMS**

Working with Hyperlinks, Working with Images, Working with Image Maps, Working with Multimedia, Working with Forms, Summary

**UNIT 6 INTRODUCTION TO HTML5**

Introduction to HTML5, New Document Structure of HTML5, Browser Support for HTML5, Defining HTML Markup, Summary

**UNIT 7 INTRODUCTION TO NEW ELEMENTS IN HTML5**

Markup Elements in HTML5, The canvas Element, New Elements in Forms , Summary

---

## BLOCK 3: DYNAMIC HTML CONCEPTS(CSS,JAVA SCRIPT)

**UNIT 8 CASCADING STYLE SHEET**

Introduction to Cascading Style Sheet, CSS Syntax, CSS Selectors, Selectors Grouping, CSS Comments, Types of Style Sheets, Summary

**UNIT 9 CASCADING STYLE SHEET ATTRIBUTES AND PROPERTIES**

CSS Color Attribute, CSS Background Attributes, CSS Font Attributes, CSS Text Attributes, CSS Border Attributes, CSS Margin Attributes, CSS Height and Width Attributes, CSS Padding Attributes, CSS List Attributes, CSS Table Attributes, CSS Position Attributes, Summary

**UNIT 10 INTRODUCTION TO JAVA SCRIPT**

Introduction to Java Script, Select Developing Environment for Java Script, HTML and Java Script, Elements of Java Script, Java Script Variables, Types of Data in Java Script, Java Script Operators, Java Script Flow Control Statements, Summary

**UNIT 11 ARRAYS, FUNCTIONS, EVENTS AND DIALOG BOXES IN JAVA SCRIPT**

Java Script Arrays, Java Script Functions, Java Script Popup Boxes, Java Script Events, Summary

---

## BLOCK 4: JAVA SCRIPT OBJECTS, JQUERY AND XML

**UNIT 12 JAVA SCRIPT OBJECTS, METHODS AND PROPERTIES**

Java Script Document Object, Java Script Array Object, Java Script String Object, Java Script Date Object, Java Script Math Object, Java Script Window Object, Summary

**UNIT 13 INTRODUCTION TO JQUERY**

What is JQuery, Features and Advantages of JQuery, Using JQuery, JQuery Syntax, Connecting JQuery to Load Event(Document Ready Event), Summary

**UNIT 14 JQUERY SELECTORS, FUNCTIONS, EFFECTS AND EVENTS**

JQuery Selectors, JQuery Traversing, JQuery Attributes, JQuery Effects, JQuery Events, Summary

**UNIT 15 INTRODUCTION TO XML**

What is XML, XML Verses HTML, XML Syntax, XML References, XML Declaration, XML Comments, XML Terminologies, XML Namespace, Summary

Dr. Babasaheb Ambedkar Open University

**BSc(IT)-305**

# Introduction to Web-Deisigning

## BLOCK1: INTRODUCTION TO WEB AND HTML


**UNIT 1**

INTRODUCTION TO WEB 10

**UNIT 2**

HYPERTEXT MARKUP LANGUAGE 25

**UNIT 3**

WORKING WITH TEXT 43

**UNIT 4**

WORKING WITH LISTS,TABLES AND FRAMES 71

# BLOCK 1: INTRODUCTION TO WEB AND HTML

## Block Introduction

In this block-1 of web technhologies, I have tried to emphasis on: What is web? And What is the importance of it? What is HTML? Basically, Iintroduced basics of HTML, how to create HTML document . First aweb designer has to know the basic web page development language. Then information related to language has to be gathered and analyze. As an output of this small web pages can be created and merged.

I have also described, basic tags used to write a HTML document, and tags used for text, use to create list, create tables and create frames.Basically, these tags gives developer more flexibility to create document.

## Block Objective

The objective of the block is to explain what is an web and what is HTML. Students will able to learn about web and hyper text markup languagetocreate web pages. Detail study of tags is required to build website.

By learning this block of web technology student will learn about different tags of HTML, to createweb page. Reader of this block, will know web page development through various tags.

Different websites have different challenges, depending upon nature of the problem, Different tags are used to find the solution and create web pages. This block servers knowledge of different tags. I hope, this block will clear the idea of what is web and hypertext markup language, it’s requirement so that student can become ready for web page development.

# Unit 1: Introduction to Web

## Unit Structure


1.1. Introduction to WWW

1.2. The Web System Architecture

1.3. Exploring HTTP

1.4. Uniform Resource Locator

1.5. Domain Name and IP Address

1.6. Web Browsers

1.7. Web Pages

1.8. Use of Cookies to Store Information

1.9. Summary

# 1.1 Introduction to WWW

WWW is a collection of internetresources (such as FTP, telnet), hyperlinked text, audio, and videofiles, and remotesites that can be accessed and searched by browsers based on standards such as HTTP and TCP/IP.

The Web, or World Wide Web, is basically a system that support specially formatted documents. The documents are formatted in a markup language called HTML (*HyperText Markup Language*) that supports links to other documents, as well as graphics, audio, and video files.

This means you can jump from one document to another simply by clicking.

There are several applications called Web browsers that make it easy to access the World Wide Web.

**Example**Firefox ,Microsoft's Internet Explorer,crome etc.

## Evolution of World Wide Web

In the beginning, World Wide Web was introduced as a medium for sharing scientific and research documents, especially, between government organizations and academic institutions.

It was created in 1989 by the UK physicist Tim Berners-Lee while working at the European Particle Physics Laboratory called CERN(European Organization for Nuclear Research) in Switzerland, as an easier way to accessinformation scattered across the internet.

By 1991, it became available to anyone using internet.

Evolution of web from Web 1.0 (The World Wide Web) to Web 2.0 (The Social Web) and then to Web 3.0 (The Semantic Web) is shown in following figure 1.

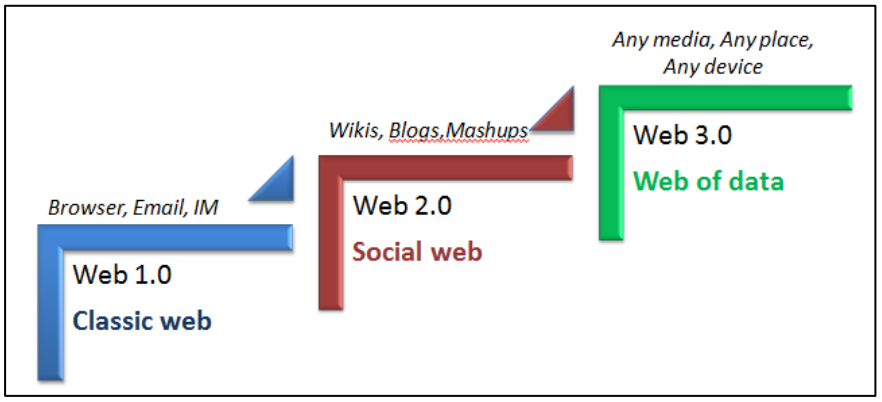



Figure 1: showing the Evolution of Web page

## Basic elements (components) of World Wide Web

Figure 2 shows the Elements of WWW

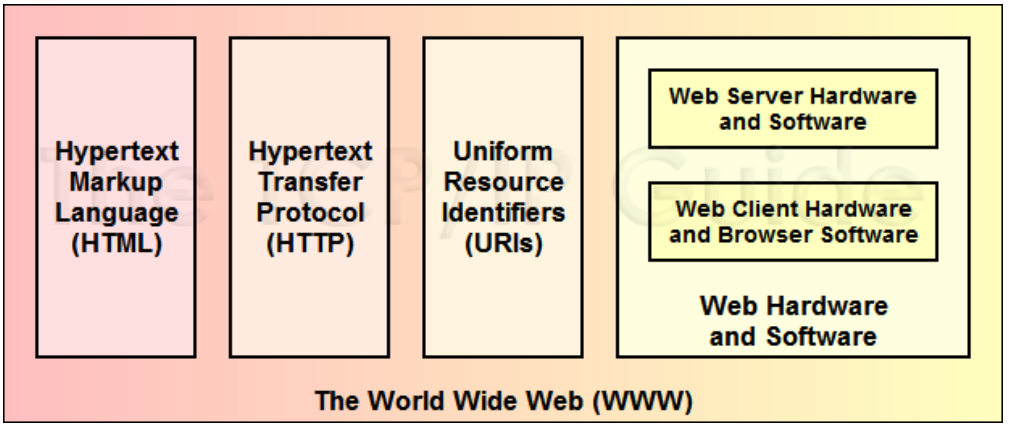



Figure 2: showing the Elements of WWW

- HTML: HyperText Markup Language. The markup (formatting) language for the Web.

- URI: Uniform Resource Identifier. A kind of “address” that is unique and used to identify to each resource on the Web. It is also commonly called a URL.

- HTTP: Hypertext Transfer Protocol. Allows for the retrieval of linked resources from across the Web.

## 1.2 The Web System Architecture

The web application architecture describes the interactions between applications, databases, and middleware systems on the web. It ensures that multiple applications work simultaneously. Let us understand it with a simple example of opening a webpage.

As soon as the user hits the go button after typing a URL in the address bar of a web browser, it requests for that particular web address. The server sends files to the browser as a response to the request made. The browser then executes those files to show the requested page.

Finally, the user is able to interact with the website. The most important thing to note here is the code parsed by the web browser. A web app works in a similar way.

Figure 3 shows the Web System Architecture

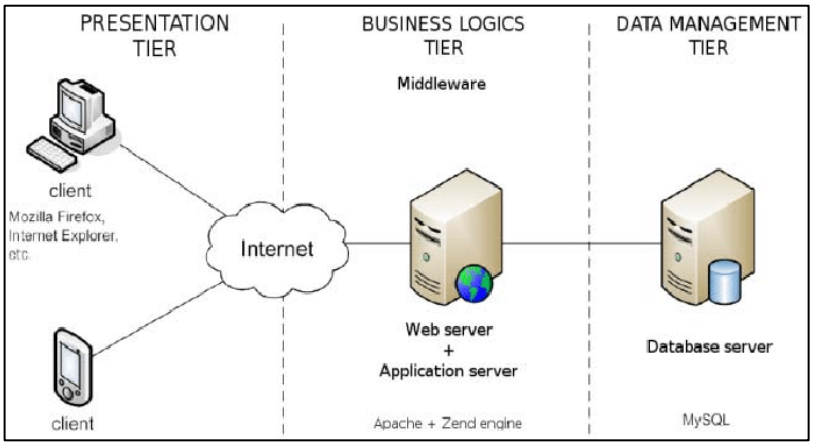



Figure 3: showing the Web System Architecture

## 1.3 Exploring HTTP

HTTP stands for Hyper Text Transfer Protocol.The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (i.e. internet) since 1990. HTTP is a generic and stateless protocol which can be used for other purposes as well using extensions of its request methods, error codes, and headers.

### Basic Features

There are three basic features that make HTTP a simple but powerful protocol:

- **HTTP is connectionless:** The HTTP client, i.e., a browser initiates an HTTP request and after a request is made, the client waits for the response. The server processes the request and sends a response back after which client disconnect the connection. So client and server knows about each other during current request and response only. Further requests are made on new connection like client and server are new to each other.
- **HTTP is media independent:** It means, any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type using appropriate MIME-type.
- **HTTP is stateless:** As mentioned above, HTTP is connectionless and it is a direct result of HTTP being a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages.

## Basic Architecture

The following diagram shows a very basic architecture of a web application and depicts where HTTP sits:

Figure 4 shows the HTTP Architecture

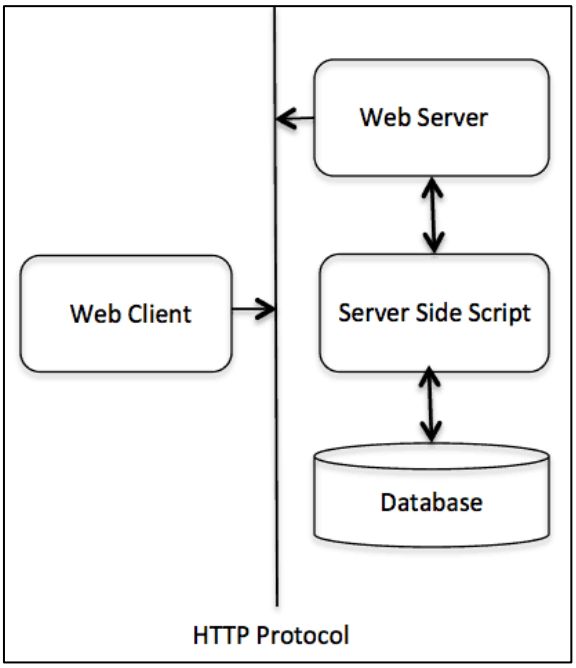



Figure 4: showing the HTTP Architecture

The HTTP protocol is a request/response protocol based on the client/server based architecture where web browsers, robots and search engines, etc. act like HTTP clients, and the Web server acts as a server.

**Client**

The HTTP client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a TCP/IP connection.

**Server**

The HTTP server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta information, and possible entity-body content.

Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses

**HTTP Request / Response**

Communication between clients and servers is done by **requests** and **responses**:

1. A client (a browser) sends an **HTTP request** to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP response** (output) to the browser
5. The client (the browser) receives the response

## 1.4 Uniform Resource Locator

A client that wants to access a web page needs the address. To facilitate the access of documents distributed throughout the world HTTP uses locator.

**DEFINATION**

“The Uniform Resource Locator (URL) is a standard for specifying any kind of information on the internet .the URL defines four things: protocol, host computer, port and path”

Figure 5 shows the syntax of URL

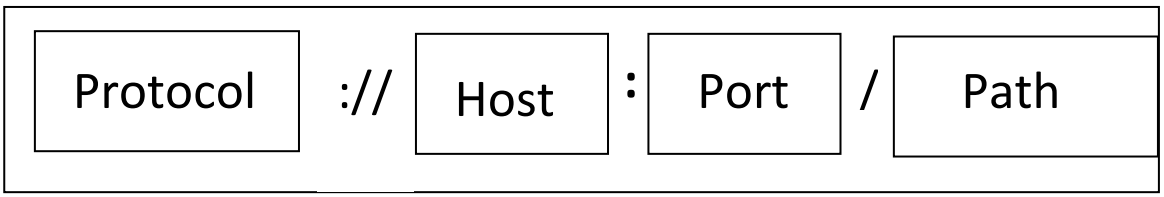



Figure 5: showing the URL Syntax

The **Protocol** is the client/server program used to retrieve the document .many different protocol can retrieve a document; among them FTP or HTTP .the most common today is HTTP.

The **Host** is the computer on which the information is located, although the name of the computer can be an alias .web pages are usually stored in computer and computers are given alias names that usually begin with the character “WWW” .this is not mandatory, however, as the host can be any name given to the computer that hosts the web page.

The **URL** can optionally contain the port number of the server .if the port is included, it is inserted between the host and the path, and it is separated from the host by a colon.

The **Path** is the pathname of the file where the information is located. Note that the path can itself contain slashes that, in the UNIX operating system .separate the directories from subdirectories and files.

## 1.5 Domain Name and IP Address

### Domain Name

Domain Name System, a system for naming computers and network services that is organized into a hierarchy of domains.It is used to locate computers and services through user-friendly names.

Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address.

The domain name system (DNS) is the way that Internet domain names are located and translated into Internet Protocol addresses. A domain name is a meaningful and easy-to-remember "handle" for an Internet address.

The following figure shows a basic use of DNS, which is finding the IP address of a computer based on its name.

Figure 6 shows How DNS Works

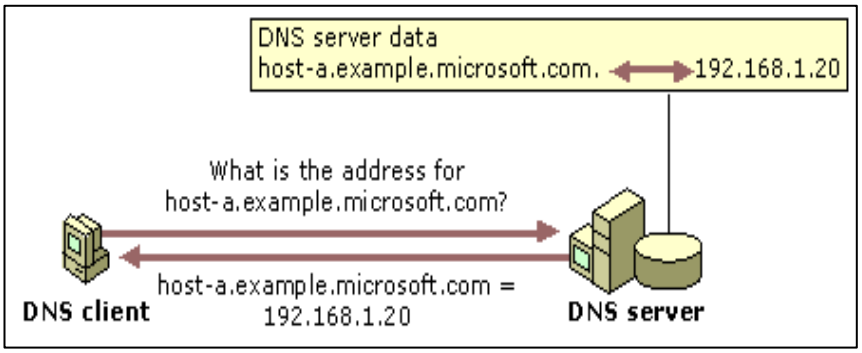



Figure 6: showing How DNS Works

In this example, a client computer queries a DNS server, asking for the IP address of a computer configured to use *host-a.example.microsoft.com* as its DNS domain name.

Because the DNS server is able to answer the query based on its local database, it replies with an answer containing the requested information, which is a host (A) resource record that contains the IP address information for *host-a.example.microsoft.com*.

Two ways how these domain names are divided

### 1. Geographical

| Domain Extension | Country |
|---|---|
| .IN | India |
| .AU | Australia |
| .CN | China |
| .EG | Egypt |
| .BR | Brazil |
| .PK | Pakistan |
| .NP | Nepal |
| .NZ | New Zealand |

### 2. Non- Geographical

| DNS Domain Name | Type of Organization |
|---|---|
| .com | Commercial organizations |
| .edu | Educational institutions |
| .org | Non-profit organizations |
| .net | Networks (the backbone of the Internet) |
| .gov | Non-military government organizations |
| .mil | Military government organizations |

## IP address (Internet Protocol Address)

Every computer on network requires an IP address to communicate with other computer.

Each computer on the Internet **known as a host** has at least one IP address that uniquely identifies it from all other computers on the Internet.

An IP is a 32-bit number comprised of a host number and a network prefix, both of which are used to uniquely identify each node within a network.

When you send or receive data (for example, an e-mail note or a Web page), the message gets divided into little chunks called **packets.**

**IP is a connectionless protocol**, which means that there is no continuing connection between the end points that are communicating. Each packet that travels through the Internet is treated as an independent unit of data without any relation to any other unit of data.

The most widely used version of IP today is Internet Protocol Version 4 (IPv4). However, IP Version 6 (IPv6)

**The Format of an IP Address**

The format of an IP address is a 32-bit numeric address written as four numbers separated by periods and **each part is known as octet**. Each number can be zero to 255.

Each octet is of 8 bits.

**Example of an IP address: 192.168.1.1**

The first part of an Internet address identifies the network on which the host resides, while the second part identifies the particular host on the given network. This creates the two-level addressing hierarchy.

All hosts on a given network share the same network prefix but must have a unique host number. Similarly, any two hosts on different networks must have different network prefixes but may have the same host number.

IP addresses are divided in five class.

Class A addresses range from  1-126

Class B addresses range from 128-191

Class C addresses range from 192-223

Class D addresses range from 224-239

Class E addresses range from 240-254

0 is reserved and represents all IP addresses;

127 is a reserved address and is used for loop back tasting:

255 is a reserved address and is used for broadcasting purposes.

## 1.6 Web Browsers

Web browser, a browser is a software application used to locate, retrieve and display content on the World Wide Web, including Web pages, images, video and other files.

Client/server model, the browser is the client run on a computer that contacts the Web server and requests information. The Web server sends the information back to the Web browser which displays the results on the computer or other Internet-enabled device that supports a browser.

The first Web browser with a graphical user interface was Mosaic, which appeared in 1993.Common web browsers include Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, and Apple Safari.

## 1.7 Web Pages

A web page is a single hypertext document available on World Wide Web (WWW). *It is composed of HTML elements and displayed on the user's browser such as* **Mozilla, Firefox**, **Chrome**, etc. **It is also referred to as "**Page."

### What is a Webpage?

A webpage is a document written in HTML and can be viewed on any web browser. It is contained within the web server, which can be accessed by entering the URL for that web page, and once it is loaded, it appears on the user's web browser. Each webpage is linked with a unique URL; hence two pages cannot have the same URL.

A webpage may contain ***text, links for other pages, graphics, videos, etc***. Moreover, it is mainly used to provide information to the user in text, images, etc.

The WWW or Internet contains millions of web pages, and daily, a lot is being added. Tim Berners-Lee developed **the first webpage.**

### Characteristics of a Web Page

Following are some characteristics of a Web page:

- A simple webpage can be created very quickly.
- It takes very little time to create a webpage compared to a Website.
- A web page and a website should be compatible with any device, such as Mobile, Desktop, Laptop, etc.
- The search engine provides a web page through a link, and when a user clicks on that link, it is redirected to the webpage of a website.
- A webpage can have any type of information including videos, and audios.
- It can be made up of only HTML(Hypertext Markup Language), or CSS, or JavaScript for dynamic and attractive behavior.

## Difference between a Webpage and a Website

| **Website** | **Webpage** |
|---|---|
| A website is a collection of different web pages that are linked together with hyperlinks. | A webpage is a single hypertext document. |
| It consists of more than one webpage. | It is a single document that is displayed on the user's browser. |
| To develop a website, developers need more skills and more time compared to a webpage. | To develop a webpage, developers need basic HTML knowledge and less time. |
| A website is accessed through its domain name, and it does not include any extension in the URL. | A webpage is accessed through a unique URL with some extension. |
| It can contain information for different entities, such as Javatpoint.com, which contains information about different technologies. | It can contain information for a single entity, such as currently viewing a web page containing information about this page only. |
| It is a little challenging to create a perfect website and requires lots of programming. | It is very simple to create a webpage. |
| Some examples of the website are Myntra.com, Amazon.com, etc. | Some examples of Webpages are the currently viewing page, contact page, registration page, the home page, etc. |

## How does a Web Page Work?

A simple web page is created using HTML, which is a markup language. However, we can also use CSS and JavaScript to add more functionalities and make it more attractive.

It is created using HTML, hence containing different markup tags that specify how the data should be formatted on screen.

The webpage is contained within the webserver. To load this webpage, a client sends the request to the server, and generally, the browser is known as the client, which can request the page on the internet.

The web browser requests the page on the internet. Once it is responded to by the server, the browser interprets the markup tags and displays them on the user's screen in the correct format.

Figure 7 shows How Web Page Works

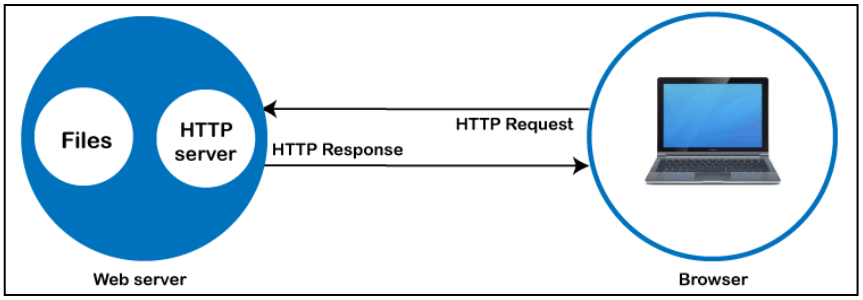



Figure 7: showing How Web Page Works

The browser sends the request for a page or a file via **an HTTP request.** The HTTP is the **Hypertext Transfer Protocol**, a network protocol that allows transferring hypermedia documents over the internet between a browser and server.

Once the request reaches the server, the HTTP server accepts the request, finds the requested page, and sends it back to the browser through **the HTTP response**. If a server is unable to find the requested page, it returns a **404 response**.

## Types

### Static Web page

**Static web pages** are also known as flat or stationary web page. They are loaded on the client's browser as exactly they are stored on the web server. Such web pages contain only static information. User can only read the information but can't do any modification or interact with the information.

Static web pages are created using only HTML. Static web pages are only used when the information is no more required to be modified.

Figure 8 shows How Static Web Page Works

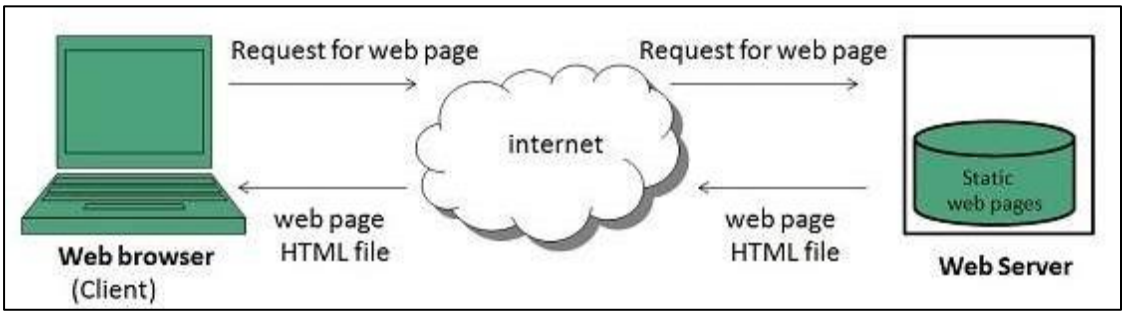



Figure 8: showing How Static Web Page Works

## Dynamic Web page

**Dynamic web page** shows different information at different point of time. It is possible to change a portaion of a web page without loading the entire web page. It has been made possible using **Ajax** technology.

### Server-side dynamic web page

It is created by using server-side scripting. There are server-side scripting parameters that determine how to assemble a new web page which also include setting up of more client-side processing.

### Client-side dynamic web page

It is processed using client side scripting such as JavaScript. And then passed in to **Document Object Model (DOM).**

Figure 9 shows How Dynamic Web Page Works

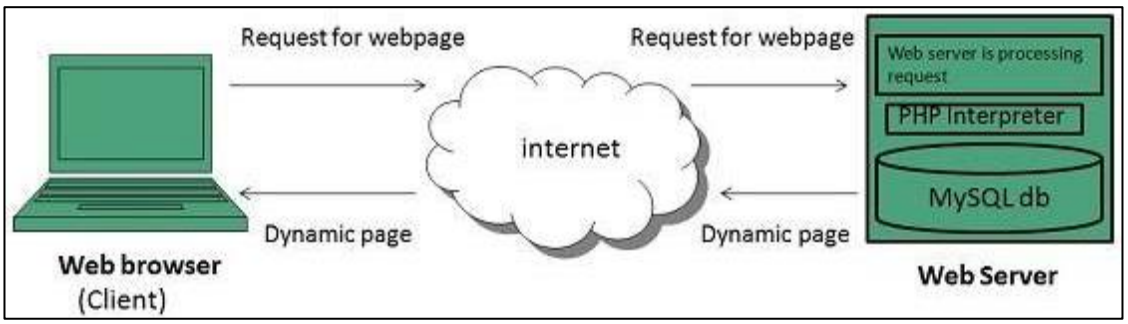



Figure 9: showing How Dynamic Web Page Works

# 1.8 Use of Cookies to Store Information

**Cookies** are text files with small pieces of data — like a username and password — that are used to identify your computer as you use a computer network. Specific cookies known as HTTP cookies are used to identify specific users and improve your web browsing experience.

Data stored in a cookie is created by the server upon your connection. This data is labeled with an ID unique to you and your computer.

When the cookie is exchanged between your computer and the network server, the server reads the ID and knows what information to specifically serve to you.

## What are HTTP Cookies?

**HTTP cookies,** or internet cookies, are built specifically for Internet web browsers to track, personalize, and save information about each user's session. A "session" just refers to the time you spend on a site.

Cookies are created to identify you when you visit a new website. The web server — which stores the website's data — sends a short stream of identifying info to your web browser.

## What Are Cookies Used For?

Websites use HTTP cookies to streamline your web experiences. Without cookies, you'd have to login again after you leave a site or rebuild your shopping cart if you accidentally close the page. Making cookies an important a part of the internet experience.

Based on this, you'll want to understand why they're worth keeping — and when they're not.

Here's how cookie are intended to be used:

1. **Session management**. For example, cookies let websites recognize users and recall their individual login information and preferences, such as sports news versus politics.

2. **Personalization**. Customized advertising is the main way cookies are used to personalize your sessions. You may view certain items or parts of a site, and cookies use this data to help build targeted ads that you might enjoy.

3. **Tracking**. Shopping sites use cookies to track items users previously viewed, allowing the sites to suggest other goods they might like and keep items in shopping carts while they continue shopping.

## 1.9 Summary

In this chapter you have learned about:

- WWW
- The Web System Archtecture
- About HTTP
- Uniform Resource Locator
- Domain Name and IP Address
- Web Browsers
- Web Pages
- Use of Cookies to Store Information

# Unit 2: Hypertext Markup Language (HTML)

## Unit Structure


2.1. Introduction to HTML

2.2. HTML Editors

2.3. Basic structure of HTML Document

2.4. Getting started with HTML

2.5. Displaying Web page in Web Browser

2.6. Modifying the Background of HTML page

2.7. Specifying Metadata of HTML page

2.8. Summary

Hypertext Markup Language(HTML) is the most commonly used markup language for creating Web pages.HTML describes the structure of a Web page.HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages.

Hypertext is called ‘hyper’ because the navigation through the pages using the hypertext is not linear.It means that if you click the hypertext present on a Web page, the Web page you are redirected may be on a next page or any page on the same website.World Wide Web Consortium (W3C) is an organization, which defines new specifications for HTML and its responsible for updating the language.

## 2.1 Introduction to HTML

HTML is markup language used to create Web pages, a Web page is a document stored in the World Wide Web (WWW) or Web.Each HTML page contains the data to be included in the page, between the HTML tags.A Web browser understands these tags and displays the corresponding Web pages.

HTML allows you to format, arrange and group text, display text as link and add images and multimedia to a Web page. It also allows you to create controls, create and work with style sheets and embed scripting language code (such as JavaScript code) into a Web page.

HTML was created by Tim Berners-Lee in 1991. The first-ever version of HTML was HTML 1.0, but the first standard version was HTML 2.0, published in 1999.

Figure 1 shows HTML Versions

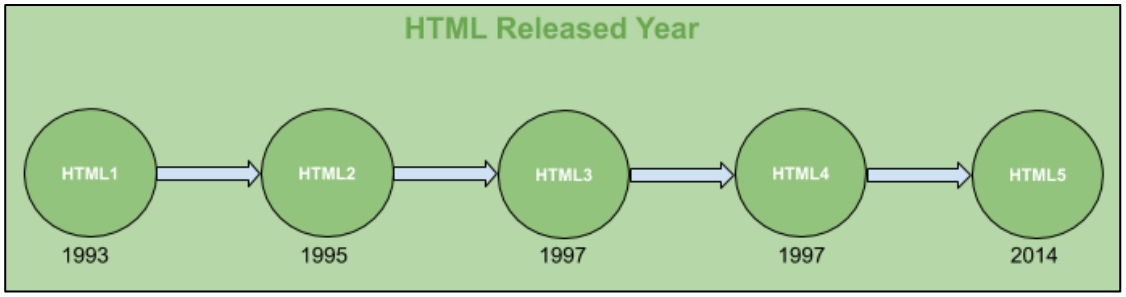


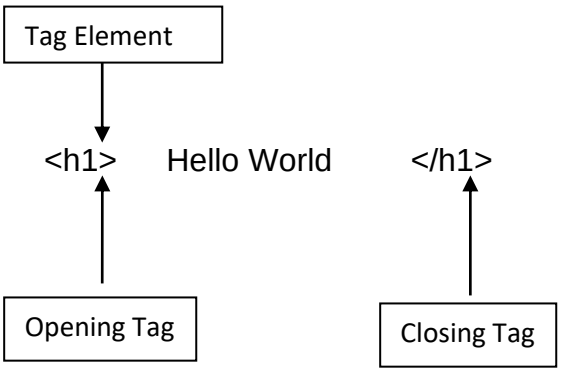
Figure 1: showing HTML Versions

## Elements and Tags

HTML uses predefined tags and elements which tell the browser how to properly display the content. Remember to include closing tags. If omitted, the browser applies the effect of the opening tag until the end of the page.

An HTML element is defined by a start tag, some content, and an end tag:

<tagname> Content goes here... </tagname>




**Note:** Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

The anatomy of our element is:

- **The opening tag:** This consists of the name of the element (in this example, *h1* for paragraph), wrapped in opening and closing angle brackets. This opening tag marks where the element begins or starts to take effect.
- **The content:** This is the content of the element. In this example, it is the heading text.
- **The closing tag:** This is the same as the opening tag, except that it includes a forward slash before the element name. This marks where the element ends. Failing to include a closing tag is a common beginner error that can produce peculiar results.

## ATTRIBUTE

An Attribute is a special word used inside tag to specify additional informationto tag such as color, alignment etc.

Some examples of HTML tags with attributes are:

E.g <body bgcolor = “red”>

**Features of HTML**

- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to the text.
- It is a markup language.

**Advantages of HTML**

- HTML is used to build websites.
- It is supported by all browsers.
- It can be integrated with other languages like CSS, JavaScript, etc.

**Disadvantages of HTML**

- HTML can only create static web pages. For dynamic web pages, other languages have to be used.
- A large amount of code has to be written to create a simple web page.
- The security feature is not good.

## 2.2 HTML Editors

HTML editor is software used for writing code in HTML,which is used for structuring and creating websites.Essentially, a HTML editor either converts text and layout interface input into actual HTML code, or allows users to scan HTML code to look for appropriate syntax in design. The former type of editor can be called a WYSIWYG or 'what you see is what you get’ editor where the visual platform allows individuals to effectively code in HTML, without knowing HTML.

### Types of HTML Editors

There are broadly two types of HTML Editors:

There are two common forms of HTML editors: **text and WYSIWYG (What You See Is What You Get).**

Figure 2 shows Types of HTML Ediotrs

Figure 2: showing Types of HTML Editors

### 1. Textual HTML Editor

These are text-based editors where the developers can write their codes and compile them. The code appears in the same manner we write it, thus it requires basic knowledge of HTML.

Some of these editors also provide features of making a project, managing all the files related to the web, etc. Examples of HTML Text editors include-Notepad++, VSCode,Sublime Text.

### 2. WYSIWYG HTML Editor

'What you see is what you get' is its full form. WYSIWYG are editors that provide the preview of the output of the source code i.e. as it would appear on a browser. There is a drag and drop feature available in most of them that eases the handling.

It does not require any hardcore knowledge of HTML, thus enabling non-technicals to easily develop websites. Examples include-Adobe Dreamweaver, Microsoft FrontPage etc.

HTML can be written in simple text editors such as Notepad or TextEdit

## 2.3 Basic Structure of HTML Document

An HTML Document is mainly divided into two parts:

- **HEAD**: This contains the information about the HTML document. For Example, Title of the page, version of HTML, Meta Data etc.
- **BODY**: This contains everything you want to display on the Web Page.

Figure 3 shows Basic Structure of HTML Document

Figure 3: showing Basic Structure of HTML Document

**The html Tag**

The <html> element marks the beginning and the end of the HTML document. All HTML content has to be within the start and end tag (<html>...</html>).

**The head and title Tags**

The <head> tag contains information about the document, including its title, and documentdescription.The <head> element contains all the page information which is not shown in the browser. For instance, the <title> element, metatags and more elements.

The <title> element contains the page title, which will be shown in the web browsers title-bar. The title should match the content of the HTML page, meaning it should tell what the page contains. The <title> element can be defined individually for every single HTML page.

**The body tag**

The <body> element contains the visible page content. It can contain text, images, graphs, forms, tables etc.

## 2.4 Getting started with HTML

Before you create an HTML document, make sure that you have an editor, such as Dreamweaver or Notepad and a Web browser, such as Internet Explorer, Firefox or Google Chrome. The most common editor and the browser are Notepad and Internet Explorer respectively.

### Opening the Notepad Text Editor

HTML code can be written in any of the text editors such as Notepad or Dreamweaver. In our case, we use Notepad text editor.

### Writing HTML code

All the text of an HTML document is written inside the HTML tags. Tags are instructions used by an HTML document to provide the display information of the content written inside them. These instructions are interpreted by the Web browser.

Write or copy the following HTML code into Notepad:

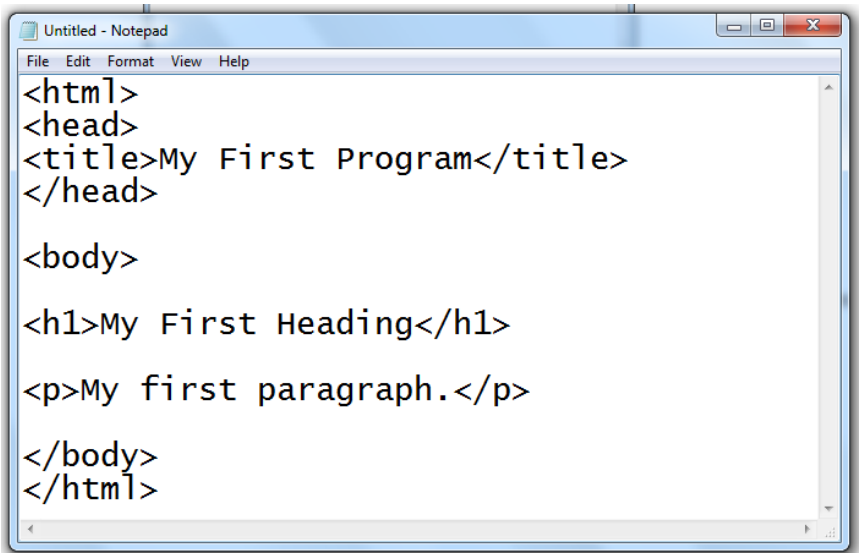



Figure 6: Displaying the Notepad with tags

### Saving an HTML code

When you save a Web page, an image or other files on a website, there are certain conversions that you should follow. For example, you should give the file a unique name and the appropriate file extension.

An HTML document (file) is always saved with the .html or .htm extension.

## 2.5 Displaying Web Page in Web Browser

A Web browser performs the function of reading HTML files and displaying them as Web pages. The tags written inside an HTML file are used to interpret the content of the Web page. The browser does not display the HTML tags.

## 2.6 Modifying the Background of an HTML Web Page

Background of an HTML Web page can be modified in many ways, for example by adding a background color to the Web page or by adding a background image to the Web page

### Adding a Background Color to the Web page

You can add either a predefined color value or a color name as the background color for an HTML Web page using the bgcolor attribute of the <body> element.

Table 1 lists color values for each of the W3C predefined colors.

| **Table 1 color values for each of the W3C predefined colors** | |
|---|---|
| Color | Value |
| silver | #c0c0c0 |
| gray | #808080 |
| maroon | #800000 |
| olive | #808000 |
| lime | #00ff00 |
| aqua | #00ffff |
| teal | #008080 |

| | |
|---|---|
| navy | #000080 |
| fuchsia | #ff00ff |
| purple | #800080 |
| indianred | #cd5c5c |
| lightcoral | #f08080 |
| red | #ff0000 |
| Yellow | #ffff00 |

### Adding a Background Image to the Web page

Similar to modifying the background of an HTML Web page by adding a backgroung color to it, you can also modify the background of an HTML Web page by adding a background image to it.

## 2.7 Specifying Metadata of HTML page

HTML lets you specify metadata - additional important information about a document in a variety of ways. It specifies page description, keywords, copyright, language, author of the documents, etc.

The metadata does not display on the webpage, but it is used by search engines, browsers and other web services which scan the site or webpage to know about the webpage.

With the help of meta tag, you can experiment and preview that how your webpage will render on the browser.

The <meta> tag is placed within the <head> tag, and it can be used more than one times in a document.

### Adding Meta Tags to Your Documents

You can add metadata to your web pages by placing <meta> tags inside the header of the document which is represented by <head> and </head> tags. A meta tag can have following attributes in addition to core attributes (Table 2)

| Table 2 Attributes, value and Description of Meta tag | | |
|---|---|---|
| **Attribute** | **Value** | **Description** |
| charset | character_set | It specifies the character encoding of an HTML page. You can set the value as charset=”utf-8” |
| content | text | It contains the value of the attribute "name" and "http-equiv" in an HTML document, depending on context. |
| http-equiv | ○ Content-type<br>○ default-style<br>○ refresh | It specifies the Information given by the web server about how the web page should be served. |
| name | ○ application-name<br>○ author<br>○ description<br>○ generator<br>○ keywords | It specifies the name of document-level metadata. |
| scheme | format/URL | It specifies the scheme in which metadata is described. (Not Supported in HTML5) |

Now, let’s take examples of each attributes showing in (table 3)

| Table 3 Attribute description with example | |
|---|---|
| **Description** | **Example** |
| **Define keywords for search engines:** | <meta name="keywords" content="HTML, CSS, JavaScript"> |
| **Define a description of your web page:** | <meta name="description" content="Free Web tutorials for HTML and CSS"> |
| **Define the author of a page:**<br>**Refresh document every 30 seconds:** | <meta name="author" content="Khyati Shah"><br><meta http- equiv="refresh" content="30"> |
| **Setting the viewport to make your website look good on all devices:** | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |

## 2.8 Summary

In this chapter you have learned about:

- Introduction to HTML
- HTML Editors
- Basic structure of HTML Document
- Displaying Web page in Web Browser
- Modifying the Background of HTML page
- Metadata of HTML page

# Unit 3: Working with Text

## Unit Structure


3.1. Adding plain Text to an HTML Web page

3.2. Adding Text in New Line

3.3. Creating Headings on a Web page

3.4. Creating a Paragraph

3.5. Creating a Horizontal Ruler Line

3.6. Inserting the <pre> tag

3.7. Formatting Tags

3.8. Aligning the Text

3.9. Grouping the Text

3.10. Indenting Quotations

3.11. Working with Character Entities

3.12. Commenting the Text

3.13. Summary

Hypertext Markup Language (HTML) describes the structure of text-based information on a Web page by marking the text as heading and paragraphs.The basic text-fromatting tags in HTML are <br>, <p> and <hr>, which are used to add text in a new line, create paragraphs, and insert horinzontal ruler, respectively.HTML also provides tags to apply various styles to the text such as <font>, <b, <i>, <strike>and <pre>.HTML provides elements such as subscript, superscript and blockquote that can be used to enhance the look of the text on a Web page.

Apart from this, characters such as graterer than(>) , less than(<) ,quotation mark(“) etc., which are reserved characters in HTML , can be displayed on a Web page using the character entities.In this chapter, you learn how to work with text in HTML.

## 3.1 Adding plain Text to an HTML Web page

Text can be added to an HTML document by typing between the <body> tags of HTML document. If you enter plain text in an HTML document, the Web browser displays the text in its default font, color and size.

HTML recognizes only single spaces between characters. This means if there are more than single space spaces between characters, HTML does not recognize it. Even if you type the text in a different line in Notepad, HTML considers it as a single space and displays the text in the same line on the Web page.

## 3.2 Adding Text in a new line

When you're writing HTML, you often need to insert line breaks. A line break is essential in addresses, poems, or when text exceeds the available browser width.

In HTML, the <br> tag is used for line break. It is an empty tag i.e. no need to add an end tag. Writing <br> tag is perfectly fine. If you place the <br> tag in the HTML code, then it works the same as pressing the enter key in a word processor.

## 3.3 Creating Headings ona Web page

A HTML heading can be defined as a title or a subtitle which you want to display on the webpage. When you place the text within the heading tags, it is displayed on the browser in the bold format and size of the text depends on the number of heading.

There are six different HTML headings which are defined with the <h1> to <h6> tags, from highest level h1 (main heading) to the least level h6 (least important heading).

h1 is the largest heading tag and h6 is the smallest one.The heading tags are pairs with an opening tag and a closing tag.

**While displaying any heading, browser adds one line before and one line after that heading.**

## 3.4 Creating a Paragraph

HTML paragraph or HTML p tag is used to define a paragraph in a webpage. Let's take a simple example to see how it work. It is a notable point that a browser itself add an empty line before and after a paragraph. An HTML <p> tag indicates starting of new paragraph.

## 3.5 Creating a Horizontal Ruler Line

HTML <hr> tag is used to draw a horizontal line. It is also called a Horizontal Rule in HTML.

## 3.6 Inserting <Pre> Tag

The **HTML <pre> tag** is used *to specify pre formatted texts*. Texts within <pre>.....</pre> tag is displayed in a fixed-width font. Usually it is displayed in Courier font. It maintains both space and line break.

It is widely used to display language examples e.g. Java, C#, C, C++ etc because it displays the code as it is typed.

# 3.7 Formatting Tags

## 1.  Tag

HTML bold tag is represented by  tag.

**HTML  tag** is used *to display the written text in bold format*. It is strictly a presentational element. If you want to show your text in bold letters and not have real semantic meaning, then put it within ..... tag.

## 2. <i> Tag

HTML <i> tag is used to represent a part of text in a different voice from the surrounding text. The content within <i> tag usually renders in italic type on the browser. It can be useful to represent some technical terms, phrase, fictional character thoughts, etc.

## 3. <u> Tag

HTML <u> tag is used to define a span of inline text with a non-textual annotation. It rendered as an solid underlined text,

## 4. <sup> Tag

HTML <sup> tag is termed as a superscript tag which is used to define superscript text. The text within <sup> tag appears with an upper baseline and renders with smaller font size than surrounding text.

The <sup> tag is useful for defining Mathematical formulas and footnotes.

## 5. <sub> Tag

HTML <sub> tag is termed as Subscript tag and which is used to define subscript text. The text within <sub> renders with a lower baseline and with a smaller font than surrounding text font.

The <sub> tag is useful for presenting mathematical formula and chemical formulas such as $H_2O$.

### 6. <big> Tag

HTML <big> tag was used to increase the text font size one level bigger than the document's base font size or surrounding text size, such as small to medium, medium to large, etc.

### 7. <small> Tag

HTML <small> tag makes text font by one size smaller than the document?s base font size (Such as large to medium, medium to small, etc.)

In HTML5, <small> tag is used for identifying secondary importance such as copyright, side comments, and legal notices.

### 8. <strike> Tag

HTML <strike> tag was used to strike a line through the text, contained within it.

### 9. <tt> Tag

HTML <tt> tag was used to define text in monospaced font or fixed-width fonts so that it would render as teletype, text-only screen, or line printer on the browser.

## 3.8 Aligning The Text

The align Attribute in HTML is used to is used to specify the alignment of text content of The Element. this attribute is is used in all elements.

**Attribute Values:**

- **left**: It sets the text left-align.
- **right**: It sets the text right-align.
- **center:** It sets the text center-align.
- **justify**: It stretch the text of paragraph to set the width of all lines equal.

## 3.9 Grouping The Text

In Html you can group several contents together to make sections or subsections in a page. Grouping content makes it easier to manage the content. It is easier for both the programmer and readers as it looks good while we group similar contents together.

In Html the tags available for grouping contents are  and 

### Grouping using Div element

You can use the  tag to group elements in html. The  tag is the most used grouping element in html. The  tag stretches to the whole page or the whole container.

The contents before and after the Div element will be displayed in separate lines.

### Grouping using Span element

Span is an inline grouping element. The text written before and after the ….element all will be displayed in a single line.

## 3.10 Indenting Quotations

HTML <blockquote> tag is used to define a block of text which is quoted from another source. The Browser usually displays the content within <blockquote> tag as indented text.

If you want to insert a long quote then use <blockquote> and for short or inline quote use <q> tag.

## 3.11 Working With Character Entities

Some characters are reserved in HTML and they have special meaning when used in HTML document. For example, you cannot use the greater than and less than signs or angle brackets within your HTML text because the browser will treat them differently and will try to draw a meaning related to HTML tag.

HTML processors must support following five special characters listed in the table that follows.

| Symbol | Description | Entity Name | Number Code |
|---|---|---|---|
| " | quotation mark | " | " |
| ' | apostrophe | ' | ' |
| & | ampersand | & | & |
| < | less-than | < | < |
| > | greater-than | > | > |

## 3.12 Commenting the Text

Comments are some text or code written in your code to give an explanation about the code, and not visible to the user. Comments which are used for HTML file are known as HTML comments. Anything written between these tags will be ignored by the browser, so comments will not be visible on the webpage.

Comments of any code make code easy to understand and increase readability of code.

## 3.13 Summary

In this chapter you have learned about:

- Adding plain Text to an HTML Web page
- Adding Text in New Line
- Creating Headings on a Web page
- Creating a Paragraph
- Creating a Horizontal Ruler Line

- Inserting the <pre> tag
- Formatting Tags
- Aligning the Text
- Grouping the Text
- Indenting Quotations
- Character Entities
- Commenting the Text

# Unit 4:  Working with Lists, Tables, and Frames

## Unit Structure


4.1. Working with Lists

4.2. Working with Tables

4.3. Working with Frames

4.4. Summary

# 4.1 Working with Lists

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements. Lists may contain −

- <ul> − An unordered list. This will list items using plain bullets.
- <ol> − An ordered list. This will use different schemes of numbers to list your items.
- <dl> − A definition list. This arranges your items in the same way as they are arranged in a dictionary.
- Nested list – Combine one or more list

## 1) HTML Unordered Lists

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML <ul> tag. Each item in the list is marked with a bullet.

Let’s perform steps to create Unordered list on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 1, in the document

Listing 1: Adding Unordered list to an HTML Web page

```
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
<ul>
<li>Beetroot</li>
<li>Ginger</li>
```

<li>Potato</li>

<li>Radish</li>

</ul>

</body>

</html>

---

2. Save the document with appropriate name and .html extension. Here we have named HTML document as unorderliist.html.
3. Open the HTML document in browser, as shown in figure 1

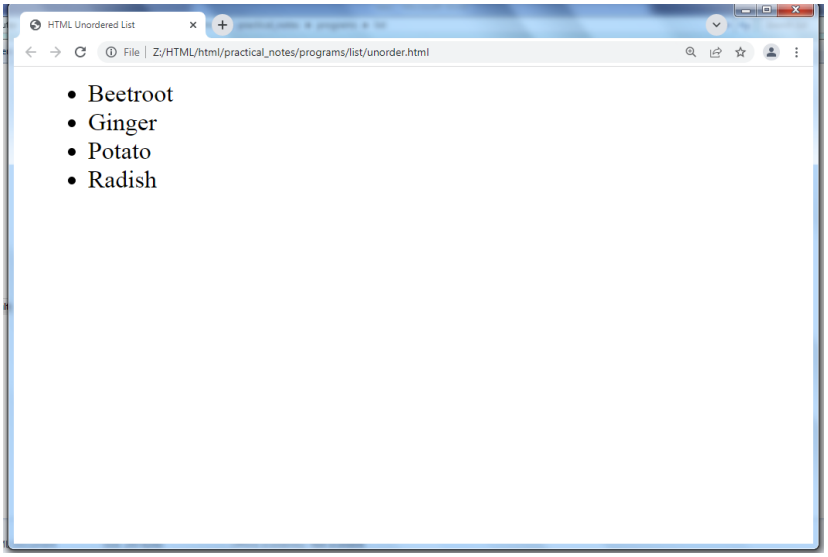



Figure 1: Displaying unordered list on web page

## Attribute

### 1. Type

You can use type attribute for <ul> tag to specify the type of bullet you like. By default, it is a disc. Following are the possible options −

<ul type = "square">

<ul type = "disc">

<ul type = "circle">

## 2) HTML Ordered Lists

If you are required to put your items in a numbered list instead of bulleted, then HTML ordered list will be used.

This list is created by using <ol> tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with <li>.

Let's perform steps to create ordered list on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 5, in the document

Listing 5: Adding ordered list to an HTML Web page

```
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as orderliist.html.
3. Open the HTML document in browser, as shown in figure 5

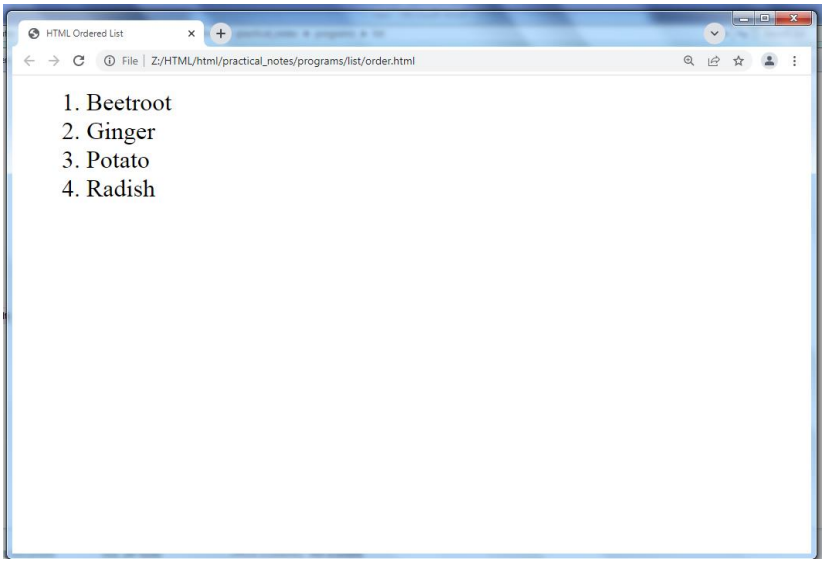



Figure 5: Displaying ordered list on web page

## Attributes

### 1. Type

You can use type attribute for <ol> tag to specify the type of numbering you like. By default, it is a number. Following are the possible options

<ol type = "1"> - Default-Case Numerals.

<ol type = "I"> - Upper-Case Numerals.

<ol type = "i"> - Lower-Case Numerals.

<ol type = "A"> - Upper-Case Letters.

<ol type = "a"> - Lower-Case Letters.

### 2. Start

You can use start attribute for <ol> tag to specify the starting point of numbering you need. Following are the possible options −

<ol type = "1" start = "4">  - Numerals starts with 4.

<ol type = "I" start = "4">  - Numerals starts with IV.

<ol type = "i" start = "4">  - Numerals starts with iv.

<ol type = "a" start = "4">  - Letters starts with d.

<ol type = "A" start = "4">  - Letters starts with D.

### 3. Reversed

This is a Boolean attribute of HTML <ol> tag, and it is new in HTML5 version. If you use the reversed attribute with tag then it will numbered the list in descending order (7, 6, 5, 4.....1).

## 3) HTML Definition Lists

The definition list is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following three tags.

- <dl> − Defines the start of the list
- <dt> − A term
- <dd> − Term definition
- </dl> − Defines the end of the list

Let's perform steps to create definition list on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 10, in the document

Listing 9: Adding definition list to an HTML Web page

```
<html>

<head>

<title>HTML Definition List</title>

</head>

<body>

<dl>
```

```
<dt><b>HTML</b></dt>
<dd>This stands for Hyper Text Markup Language</dd>
<dt><b>HTTP</b></dt>
<dd>This stands for Hyper Text Transfer Protocol</dd>
</dl>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Definition list.html.
3. Open the HTML document in browser, as shown in figure 10

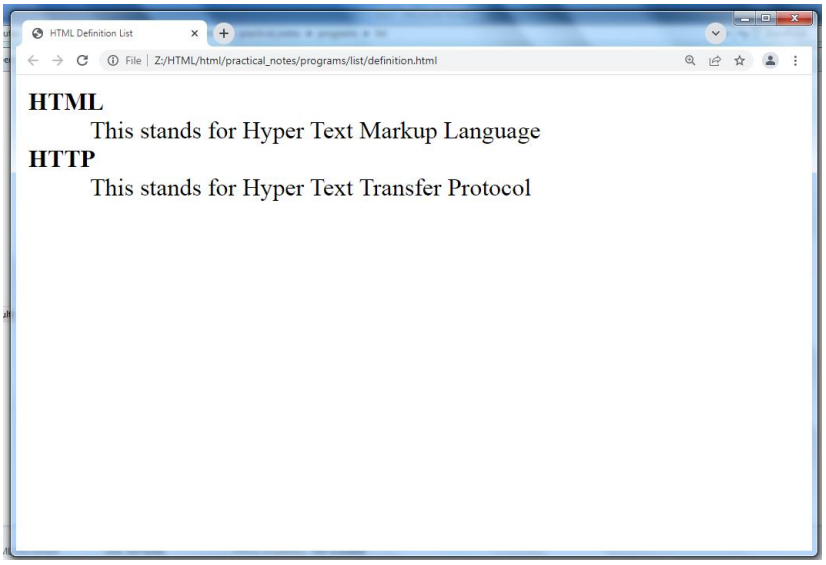



Figure 10: Displaying definition list on web page

## 4) Nested Lists

A nested list is a list inside another list. The following are different combination of nested list

1. Unordered List within Unordered List
2. ordered List within ordered List
3. Unordered List within ordered List
4. ordered List within Unordered List

Now, let’s see all above combination of nested list with example..

### 1. Unordered List within Unordered List

Let’s perform steps to create Unordered List within Unordered List on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 11, in the document

Listing 11: Adding Unordered List within Unordered List to an HTML Web page

```
<html>
<head>
<title>HTML Unordered List within Unordered List</title>
</head>
<body>
    <h2>An Unordered List within Unordered List </h2>
        <ul><!-- start of  main list-->
        <li>HTML</li>
        <li>CSS</li>
        <li>JavaScript
        <ul><!-- start of nested list-->
        <li>Angular</li>
        <li>React</li>
        <li>Vue</li>
        </ul><!--end of nested list-->
        </li>
        </ul><!--end of  main list -->
        </body>
        </html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Unordered List within Unordered List.html.
3. Open the HTML document in browser, as shown in figure 11

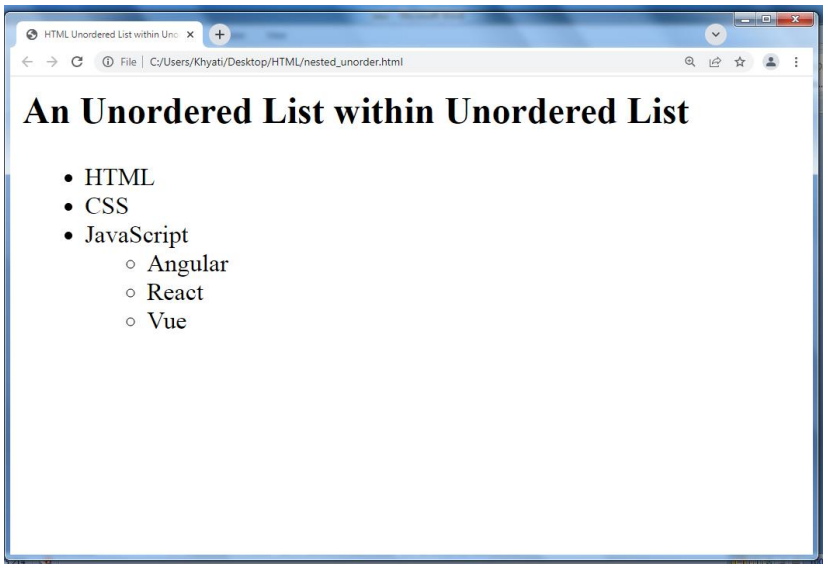



Figure 11: Displaying Unordered List within Unordered List on web page

## 2. ordered List within ordered List

Let's perform steps to create ordered List within ordered List on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 12, in the document

Listing 12: Adding ordered List within ordered List to an HTML Web page

```
<html>
<body>
<h2>An Ordered List within Ordered List </h2>
<ol>
<li>Coffee</li>
<li>Tea
<ol>
<li>Black tea</li>
<li>Green tea</li>
```

```
</ol>
</li>
<li>Milk
<ol>
<li>Hot</li>
<li>Cold</li>
</ol>
</li>
</ol>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as ordered List within ordered List.html.
3. Open the HTML document in browser, as shown in figure 12

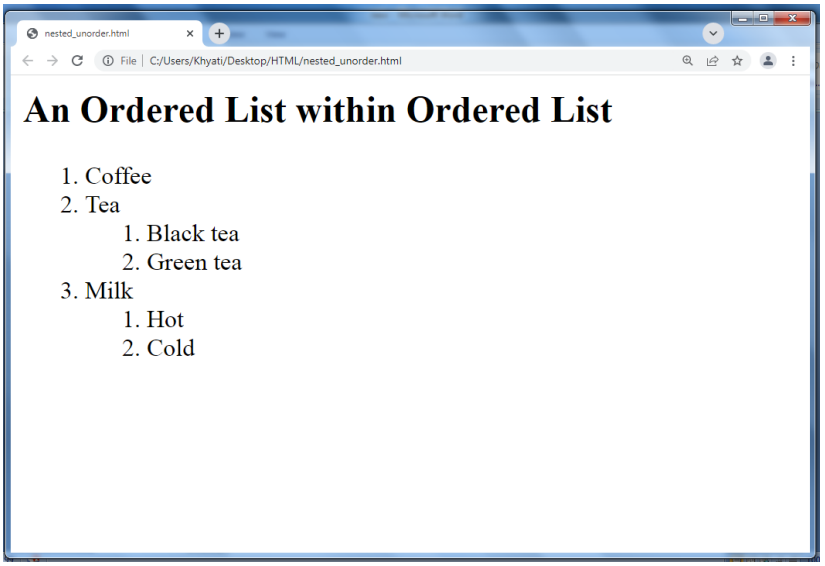



Figure 12: Displaying ordered List within ordered List on web page

### 3. Unordered List within ordered List

Let's perform steps to create Unordered List within ordered List on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 13, in the document

Listing 13: Adding Unordered List within ordered List to an HTML Web page

```
<html>
<body>

<h2>A Unordered List within Ordered List</h2>

<ol>
<li>Coffee</li>
<li>Tea
<ul>
<li>Black tea</li>
<li>Green tea</li>
</ul>
</li>
<li>Milk</li>
</ol>

</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Unordered List within ordered List.html.
3. Open the HTML document in browser, as shown in figure 13

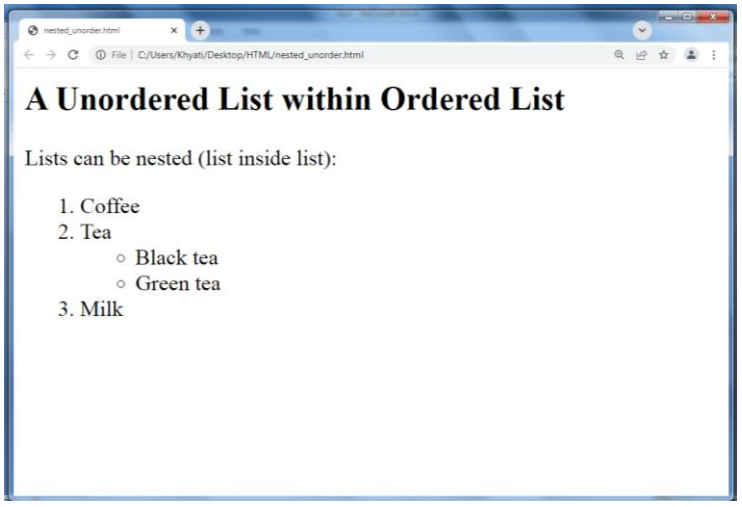


Figure 13: Displaying Unordered List within ordered List on web page

### 4. ordered List within Unordered List

Let's perform steps to create ordered List within Unordered List on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 14, in the document

Listing 14: Adding ordered List within Unordered List to an HTML Web page

```
<html>
<body>

<h2>A Ordered List within Unordered List</h2>

<ul>
<li>Coffee</li>
<li>Tea
<ol>
<li>Black tea</li>
<li>Green tea</li>
</ol>
</li>
<li>Milk</li>
</ul>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as ordered List within Unordered List.html.
3. Open the HTML document in browser, as shown in figure 14

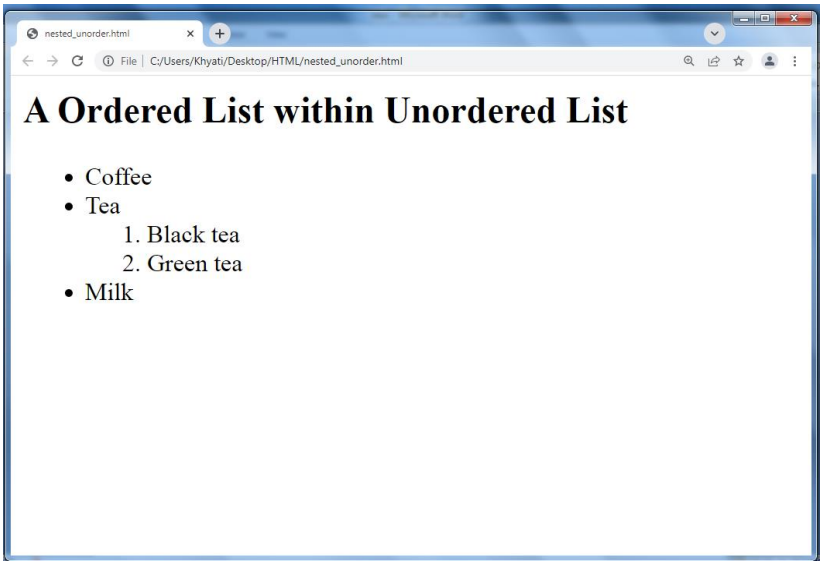



Figure 14: Displaying ordered List within Unordered List on web page

## 4.2 Working with Tables

The HTML tables are created using the <table> tag in which the <tr> tag is used to create table rows and <td> tag is used to create data cells. **The elements under <td> are regular and left aligned by default**

Let's perform steps to create Table on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 15, in the document

Listing 15: Adding Table to an HTML Web page

```
<html>

<head>

<title>HTML Tables</title>

</head>

<body>

<table border = "1">
```

```
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Table.html.
3. Open the HTML document in browser, as shown in figure 15

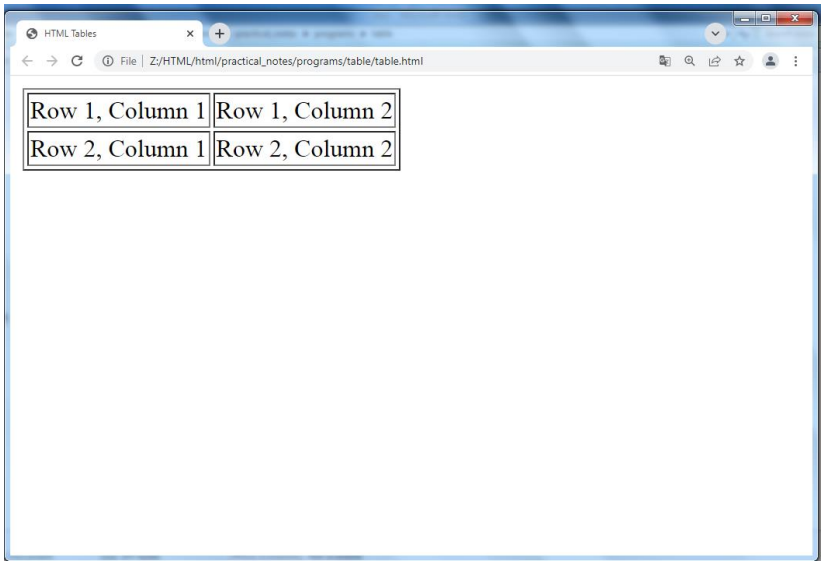



Figure 15: Displaying Table on web page

## Table Heading

Table heading can be defined using <th> tag. This tag will be put to replace <td> tag, which is used to represent actual data cell. Headings, which are defined in <th> tag are centered and bold by default.

## Attributes

### 1. Cellpadding and Cellspacing

There are two attributes called cellpadding and cellspacing which you will **use to adjust the white space in your table cells.** The **cellspacing attribute defines space between table cells,** while **cellpadding represents the distance between cell borders and the content within a cell.**

### 2. Colspan and Rowspan

You will use colspan attribute if you want to merge two or more columns into a single column. Similar way you will use rowspan if you want to merge two or more rows.

### 3. Table Backgrounds

You can set table background using one of the following two ways −

- **bgcolor attribute** − You can set background color for whole table or just for one cell.
- **background attribute** − You can set background image for whole table or just for one cell.

You can also set border color also using bordercolor attribute.

**Note** − The bgcolor, background, and bordercolor attributes deprecated in HTML5. Do not use these attributes.

### 4. Height and Width

You can set a table width and height using width and height attributes. You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

## Table Caption

The caption tag will serve as a title or explanation for the table and it shows up at the top of the table.

## Nested Tables

You can use one table inside another table. Not only tables you can use almost all the tags inside table data tag <td>.

Let's perform steps to create Nested Table on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 23, in the document

Listing 23: Adding Nested Table to an HTML Web page

```
<html>
<head>
<title>HTML Table</title>
</head>
<body>
<table border = "1" width = "100%">
<tr>
<td>
<table border = "1" width = "100%">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
```

```
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Nested_Table.html.
3. Open the HTML document in browser, as shown in figure 23

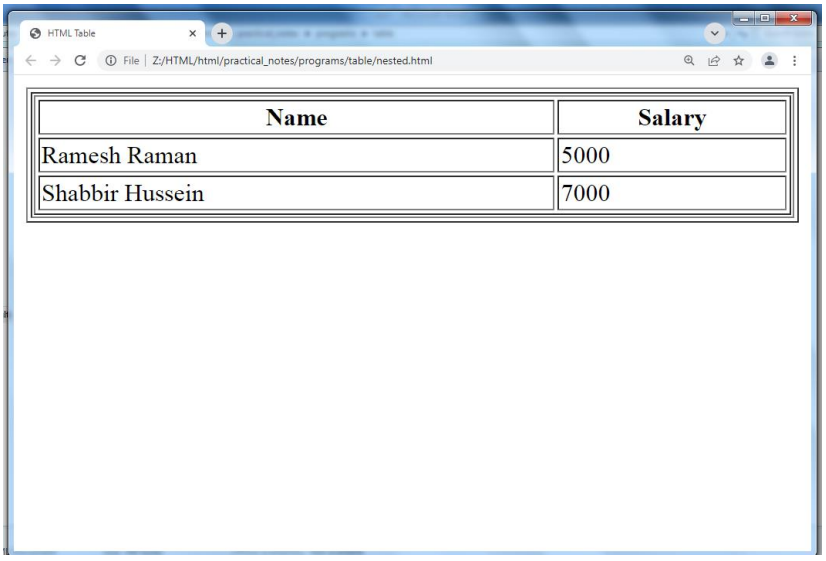



Figure 23: Displaying Nested Table on web page

# 4.3 Working with Frames

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document.

A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns

## Creating Frames

To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines, how to divide the window into frames.

The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames.

Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

## Creating Vertical Columns

To create a set of four vertical columns, we need to use the frameset element with the cols attribute.

The cols attribute is used to define the number and size of columns the frameset will contain.

In our case, we have four files to display, so we need four frames. To create four frames we need to assign four comma-separated values to the cols attribute.

To make things simple we're going to assign the value * to each of the frames, this will cause them to be automatically sized to fill the available space.

Let's perform steps to create Frame with Vertical Columns on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 24, in the document

Listing 24: Adding Frame with Vertical Columnsto an HTML Web page

```
<html>

<frameset cols="*,*,*,*">
```

```
<frame src="frame1.html">
<frame src="frame2.html">
<frame src="frame3.html">
<frame src="frame4.html">
</frameset>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Frame_verticalcolumns.html.
3. Open the HTML document in browser, as shown in figure 24

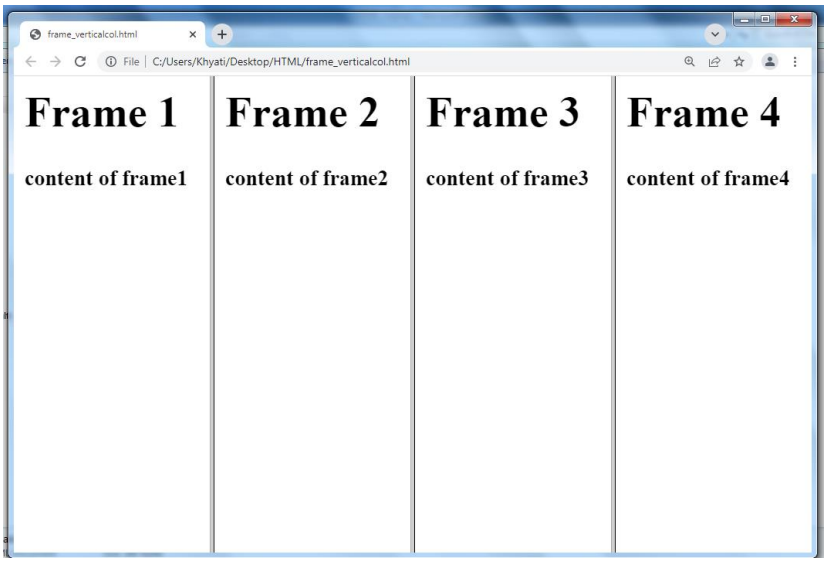



Figure 24: Displaying Frame with Vertical Columnson web page

## Creating Horizontal Rows

Rows of frames can be created by using the rows attribute rather than the cols attribute as shown in the HTML below.

Let's perform steps to create Frame with Horizontal Rows on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 25, in the document

Listing 25: Adding Frame with Horizontal Rows to an HTML Web page

```
<html>
<frameset rows="*,*,*,*">
<frame src="frame1.html">
<frame src="frame2.html">
<frame src="frame3.html">
<frame src="frame4.html">
</frameset>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Frame_Horizontal Rows.html.
3. Open the HTML document in browser, as shown in figure 24

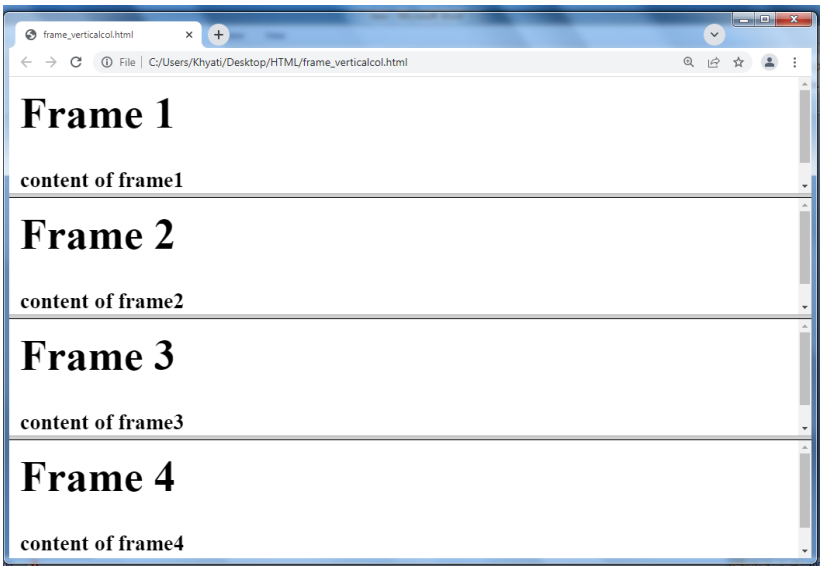



Figure 25: Displaying Frame with Horizontal Rows on web page

## Mixing Columns and Rows

Columns and rows of frames can both appear on the same webpageby nesting one frameset inside of another.

To do this, we first create a frameset and then nest a child frameset within the parent element.

Let’s perform steps to create Frame with nesting rows and columns on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 26, in the document

Listing 26: Adding Frame with nesting rows and columns to an HTML Web page

```
<frameset cols="*,*,*">
<frameset rows="*,*">
<frame src="frame1.html">
<frame src="frame2.html">
</frameset>
<frame src="frame3.html">
<frame src="frame4.html">
</frameset>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Frame_nesting rows and columns.html.
3. Open the HTML document in browser, as shown in figure 26

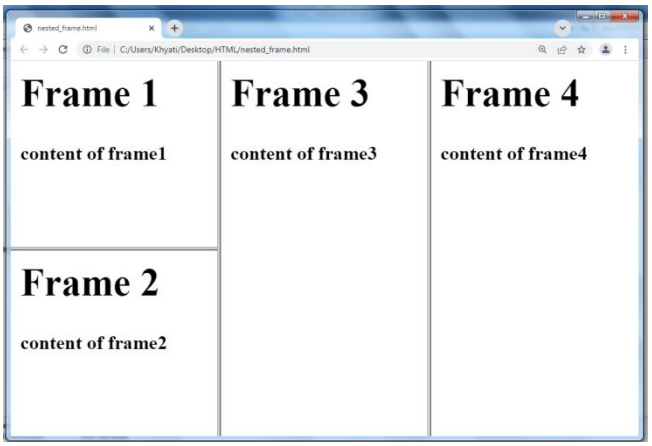



Figure 26: Displaying Frame with nesting rows and columnson web page

The nested frameset takes the place of the first frame within the parent element. The nested element can be placed in any position.

If we wanted the nested element to appear in the center position we would just rearrange the elements.

One more way to create a combination of rows and columns is to define a grid of columns and rows in a single frameset.

## 4.4 Summary

In this chapter you have learned about:

- List and Its Types
- Working with Tables
- Working with Frames

# Introduction to Web Designing

## BLOCK2: ADVANCED HTML AND HTML5

**CHAP 5**

WORKING WITH HYPERLINKS, IMAGES, MULTIMEDIA AND FORMS 118

**CHAP 6**

INTRODUCTION TO HTML5 155

**CHAP 7**

INTRODUCTION TO NEW ELEMENTS IN HTML5 166

# BLOCK 2: ADVANCED HTML AND HTML5

## Block Introduction

In this block-2 of web technhologies, I have tried to emphasis on: advanced HTML and introduction of new version of HTML which is HTML5. Basically, I introduced advanced tags for creating hyperlinks, put image along with content, add multimedia file like audio and video and create form. Along with this a new version of HTML has introduce with its own structure and new elements in HTML5.

## Block Objective

The objective of the block is to explain advance tags of HTML and introduce HTML5. Students will able to learn about various advance tags which is useful for creating web page attractive and also new elements introduce in HTML5.

By learning this block of web technology student will learn about more tags of HTML, to create eye-caching web page. Reader of this block, will know web page development through various tags and add some new tags of HTML5.

Different tags are used to create web pages which include multiple tags. This block servers knowledge of different tags. I hope, this block will clear the idea of more tags of HTML and new version of HTML.

# Unit 5: Working with Hyperlinks, Images, Multimedia and Forms

## Unit Structure


5.1. Working with Hyperlinks

5.2. Working with Images

5.3. Working with Image Maps

5.4. Working with Multimedia

5.5. Working with Forms

5.6. Summary

# 5.1 Working with Hyperlinks

Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

There are two types of links
1. External link
2. Internal link

## 1. External link

### Linking Documents

The tags used to produce links are the <a> and </a>. The <a> tells where the link should start and the </a> indicates where the link ends.

Following is the simple syntax to use <a> tag.

```
<a href = "Document URL" ... attributes-list>Link Text</a>
```

### Link to an Email Address

Use mailto: inside the href attribute to create a link that opens the user's email program (to let them send a new email):

### Absolute URLs vs. Relative URLs

An absolute URL (a full web address) in the href attribute.

A local link (a link to a page within the same website) is specified with a relative URL (without the "https://www" part):

### Attributes

#### 1. target

By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.

The target attribute specifies where to open the linked document.

The target attribute can have one of the following values:

- _self - Default. Opens the document in the same window/tab as it was clicked
- _blank - Opens the document in a new window or tab
- _parent - Opens the document in the parent frame
- _top - Opens the document in the full body of the window

**2. colors for all links**

The general color of text links is specified in the <body> tag, like in the example below:

```
<body link="#C0C0C0" vlink="#808080" alink="#FF0000">
```

- link - standard link - to a page the visitor hasn't been to yet. (Standard color is blue).
- vlink - visited link - to a page the visitor has been to before. (Standard color is purple).
- alink - active link - the color of the link when the mouse is on it. (Standard color is red ).

## 3. Internal link

Linking to a Page Section

You can create a link to a particular section of a given webpage by using name attribute. This is a two-step process.

First create a link to the place where you want to reach with-in a webpage and name it using <a...> tag as follows −

```
<h1>HTML Text Links <a name = "top"></a></h1>
```

Second step is to create a hyperlink to link the document and place where you want to reach −

```
<a href = "/html/html_text_links.htm#top">Go to the Top</a>
```

Let’s perform steps to create Internal Link on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 5, in the document

Listing 5: Addingcreate Internal Link to an HTML Web page

```
<html>
<head>
<title>links</title>
</head>
<body>
<a name="top">
<h2>this is top of the page</h2>
<h1>linking to a section in a page</h1>

<h2>click here to go to the
<a href="#bottom">bottom </a>
of the page</h2>

<br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br>
<hr>
<a name="bottom">
<a href="#top">top</a>
<h3>this is a bottom of the page</h3>
</a>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as Internal_Link.html.

3. Open the HTML document in browser, as shown in figure 5

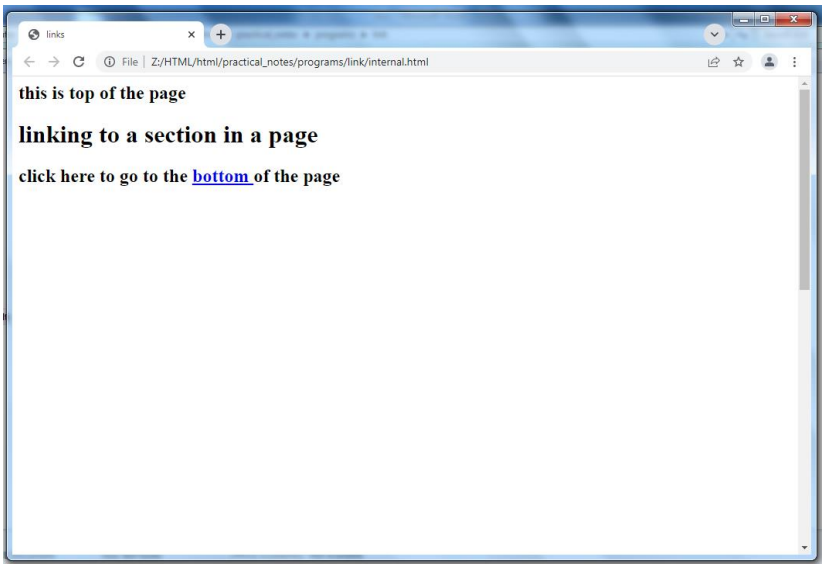



Figure 5: Displaying Internal Linkon web page

# 5.2 Working with Images

## Inserting Images into Web Pages

The <img> tag is used to insert images in the HTML documents. It is an empty element and contains attributes only.

The syntax of the <img> tag can be given with:

<img src="url" ….attribute list>

Let's perform steps to putimage on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 6, in the document

Listing 6: Addingput image to an HTML Web page

```
<html>
<head>
<title>Using Image in Webpage</title>
</head>
    <body>
```

```
<p>Simple Image Insert</p>
<img src = "computer.jpg" />
</body>
    </html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as image.html.
3. Open the HTML document in browser, as shown in figure 6

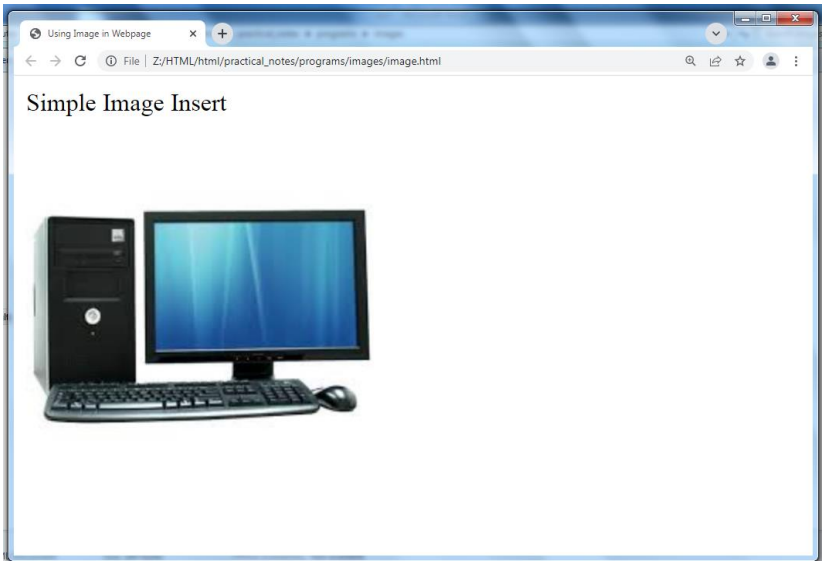



Figure 6: Displaying image on web page

## Attributes

## 1. Alternate text (alt)

Sometimes an inserted image does not appear in web browser because of incorrect path or file name. In this case you can specify the alternate text for an image.

The basic purpose of alternate text is to provide a short description about an image so that user can get idea of what the image is all about.

## 2. Width and Height

You can set image width and height based on your requirement using width and height attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

## 3. Image Border

By default, image will have a border around it, you can specify border thickness in terms of pixels using border attribute. A thickness of 0 means,no border around the picture.

## 4. Alignment

By default, image will align at the left side of the page, but you can use align attribute to set it in the center or right.

## Using Image as Links

It's simple to use an image as hyperlink. We just need to use an image inside hyperlink at the place of text

Let’s perform steps to displayimage as link on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 11, in the document

Listing 11: Addingimage as link to an HTML Web page

```
<html>

<head>

<title>Image Hyperlink Example</title>

</head>

<body>

<p>Click following link</p>

<a href = "order.html" target = "_self">

<img src ="computer.jpg" />

</a>

</body>
```

</html>

2. Save the document with appropriate name and .html extension. Here we have named HTML document as imageaslink.html.
3. Open the HTML document in browser, as shown in figure 11

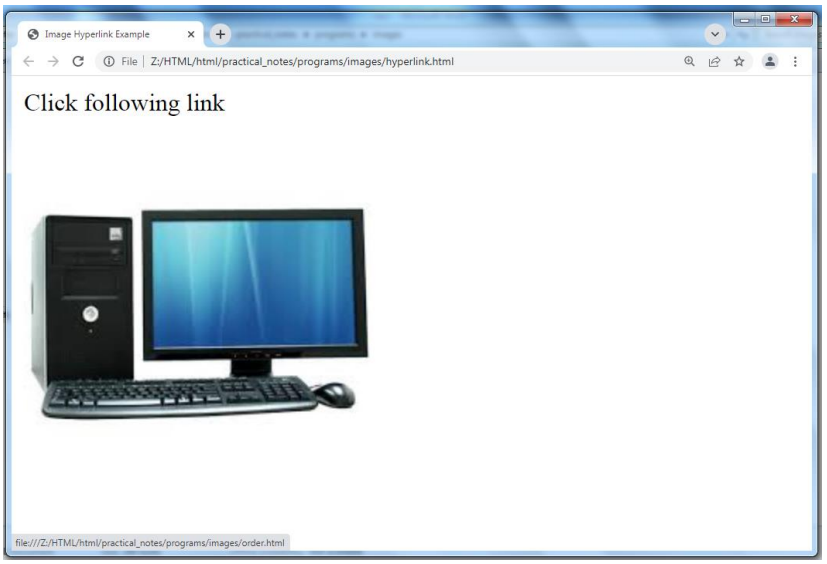



Figure 11: Displaying image as link on web page

## 5.3 Working with Image Maps

An image-map is an image with clickable areas.

Image map is a technique that divides the image into multiple sections and allows linking of each section to different web pages.

Linked regions of an image map are called hot regions. Each hot regions are associated with an HTML file.

### HTML Elements Used to Create Image Maps

There are three HTML elements used to create image maps:

- img: specifies the location of the image to be included in the map.
- map: is used to create the map of clickable areas.
- area: is used within the map element to define the clickable areas.

## Add the Map

Use the HTML <map> tag to create a map with a name. Inside <map>tag, you will specify where the clickable areas are with the HTML <area>tag.

We use the <area> tag in conjunction with the shape and coords attributes. These accept the following attributes:

| | |
|---|---|
| shape | Defines a shape for the clickable area. Possible values:<br>• default<br>• rect<br>• circle<br>• polygon |
| coords | Specifies the coordinates of the clickable area. Coordinates are specified as follows:<br>rect<br>left, top, right, bottom<br>circle<br>center-x, center-y, radius<br>polygon<br>x1, y1, x2, y2, ... |

Let’s perform steps to displayimage map on HTML Web page:

1. Open blank document in Notepad and add the code, given in listing 12, in the document

Listing 12: Addingimage map to an HTML Web page

```
<html>
<body>
<img src="computer.jpg" width="145" height="126" alt="Planets"
usemap="#cmap">
<map name="cmap">
<area shape="rect" coords="17,52,41,155" href="order.html">
<area shape="circle" coords="115,20,70"  href="unorder.html">
```

```
<area shape="polygon" coords="37,80,160,140,132,140" href="nestedlist.html">
</map>
</body>
</html>
```

2. Save the document with appropriate name and .html extension. Here we have named HTML document as imagemap.html.
3. Open the HTML document in browser to view output.

## 5.4 Working with Multimedia

Embedding Multimedia on a web page

### 1. <audio> tag

The <audio> tag is an inline element used to embed sound files into a web page. It is useful when you want to add any audio, such as a song or interview, to your web pages.

Attributes of audio tags are as follows

| Attribute | Value | Description |
|---|---|---|
| autoplay | autoplay | This Boolean attribute specifies that the audio will automatically start playing as soon as it can do so without stopping to finish loading the data. |
| controls | controls | If specified, the browsers will display controls to allow the user to control audio playback, such as play/pause, volume, etc. |
| loop | loop | This Boolean attribute specifies that the audio will automatically start over again, upon reaching the end. |
| muted | muted | This Boolean attribute specifies whether the audio will be initially silenced. The default value is false, meaning that the audio will be played. |
| preload | auto<br>metadata | Provides a hint to the browser about whether to download of the audio itself or its metadata. The |

| | | |
|---|---|---|
| | none | autoplay attribute can override this attribute, because if you want to automatically play a audio, the browser will obviously need to download it. |
| src | URL | Specifies the location of the audio file. |

## 2. <video> Tag

The <video> element is used to embed video content in an HTML document without requiring any additional plugin like Flash player. Support for the <video> element varies accross browsers.

The following table shows the attributes that are specific to the <video> tag.

| Attribute | Value | Description |
|---|---|---|
| autoplay | autoplay | This Boolean attribute specifies that the video will automatically start playing as soon as it can do so without stopping to finish loading the data. |
| controls | controls | If specified, the browsers will display controls to allow the user to control video playback, such as play/pause, volume, etc. |
| height | pixels | Sets the height of the video's display area. |
| loop | loop | This Boolean attribute specifies that the video will automatically start over again, upon reaching the end. |
| muted | muted | This Boolean attribute specifies whether the video will be initially silenced. The default value is false, meaning that the audio will be played when the video is played. |
| poster | URL | Specifies an image to be shown while the video is downloading, or until the user hits the play button.<br>i.e displays an image in case the video is unavailable or not yet loaded. |
| preload | auto | Provides a hint to the browser about whether |

| | | |
|---|---|---|
| | metadata<br>none | to download of the video itself or its metadata. |
| src | URL | Specifies the location of the video file to embed. Alternatively, you can use the preferred <source> tag as it allows for multiple options. |
| width | pixels | Sets the width of the video's display area. |

## 3. <source> Tag

The HTML <source> tag is used to specify multiple media resources on media elements (such as <audio> and <video>).

**Attributes**

- The following table shows the attributes that are specific to this tag/element.

| Attribute | Description |
|---|---|
| src | Specifies the location of the audio/video file. Its value must be the URL of an audio/video file. |
| type | Specifies the type of the embedded content. If specified, the value must be a MIME type. |
| media | Specifies the type of media resource, so the browser can determine whether it can play it or not. If not, it can choose not to download it. If specified, the value must be a valid media query. |

## 4. <embed> Tag

The easiest way to add video or sound to your web site is to include the special HTML tag called <embed>. This tag causes the browser itself to include controls for the multimedia automatically provided browser supports <embed> tag and given media type.

**Attributes**

The following table shows the attributes that are specific to this tag/element.

| Attribute | & Description |
|---|---|
| align | Determines how to align the object. It can be set to either center, left or right. |
| autostart | This boolean attribute indicates if the media should start automatically. You can set it either true or false. |
| loop | Specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false) |
| hidden | Specifies if the multimedia object should be shown on the page. A false value means no and true values means yes. |
| width | Width of the object in pixels |
| height | Height of the object in pixels |

**Browser that do not support embed**

You can also include a <noembed> tag for the browsers which don't recognize the <embed> tag.

## 5. <object> Tag

The HTML <object> tag is used for embedding an object within an HTML document. Use this tag to embed multimedia and other files in your web pages

Attributes

| Attribute | Description |
|---|---|
| data | Specifies the location of data to be used by the object. The value must be a valid URL. |
| type | Specifies the object type as specified in the data attribute. Must be a valid MIME |

| | |
|---|---|
| | type. |
| name | Assigns the name of the object. |
| width | Specifies the width, in pixels, to display the external content. |
| height | Specifies the height, in pixels, to display the external content.<br>Possible values: |

## 5.5 Working with Forms

The HTML <form> element provides a document section to take input from user.

It provides various interactive controls for submitting information to web server such as text field, text area, password field, etc.

Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

**Attributes**

| Attribute | Description |
|---|---|
| name | Assigns a name to the form. This is used when referencing the form with stylesheets or scripts. If there are multiple forms, the name of each form must be unique. |
| method | Specifies the HTTP method to use when the form is submitted.<br>Possible values:<br>• get (The form data is appended to the URL when submitted. This is the default value.)<br>• post (The form data is not appended to the URL.) |
| autocomplete | Specifies whether the form fields should be |

| | |
|---|---|
| | automatically completed based on the user's history (i.e. based on previous forms that the user has completed) |
| action | Specifies a URI/URL of the page that will process the form.i.e defines where to send data after submitting form. |

**HTML Form Syntax**

```
<form >
  //input controls e.g. textfield, textarea, radiobutton, button
</form>
```

**HTML Form Tags**

| Tag | Description |
|---|---|
| <form> | It defines an HTML form to enter inputs by the used side. |
| <input> | It defines an input control. |
| <textarea> | It defines a multi-line input control. |
| <label> | It defines a label for an input element. |
| <fieldset> | It groups the related element in a form. |
| <legend> | It defines a caption for a <fieldset> element. |
| <select> | It defines a drop-down list. |
| <optgroup> | It defines a group of related options in a drop-down list. |
| <option> | It defines an option in a drop-down list. |
| <button> | It defines a clickable button. |

## <input> element

The HTML <input> element is fundamental form element. It is used to create form fields, to take input from user. We can apply different input filed to gather different information form user. Following is the example to show the simple text input.

- **TextField Control**

The type="text" attribute of input tag creates textfield control also known as single line textfield control. The name attribute is optional.

**Example**

```
<form>

   First Name: <input type="text" name="firstname"/> <br/>

   Last Name:  <input type="text" name="lastname"/> <br/>

 </form>
```

- **Password Field Control**

The password is not visible to the user in password field control.

**Example**

```
<form>
    Password: <input type="password" id="password" name="password"/> <br/>
</form>
```

- **Email Field Control**

The email field in new in HTML 5. It validates the text for correct email address. You must use @ and . in this field.

**Example**

```
<form>
        <label for="email">Email: </label>
         <input type="email" id="email" name="email"/> <br/>
</form>
```

- **Radio Button Control**

The radio button is used to select one option from multiple options. It is used for selection of gender, quiz questions etc.

If you use one name for all the radio buttons, only one radio button can be selected at a time.

Using radio buttons for multiple options, you can only choose a single option at a time.

**Example**

```
<form>
        <label for="gender">Gender: </label>
          <input type="radio" id="gender" name="gender" value="male"/>Male

<input type="radio" id="gender" name="gender" value="female"/>Female <br/>
</form>
```

- **Checkbox Control**

The checkbox control is used to check multiple options from given checkboxes.

**Example**

```
<form>
        Hobby:
          <input type="checkbox" id="cricket" name="cricket" value="cricket"/>
         Cricket
        <input type="checkbox" id="football" name="football" value="football"/>
        Football:
      <input type="checkbox" id="hockey" name="hockey" value="hockey"/>
         Hockey:
</form>
```

- **<textarea> tag in form**

The <textarea> tag in HTML is used to insert multiple-line text in a form. The size of <textarea> can be specify either using "rows" or "cols" attribute

**Example**

```
<form>

        Enter your address:<br>

   <textarea rows="2" cols="20"></textarea>

</form>
```

- **Adding Button**

The <input type=”button”> tag places a button control on a HTML form.

**Example**

```
<form>

<input type="button" value="hello">

</form>
```

- **Submit button control**

HTML <input type="submit"> are used to add a submit button on web page. When user clicks on submit button, then form get submit to the server.

**Example**

```
<form>
        Enter name
      <input type="text" id="name" name="name"><br>
      Enter Password
      <input type="Password" id="pass" name="pass"><br>
      <input type="submit" value="submit">
</form>
```

The type = submit ,specifying that it is a submit button

- **Graphical Submit button**

You can add graphics to your regular submit button. This implies that you can define an image to act as a button on form.

**Example**

```
<form>

      Enter name

     <input type="text" id="name" name="name"><br>

      Enter Password

     <input type="Password" id="pass" name="pass"><br>
```

```
    <input type="image" src="submit.jpg" width="100" height="30">
</form>
```

- **Reset button control**

HTML <input type="reset"> are used to reset the values of form.

**Example**

```
<form>
    Enter name
    <input type="text" id="name" name="name"><br>
    Enter Password
    <input type="Password" id="pass" name="pass"><br>
    <input type="reset" value="reset">
</form>
```

The type = reset, specifying that it is a reset button

- **File input for form**

The <input type=”file”> tag is used to upload a file on a web page.

**Example**

```
<form>
<input type="file" name="filename" size="35">
</form>
```

- **Adding URL field for form**

➢ The URL field is used to enter only web addresses, in the correct format.If the URL is not entered in the correct format then URL field validates it.

**Attribute**

**Autofocus**: helps in keeping the focus of mouse pointer on the input field

**Example**

```
<form>
        Enter URL
       <input type="url"  name="url">
</form>
```

- **Adding tel field for form**

You can use <input type="tel"> to provide a telephone number in the field.

**Example**

```
<form>
      Telephone
   <input type="tel"  pattern="[0-9]{10}" title="phone number???"><br>
   <input type="submit" value="submit">
</form>
```

- **Adding datetime,date,month,week,time and datetime-local field to a form**

In HTML5 we can easily select date and time from a date picker control using various input type values of date

**Example**

```
<form>
```

date and time:

<input type="datetime" name="dt"/><br/>

date:

<input type="date" name="dte"/><br/>

time:

<input type="time" name="tm"/><br/>

week:

<input type="week" name="wk"/><br/>

month:

<input type="month" name="mn"/><br/>

local date & time:

<input type="datetime-local" name="dtlocal"/><br/>

</form>

- **Adding number field for form**

HTML5 contains a number type attribute of input field.This type is used to validate the textbox only if the value within the field is a numerical value.

**Example**

<form>

number:

<input type="number" name="num" min="0" max="10" value="3"><br/>

</form>

- **Adding range field for form**

In HTML 5 the range of values by using the type of the input field. you can use the <input type=”range”> tag, which is used to specify the range values.

**Example**

```
<form>

    number:

     <input type="range" name="rangeno" min="0" max="20" value="1"
    title="slider"><br/>

</form>
```

- **Adding color field for form**

The <input type=”color”> tag is used to select a color from a color picker control.

**Example**

```
<form>

    color:

<input type="color"><br/>

</form>
```

- **Adding a selection control**

HTML <select> tag is used to create a drop down list with multiple options. The <option> element is nested within <select> tag for defining options in a list.

The <optgroup> element can be used for grouping related options in a list.

**Syntax**

```
<select>
    <option></option>
 </select>
```

**Attributes**

| Attribute | Value | Description |
|---|---|---|
| autofocus | autofocus | This attribute let automatically focused the drop-down list on page |

| | | |
|---|---|---|
| | | load. |
| disabled | disabled | It is used to disable the control and user cannot interact with the drop-down list. |
| multiple | multiple | If it sets then a user can select multiple options from the list. |
| name | name | It determines the name for the drop-down list. |
| required | required | If it specified, user must select that field before submitting the form. |
| size | number | It specifies the visible number of options in the list. |

**Example**

```
<form>
    City:
<select name="city" size=3>
<option value="sydney">Sydney</option>
<option value="melbourne">Melbourne</option>
<option value="India">India</option>
    <option value="canada">canada</option>
</select>
</form>
```

## Grouping Form Controls

HTML <fieldset> tag is used to group the logically related fields/labels contained within an HTML form.

The use of this tag is optional while creating an HTML form but using <fieldset>, it is easy to understand the purpose of grouped elements of form.

The <legend> tag is used with the <fieldset> element as a first child to define the caption for the grouped related fields.

**Syntax**

**<fieldset>.....</fieldset>**

## 5.6 Summary

In this chapter you have learned about:

- Working with Hyperlinks
- Working with Images
- Working with Image Maps
- Working with Multimedia
- Working with Forms

# Unit 6: Introduction to HTML5

## Unit Structure


6.1. Introduction to HTML5

6.2. New Document Structure of HTML5

6.3. Browser Support for HTML5

6.4. Defining HTML Markup

6.5. Summary

# 6.1 Introduction to HTML5

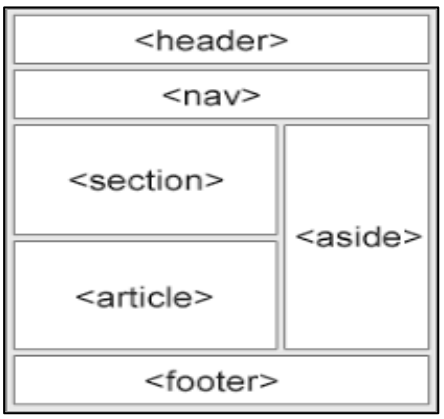
HTML5 is the fifth revision and newest version of the HTML standard. It offers new features that provide not only rich media support but also enhance support for creating web applications that can interact with users, their local data, and servers more easily and effectively than was previously possible.

It has improved the markup available for documents and has introduced application programming interfaces(API) and Document Object Model(DOM).

**Features:**

- It has introduced new multimedia features which supports audio and video controls by using <audio> and <video> tags.
- There are new graphics elements including vector graphics and tags.
- Enrich semantic content by including <header><footer>, <article>, <section> and <figure> are added.
- Drag and Drop- The user can grab an object and drag it further dropping it on a new location.
- Geo-location services- It helps to locate the geographical location of a client.
- Web storage facility which provides web application methods to store data on web browser.
- Uses SQL database to store data offline.
- Allows to draw various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.
- Easy DOCTYPE declaration i.e. <!doctype html>
- Easy character encoding i.e. <meta charset="UTF-8″>

# 6.2 New Document Structure of HTML5



**1. <header>**

The header element is used to contain the content that appears at the top of every page of your website: the logo, tagline, search prompt, and possibly a navigational menu.

**2. <nav>**

The <nav> element is used to declaring the navigational section in HTML documents. Websites typically have sections dedicated to navigational links, which enables user to navigate the site. These links can be placed inside a nav tag.

**3. <section>**

It defines different sections of webpage.

The section element is used to identify content that is a major sub-section of a larger whole.

**Example**

```
<section>
    <h1>Header</h1>
    <p>Some paragraph for example</p>
</section>
<section>
    <h1>Header 2</h1>
    <p>Another paragraph for example.</p>
</section>
```

**4. <article>**

The <article> element is used to define an independent, self-contained content (articles, blog posts, comments, etc.). The content of the element has its own meaning and it is easily differentiated from the rest of the webpage content.

**Example**

```
<article>
    <p>Text of the article</p>
</article>
```

**5. <aside>**

If your website contains information that isn't directly related to the main content of the page, it would be appropriate to wrap that information in aside tags.

Some possible uses for aside include a sidebar, a secondary list of links, or a block of advertising.

**6. <footer>**

The footer appears at the bottom of a section of a document.

The most common use of the footer element is to place it at the bottom of an HTML document to contain things like a copyright notice, links to related content, address information about the owner of the website, and links to administrative things like privacy policies and website's terms of service.

## 6.3 Browser Support for HTML5

HTML5 is supported in all modern browsers.

The major browsers are google chrome, opera, Mozilla firefox, apple's safari and intrenet explorer 8.

## 6.4 Defining HTML Markup

**The <!doctype> html element**

A basic HTML page always starts with the Document Type Declaration or doctype. That is a way to tell the browsers what type of document it is.

The doctype is always the first thing at the top of any HTML file. Then sections and subsections come, each possibly has its heading and subheading. These heading and sectioning elements helps the reader to perceive the content meaning.

To indicate that your HTML content uses HTML5, simply use:

<!DOCTYPE html>

Let's perform steps to write program in HTML5:

1. Open blank document in Notepad and add the code, given in listing 1, in the document

Listing 1: AddingElements of HTML5 Web page

```
<!DOCTYPE HTML>

<html>

<head>

<title>Title of the document</title>

<article>

<header>

<h2>Learn HTML</h2>

</header>

<p>Our free HTML tutorial for beginners will teach you HTML and how to create your website from the scratch. HTML isn't difficult, so hoping you will enjoy learning.</p>

</article>

<article>

<header>

<h2>Start Quiz "HTML Basic"</h2>

<small>You can test your HTML skills with W3docs' Quiz.</small>

</header>

<p>You will get 5% for each correct answer for single choice questions. In multiple choice question it might be up to 5%. At the end of the Quiz, your total score will be displayed. Maximum score is 100%.</p>

</article>

</section>

<aside>

<h2>Our Books</h2>

<p>HTML</p>

<p>CSS</p>

<p>JavaScript</p>

<p>PHP</p>

</aside>
```

```
<footer>
<small>Company © W3docs. All rights reserved.</small>
</footer>
</body>
</html>
```

2. Save the document with appropriate name and HTML5_Elements.html extension. Here we have named HTML document as audio.html.
3. Open the HTML document in browser,as shown in figure 1

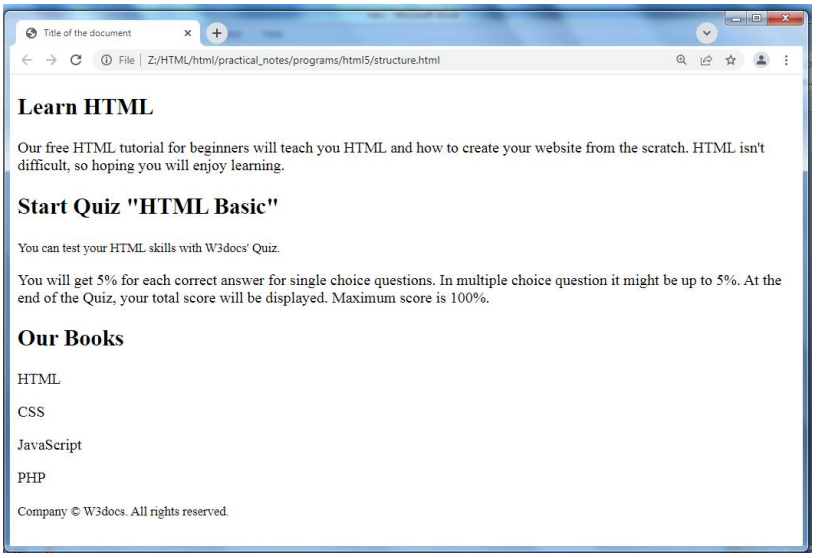



Figure 1: Displaying output on web page

### The <html> element

The <html> element follows the doctype information, which is used to inform the browser that this is an HTML document.

### Attributes

| Attribute | Description |
|---|---|
| class | The class attribute is used to associate an element with a style sheet, and specifies the class of element. Used in CSS.<br>The value of the attribute may also be a space-separated list of class names. For example −<br>class = "className1 className2 className3" |

<table>
<tr><td>dir</td><td>The dir attribute allows you to indicate to the browser about the direction in which the text should flow. The dir attribute can take one of two values<table><tr><td>ltr</td><td>Left to right (the default value)</td></tr><tr><td>rtl</td><td>Right to left</td></tr></table></td></tr>
<tr><td>lang</td><td>The lang attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML.</td></tr>
<tr><td>id</td><td>The id attribute of an HTML tag can be used to uniquely identify any element within an HTML page.<br>If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.<br>Example<br> This para explains what is HTML <br><br> This para explains what is Cascading Style Sheet </td></tr>
<tr><td>manifest</td><td>The manifest attribute gives the address of the document's application cache manifest, if there is one.<br><br>It only has an effect during the early stages of document load. is part of the legacy "offline Web applications" feature, which is in the process of being removed from the Web platform.</td></tr>
<tr><td>Version</td><td>It refer to the version of the HTML page</td></tr>
</table>

**Elements to be added to the <html> head**

<header>

<nav>

<section>

<article>

<aside>

<footer>

**The <head> element**

The <head> element contains metadata (document title, character set, styles, links, scripts), specific information about the web page that is not displayed to the user.

**Attributes**

<table>
<tr><td>Attribute</td><td>Description</td></tr>
<tr><td>class</td><td>The class attribute is used to associate an element with a style sheet, and specifies the class of element. Used in CSS.<br>The value of the attribute may also be a space-separated list of class names. For example −<br>class = "className1 className2 className3"</td></tr>
<tr><td>dir</td><td>The dir attribute allows you to indicate to the browser about the direction in which the text should flow. The dir attribute can take one of two values<br><table><tr><td>ltr</td><td>Left to right (the default value)</td></tr><tr><td>rtl</td><td>Right to left</td></tr></table></td></tr>
<tr><td>lang</td><td>The lang attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML.</td></tr>
<tr><td>id</td><td>The id attribute of an HTML tag can be used to uniquely identify any element within an HTML page.<br>If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.<br>Example<br> This para explains what is HTML <br><br> This para explains what is Cascading Style Sheet </td></tr>
<tr><td>style</td><td>The style attribute allows you to specify Cascading Style Sheet (CSS) rules within the element.</td></tr>
<tr><td>title</td><td>Holds additional information for the element.</td></tr>
</table>

**Elements to be added to the <head> head**

<title>

<style>

<script>

<link>

**The <title> element**

The HTML <title> tag is used for declaring the title, or name, of the HTML document.

The title is usually displayed in the browser's title bar (at the top).

**Attributes**

| Attribute | Description |
|---|---|
| class | The class attribute is used to associate an element with a style sheet, and specifies the class of element. Used in CSS.<br>The value of the attribute may also be a space-separated list of class names. For example −<br>class = "className1 className2 className3" |
| lang | The lang attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML. |
| id | The id attribute of an HTML tag can be used to uniquely identify any element within an HTML page.<br>If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.<br>Example<br> This para explains what is HTML <br> This para explains what is Cascading Style Sheet |
| style | The style attribute allows you to specify Cascading Style Sheet (CSS) rules within the element. |

**The <body> element**

The HTML <body> tag is used for indicating the main content section of the HTML document. The body tag is placed between the </head> and the </html> tags.

**Attributes**

| Attribute | Description |
|---|---|
| class | The class attribute is used to associate an element with a style sheet, and specifies the class of element. Used in CSS.<br>The value of the attribute may also be a space-separated list of class names. For example −<br>class = "className1 className2 className3" |
| lang | The lang attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML. |

| | |
|---|---|
| id | The id attribute of an HTML tag can be used to uniquely identify any element within an HTML page.<br>If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.<br>Example<br> This para explains what is HTML <br><br> This para explains what is Cascading Style Sheet |
| style | The style attribute allows you to specify Cascading Style Sheet (CSS) rules within the element. |
| title | Holds additional information for the element. |

## 6.5 Summary

In this chapter you have learned about:

- Introduction to HTML5
- New Document Structure of HTML5
- Browser Support for HTML5
- Defining HTML Markup

# Unit 7: Introduction to New Elements in HTML5

## Unit Structure


7.1. Markup Elements in HTML5

7.2. The canvas Element

7.3. New Elements in Forms

7.4. Summary

# 7.1 Markup Elements HTML5

Here is the table showing all Markup elements with description.

| Element | Description |
|---|---|
| <article> | The <article> tag is used to represent an article. Helps in adding content on a webpage which can be Block, inline, and text |
| <aside> | Helps in adding extra information such as note, tip etc. |
| <details> | The details element is used to represent an area where users can go to obtain additional information. |
| <summary> | The <summary> tag in HTML is used to define a summary or title for the <details> element. |
| <figure> | The <figure> tag in HTML is used to add content like illustrations, diagrams, photos in a document. |
| <figcaption> | The <figurecaption> tag in HTML is used to set a caption to the<figure >element |
| <header> | The header element typically contains the headings for a section. A header should contain title and heading information about the related content. |
| <footer> | The <footer> tag in HTML is used to define a footer of HTML document. This section contains the footer information (author information, copyright information etc. |
| <hgroup> | The <hgroup> element is used to group heading elements. |
| <mark> | The <mark> element defines a marked section of text. You can use this tag if you want to highlight a section of your text for reference purposes. |
| <meter> | The <meter> element represents a scalar measurement within a known range, or a fractional value. |
| <nav> | The <nav> element defines a section of navigation links (i.e. links to other pages or to parts within the page itself) in a |

| | |
|---|---|
| | document. |
| <progress> | This element normally used to indicate how much of a task has been completed, such as loading something on a page or registration process. It is typically displayed as a progress bar and often marked as a percentage from 0 to 100%. |
| <ruby> | The <ruby> element represents a ruby annotation. Ruby annotations are used for showing pronounciation of East Asian characters, like Chinese and Japanese Chinese. |
| <rt> | element is used within a <ruby> to define the pronunciation of character presented in a ruby annotations. Used to create annotations or pronunciation guides for words and phrases. |
| <rp> | element is used to provide fall-back parenthesis for browsers that that don't support ruby annotations. |
| <section> | The <section> element defines a section of a document, such as header, footer etc. |
| <time> | The <time> element represents a time and/or date. |

## 7.2 The Canvas Element

The HTML5 canvas element can be used to draw graphics on the webpage via scripting usually JavaScript.

The <canvas> element is only a container for graphics, you must need a scripting language to draw the graphics. The <canvas> element allows for dynamic and scriptable rendering of 2D shapes and bitmap images.

### How to create a HTML canvas?

A canvas is a rectangle like area on an HTML page. It is specified with canvas element. By default, the <canvas> element has no border and no content, it is like a container.

```
<canvas id = "mycanvas" width ="200" height ="100">
 </canvas>
```

By default the canvas element has 300px of width and 150px of height without any border and content. However the custom width and height can be defined.

### HTML Canvas Tag with JavaScript

The canvas is a two-dimensional rectangular area. The coordinates of the top-left corner of the canvas are (0, 0) which is known as origin.

The canvas element has a DOM method called getContext, used to obtain the rendering context and its drawing functions. This function takes one parameter, the type of context2d.

**Example**

1. The parameters (0,0,200,100) is used for fillRect() method. This parameter will fill the rectangle start with the upper-left corner (0,0) and draw a 200 * 100 rectangle.

2. Drawing Line on Canvas

If you want to draw a straight line on the canvas, you can use the following two methods.

moveTo(x,y): It is used to define the starting point of the line.

lineTo(x,y): It is used to define the ending point of the line.

## 7.3 New Elements in Forms

New elements of form in HTML5 are as follows

| Tag | Description |
|---|---|
| <datalist> | It specifies a list of pre-defined options for input control. |
| <keygen> | It defines a key-pair generator field for forms. |
| <output> | It defines the result of a calculation. |

## 1. <datalist> element

The HTML <datalist> tag is is used to provide an auto complete feature on form element. It provides a list of predefined options to the users to select data.

The datalist tag is introduced in HTML5.The <datalist> tag should be used with an <input>element that contains a "list" attribute. The value of "list" attribute is linked with the datalist id.

## 2. <output> element

HTML <output> tag is used to display the result of some calculation (performed by JavaScript) or the outcome of a user action (such as Input data into a form element).

The <output> tag is a newly added tag and was introduced in HTML5.

**Syntax**

<output>.....</output>

## 3. <keygen> element

The **KEYGEN Element** is commonly used for generating the key pair in the form. Whenever user hit the submit button, the **KEYGEN Element** creates two key pair, first one is Public Key and another one is Private Key.

The private key is encrypted and stored in the local key database and the public key is sent with the form data to the server. The **KEYGEN Element** is most useful when the user wants to generate the unique key for a particular form.

Here is the Syntax for KEYGEN Element

```
<keygen name="key">
```

# 7.4 Summary

In this chapter you have learned about:

- Markup Elements in HTML5
- The canvas Element
- New Elements in Forms

# Introduction to Web Designing

## BLOCK3: DYNAMIC HTML CONCEPTS (CSS,JAVA SCRIPT)

**CHAP 8**

CASCADING STYLE SHEET 187

**CHAP 9**

CASCADING STYLE SHEET ATTRIBUTES AND PROPERTIES 212

**CHAP 10**

INTRODUCTION TO JAVA SCRIPT 278

**CHAP 11**

ARRAYS, FUNCTIONS, EVENTS AND DIALOG BOXES IN JAVA SCRIPT 315

# BLOCK 3: DYNAMIC HTML CONCEPTS (CSS, JAVA SCRIPT)

## Block Introduction

In this block-3 of web technhologies, I have tried to emphasis on: Dynamic HTML for creating interactive web page. Basically, I introduced cascading style sheet with CSS attributes and properties which is used to format the layout of web page and Java script-a scripting language supports concepts of programming language. Along with this I have also include Events and various dialog boxes in Java script which helps in making web page dynamic and allow user's interaction .

## Block Objective

The objective of the block is to learn cascading style sheet with CSS attributes and properties and Java script-a scripting language. Students will able to learn about howto create uniformity throughout a web site by using numerous CSS attributes to create dynamic effects andinterpreted language Java script which is executed without complication.

By learning this block of web technology student will learn about adding dynamic concepts to web page and make web page more interactive and attractive. Reader of this block, will know web page development through various attributes of CSS and various syntax of Java script.

According to requirement we can add CSS as well as Java Script very easily to existing code of HTML . This block servers knowledge of CSS and Java script. I hope, this block will clear the idea of both these concepts of dynamic HTML.

# Unit 8: Cascading Style Sheet

## Unit Structure


8.1. Introduction to Cascading Style Sheet

8.2. CSS Syntax

8.3. CSS Selectors

8.4. Selectors Grouping

8.5. CSS Comments

8.6. Types of Style Sheets

8.7. Summary

# 8.1 Introduction to Cascading Style Sheet

## Concept of CSS

Cascading style sheets are used to format the layout of Web pages. CSS enforce standards and uniformity throughout a web site and provide numerous attributes to create dynamic effects.

They can be used to define (fonts, colors, background, borders, text formatting, link effects & so on…).These aspects of Web pages that previously could only be defined in a page's HTML.

Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Plus, CSS makes it easy to change styles across several pages at once.

**For example,** a Web developer may want to increase the default text size from 10pt to 12pt for fifty pages of a Web site. If the pages all reference the same style sheet, the text size only needs to be changed on the style sheet and all the pages will show the larger text.

## Advantages CSS

### Making changes to the layout

CSS makes it very easy to change the style of a document. **Let's say** we wanted to move the picture in the title of this page to the right by 10 pixels.

We have to do is open our CSS file which stores the layout of the site, and change the number relating to the position of the image. That will change his position throughout the whole site.

The look and layout of a site can be changed beyond recognition just by altering the CSS file. This makes CSS indispensable for large web sites.

### File Size

Probably the mostly useful feature of CSS is that all of the style and layout is removed from the html, so the html page size is very much smaller.

The CSS file is downloaded just once by the visitor's browser and re-used for different pages on a web site. This reduces the bandwidth requirements for your server and also ensures a faster download for your visitors.

**Search Engines**

A search engine normally considers the text at the start of your html page as more important than the text towards the end of the code.

CSS-based websites use simpler and better structured markup (HTML) and are therefore more accessible to search engines.

**Accessibility**

Separating style from content makes life very easy for visitors who prefer to view only the content of a web page, or to modify the content. These could be blind or partially sighted people who might use a screen reader to interpret a page.

**Consistency**

Layout and position of navigation can be completely consistent across a site. This was previously possible only using frames.

**Improved web pages download times**

By using CSS for your website's layout, you will see significant improvements in your web page download times. With CSS you will achieve faster download speeds.

**Higher search engine rankings**

Search engines like CSS-based websites and are likely to place them higher in search engine rankings because:

- There is a greater density of content compared to coding.
- CSS-based websites use simpler and better structured markup (HTML) and are therefore more accessible to search engines.
- Important content can be placed at the top of the HTML document (content before navigation).

**Easier website management**

By using CSS for style and layout, you only have to adjust one style sheet to make adjustments right across your website. Table-based layouts require each web page to be adjusted. So CSS is less time-consuming. Imagine if you had a 100 page website. It would take a long time to adjust if it used tables, but would take no time at all using CSS.

**Web page print friendly**

When a user wishes to print a web page an alternative CSS document can be called up. This document can specify that only the content and logo are to appear on the print out, with the navigation and formatting made to disappear.

**Disadvantages CSS**

**Browser compatibility**

Different browsers will render CSS layout differently as a result of browser bug or lack of support for CSS features.

CSS renders different dimensions with each browser. Programmers are required to consider and test all code across multiple browsers for compatibility before taking any website

**Lack of Variables**
CSS contain no variables. This makes it necessary to do a "replace-all" when one desires to change a fundamental constant, such as color scheme or various heights and widths.

## 8.2 CSS Syntax

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts

- **Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

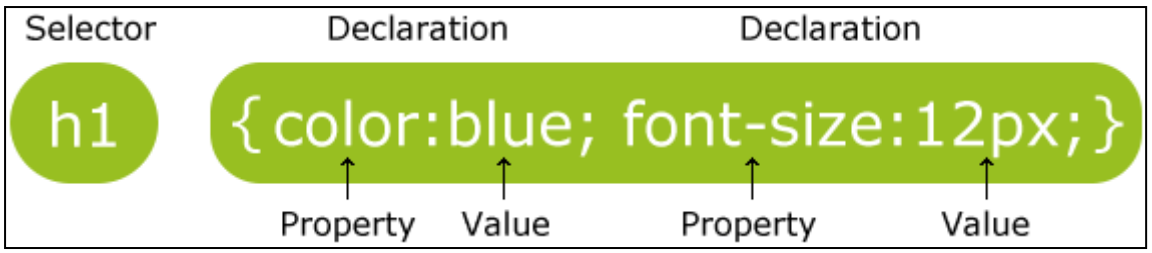
- **Property** - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

- **Value -** Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule **Syntax** as follows −

**Selector { property1: value1; property2: value2; property3: value3……. }**

A CSS rule-set consists of a selector and a declaration block:




**The selector** points to the HTML element you want to style.

**The declaration block** contains one or more declarations separated by semicolons.

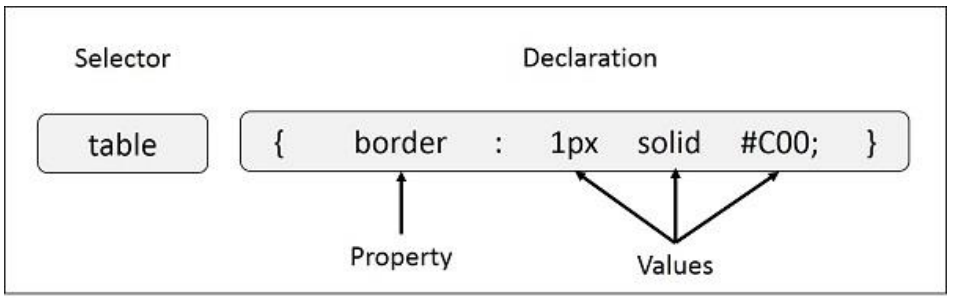
**Each declaration includes a CSS property name and a value, separated by a colon.**

**A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.**

In the following example all <p> elements will be center-aligned, with a red text color:

**Example**

```
p {
    color: red;
    text-align: center;
}
```




You can define a table border as follows −

table{ border :1px solid #C00; }

Here table is a selector and border is a property and given value 1px solid #C00 is the value of that property.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

# 8.3 CSS Selector

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

### 1. The element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

**Example**

```
p {
        text-align: center;
        color: red;
}
```

### 2. The id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element!

**To select an element with a specific id, write a hash (#) character, followed by the id of the element.**

The style rule below will be applied to the HTML element with id="para1":

**Example**

```
#para1 {
        text-align: center;
        color: red;
      }
```

### 3. The Universal Selectors

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type −

```
* {
  color: #000000;
}
```

This rule renders the content of every element in our document in black.

4. **The Descendant Selectors**

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to  element only when it lies inside <ul> tag.

**Example**

```
ul em {
        color: #000000;
      }
```

5. **The Child Selectors**

This selector matches all elements that are the immediate children of a specified element

. The combination in a child selector is a greater-than sign (>).

**Syntax**

selector1 > selector2 { *style properties* }

**Example**

```
ul>li {
   color: red;
}
```

6. **The class Selector**

The class selector selects elements with a specific class attribute.

**To select elements with a specific class, write a period (.) character, followed by the name of the class.**

In the example below, all HTML elements with class="center" will be red and center-aligned:

**Example**

```
.center {
        text-align: center;
        color: red;
    }
```

You can also specify that only specific HTML elements should be affected by a class.

In the example below, only <p> elements with class="center" will be center-aligned:

**Example**

```
p.center {
        text-align: center;
       color: red;
     }
```

HTML elements can also refer to more than one class.

In the example below, the <p> element wil be styled according to class="center" and to class="large":

**Example**

```
<p class="center large">This paragraph refers to two classes.</p>
```

## 8.4 Selector Grouping

A Declaration block may be assigned to multiple Selectors by separating the Selectors with commas.

It will be better to group the selectors, to minimize the code.

**Example**

```
h1 { color: green }      Becomes =>      h1, h2, h3 { color: green; }
h2 { color: green }
h3 { color: green }
```

**HTML code**

```
<html>

<head>

<style>
```

```
h1, h2, p {
    text-align: center;
    color: red;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Output Shows in figure 10

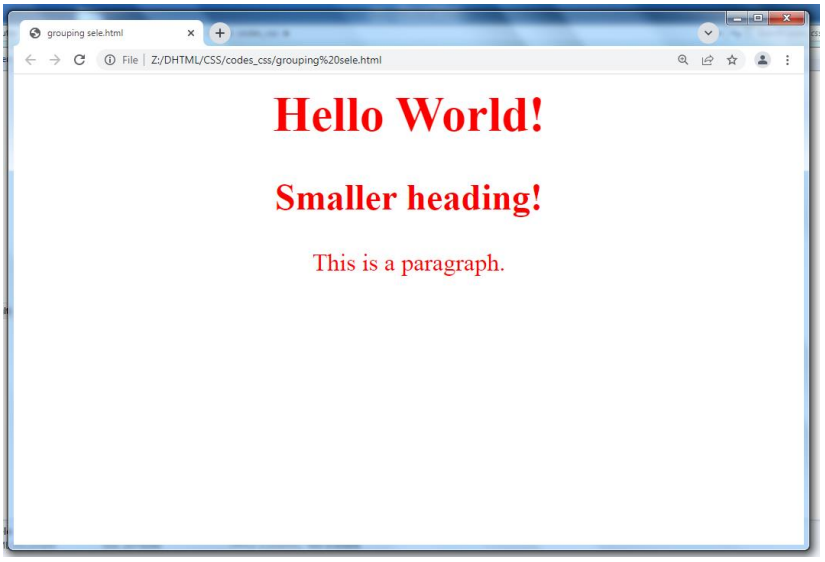



Figure 10: Displaying output of selector grouping on web page

## 8.5 CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

**Example**

```
p {
color: red;
  /* This is a single-line comment */
 text-align: center;
}
```

```
/* This is
a multi-line
comment */
```

## 8.6 Types of Style Sheet

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

---

**External Style Sheet**

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

```
body {
   background-color: lightblue;
```

```
}

h1 {
   color: navy;
   margin-left: 20px;

}
```

Following code shows how to link external style sheet

```
<html>

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>
```

**Internal Style Sheet**

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

Following code shows Internal Stylesheet

**Example:**

```
<html>

<head>

<style>

body {
```

```
    background-color: linen;
}
h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

**Inline Styles**

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

```
<html>
<body>
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

**Multiple Style Sheets**

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

**Example:**

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
    color: orange;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>The style of this document is a combination of an external stylesheet, and internal style</p>
</body>
</html>
```

So, an inline style (inside a specific HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value.

**Example:**

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
```

```
<style>

body {background-color: linen;}

</style>

</head>

<body style="background-color: lightcyan">

<h1>Multiple Styles Will Cascade into One</h1>

<p>In this example, the background color is set inline, in an internal stylesheet, and in an external stylesheet.</p>

<p>Try experimenting by removing styles to see how the cascading stylesheets work. (try removing the inline first, then the internal, then the external)</p>

</body>

</html>
```

## 8.7 Summary

In this chapter you have learned about:

- Introduction to Cascading Style Sheet
- CSS Syntax
- CSS Selectors
- Selectors Grouping
- CSS Comments
- Types of Style Sheets

# Unit 9: Cascading Style Sheet Attributes and Properties

## Unit Structure


9.1. CSS Color Attribute

9.2. CSS Background Attributes

9.3. CSS Font Attributes

9.4. CSS Text Attributes

9.5. CSS Border Attributes

9.6. CSS Margin Attributes

9.7. CSS Height and Width Attributes

9.8. CSS Padding Attributes

9.9. CSS List Attributes

9.10. CSS Table Attributes

9.11. CSS Position Attributes

9.12. Summary

# 9.1 CSS Color Attribute

Set the text-color for different elements

**Example**

```
body {
    color: red;
}

h1 {
    color: #00ff00;
}

p {
    color: rgb(0,0,255);
}
```

# 9.2 CSS Background Attributes

Set different background properties in one declaration

| Property | Description |
|---|---|
| background-color | Specifies the background color to be used |
| background-image | Specifies ONE or MORE background images to be used |
| background-position | Specifies the position of the background images. You can set this property to **top left/center/right ,center left/center/right, bottom left/center/right right top/center/bottom, left top/center/bottom, x% y%(**The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%. . Default value is: 0% 0%)**,**<br>**x-position y-position(**The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) |
| background-size | Specifies the size of the background images. |
| background-repeat | Specifies how to repeat the background images. You can set this property to **repeat, repeat-x, repeat-y or no- repeat.** |

| | |
|---|---|
| background-attachment | Specifies whether the background images are fixed or scrolls with the rest of the page. You can set this property to scroll or fixed. |

## 9.3 CSS Font Attributes

Set font related styles for the text present on an HTML web page.

| Property | Description |
|---|---|
| font-family | Specifies a prioritized list of font family name /or generic names, the browser should try to match. (serif, sans-serif, Times new roman etc.) |
| font-size | You can set the size of the text used in an element by using the font-size property. You can set this property to xx-large, x-large, larger, large, medium, small, smaller, x-small, xx-small,% (percent), a number(of pixels) |
| font-style | You can set the style of text in element. You can set this property to normal, italic or oblique |
| font-weight | You can control the weight of text in an element. You can set this property to normal, bold, bolder, lighter, or numeric value(100, 200, 300, 400, 500, 600, 700, 800, 900) |
| font-variant | Specifies whether or not a font is small-caps font; You can set this property to normal or small-caps |

## 9.4 CSS Text Attributes

The text properties allow you to format text present in HTML web page by providing various text formatting options.

| Property | Description |
|---|---|
| Color | Specifies the foreground color of the text. |
| text-align | The text-align property is used to set the horizontal alignment of a text. A text can be **left or right aligned, centered, or justified.** |
| text-decoration | The text-decoration property is used to set or remove decorations from text. You can set this property to **underline, overline, line-through or blink** |
| text-transform | The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into **uppercase or lowercase letters, or capitalize the first letter of each word.** |

| | |
|---|---|
| text-indent | The text-indent property is used to specify the indentation of the first line of a text.It can be length(pixcel(px), centimeter(cm), inch(in), point(pt)) or percentage |
| letter-spacing | The letter-spacing property is used to specify the space between the characters in a text.You can set this to normal or length(pixcel(px), centimeter(cm), inch(in), point(pt)). |
| line-height | The line-height property is used to specify the space between lines. |
| direction | The direction property is used to change the text direction of an element: |
| word-spacing | The word-spacing property is used to specify the space between the words in a text. You can set this to normal or length(pixcel(px), centimeter(cm), inch(in), point(pt)). |

## 9.5 CSS Border Attributes

The CSS border properties allow you to specify the style, width, and color of an element's border.

| Property | Description |
|---|---|
| border-style | The border-style property specifies what kind of border to display. The following values are allowed:<br>• dotted - Defines a dotted border<br>• dashed - Defines a dashed border<br>• solid - Defines a solid border<br>• double - Defines a double border<br>• groove - Defines a 3D grooved border<br>• ridge - Defines a 3D ridged border.<br>• inset - Defines a 3D inset border.<br>• outset - Defines a 3D outset border.<br>• none - Defines no border |
| border-top-style, border-right- style, border-bottom-style and border-left- style | Specify the styles of the top, right, bottom and left borders. |
| border-width | The border-width property specifies the width of the four borders. The width can be set as a specific size (in px, pt, cm etc) or by using one of the three pre-defined values: thin, medium, or thick. |
| border-top-width, border-right- width, border-bottom-width and border-left- width | Specify the width of the top, right, bottom and left borders. |
| border-color | The border-color property is used to set the color of the four borders. The border-color property can have from |

| | |
|---|---|
| | one to four values (for the top border, right border, bottom border, and the left border<br><br>The color can be set by:<br><br>• name - specify a color name, like "red"<br>• RGB - specify a RGB value, like "rgb(255,0,0)"<br>• Hex - specify a hex value, like "#ff0000" |
| border-top-color, border-right-color, border-bottom-color and border-left-color | Specify the colors of the top, right, bottom and left borders. |

If the border-style property has four values:

- border-style: dotted solid double dashed;
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

If the border-style property has three values:

- border-style: dotted solid double;
  - top border is dotted
  - right and left borders are solid
  - bottom border is double

If the border-style property has two values:

- border-style: dotted solid;
  - top and bottom borders are dotted
  - right and left borders are solid

If the border-style property has one value:

- border-style: dotted;
  - all four borders are dotted

**Border - Shorthand Property**

To shorten the code, it is also possible to specify all the individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

- border-width

- border-style (required)
- border-color

## 9.6 CSS Margin Attributes

The CSS margin properties are used to generate space around elements.

The CSS margin properties set the size of the white space OUTSIDE the border.

(The margins are completely transparent - and cannot have a background color.It is also possible to use negative values for margins)

| Property | Description |
|---|---|
| Margin | It set margin properties for all four borders.You can set it to auto, length(pixcel(px), centimeter(cm), inch(in), point(pt)) or percentage |
| Margin-top, Margin-right, Margin-bottom, Margin-left | Specify the margin of the top, right, bottom and left borders. |

**Margin - Shorthand Property**

The margin property is a shorthand property for the following individual margin properties:

- margin-top
- margin-right
- margin-bottom
- margin-left

**Use of the auto Value**

You can set the margin property to auto to horizontally center the element within its container.

The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

## 9.7 CSS Height and Width Attributes

The CSS dimension properties allow you to control the height and width of an element.

| Property | Description |
|---|---|
| height | Sets the height of an element. You can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in length values, like px, cm, etc., or in percent (%) of the containing block. |
| Width | Sets the width of an element. You can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in length values, like px, cm, etc., or in percent (%) of the containing block. |
| max-height | Sets the maximum height of an element |
| min-height | Sets the minimum height of an element |
| max-width | Sets the maximum width of an element |
| min-width | Sets the minimum width of an element |

## 9.8 CSS Padding Attributes

The CSS padding properties are used to generate space around content.

The CSS padding properties define the white space between the element content and the element border.

All the padding properties can have the following values:

- length - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element

| Property | Description |
|---|---|
| padding | A shorthand property for setting all the padding properties in one declaration |
| padding-bottom | Sets the bottom padding of an element |
| padding-left | Sets the left padding of an element |
| padding-right | Sets the right padding of an element |
| padding-top | Sets the top padding of an element |

## 9.9 CSS List Attributes

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

| Property | Description |
|---|---|
| list-style-type | Specifies the type of list-item marker. values are disc, square, circle, decimal, decimal-leading-zero(01,02), lower-roman, upper-roman, lower-alpha, upper-alpha, lower-greek(α,β), lower-latin(a,b,c), upper-latin(A,B) |
| list-style-image | Specifies an image as the list-item marker.You can set this property to none or url. |
| list-style-position | Specifies if the list-item markers should appear inside or outside the content flow. |
| list-style | Sets all the properties for a list in one declaration |

## 9.10 CSS Table Attributes

| Property | Description |
|---|---|
| border | Sets all the **border properties** in one declaration |
| border-collapse | Specifies whether or not table borders should be collapsed into a single border: |
| border-spacing | Specifies the distance between the borders of adjacent cells |
| caption-side | Specifies the placement of a table caption. values are **top,** |

| | |
|---|---|
| | **bottom** |
| empty-cells | Specifies whether or not to display borders and background on empty cells in a table. values are **show, hide** |
| table-layout | Sets the layout algorithm to be used for a table. values are **auto, fixed** |
| Width | Sets the width of table. |

## 9.11 CSS Position Attributes

The position property specifies the type of positioning method used for an element.

| Property | Description |
|---|---|
| top, right, bottom and left | Specifies top, right, bottom, left position of content |
| Position | Specifies position of content. You can set this property to static, relative, absolute, or fixed |
| z-index | Specifies the stack order of an element (which element should be placed in front of, or behind, the others). An element can have a positive or negative stack order |
| unicode-bidi | The unicode-bidi property is used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document.<br><br>Values are normal, embed, bidi-override |

There are four different position values:

- **Static**

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

- **Relative**

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position

- **Fixed**

An element is positioned relative to the view port, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

**Absolute**

This type of positioning allows you to **place your element precisely where you want it**.

The positioning is done **relative to the first relatively (or absolutely) positioned parent element**. In the case when there is no positioned parent element, it will be positioned related **directly to the HTML element (the page itself)**.

**What Is Z-Index?**

- The Z-Index property determines the stack level of an HTML element.CSS layer refer to applying z-index property to element that overlap to another element.

- An element with a higher z-index will be displayed in front of an element with a lower z-index. One thing to note is that z-index **only works with positioned elements**.

- CSS z-index possible value 0, positive (1 to 9999) and negative (-1 to -9999) value to set an element.

## 9.12 Summary

In this chapter you have learned about:

- CSS Color Attribute
- CSS Background Attributes
- CSS Font Attributes
- CSS Text Attributes
- CSS Border Attributes
- CSS Margin Attributes
- CSS Height and Width Attributes
- CSS Padding Attributes
- CSS List Attributes
- CSS Table Attributes
- CSS Position Attributes

# Unit 10: Introduction to Java Script

## Unit Structure

10.1. Introduction to Java Script

10.2. Select Developing Environment for Java Script

10.3. HTML and Java Script

10.4. Elements of Java Script

10.5. Java Script Variables

10.6. Types of Data in Java Script

10.7. Java Script Operators

10.8. Java Script Flow Control Statements

10.9. Summary

# 10.1Introduction to Java Script

JavaScript is a scripting language. It was designed by Netscape to create interactive and dynamic web pages.

It is also known as lightweight programming language.

It is usually embedded directly into HTML pages.

It is an interpreted language which is executed without complication.

**Advantages of JavaScript**

1) **It is an interpreted language.**
    - It requires no compilation steps. All steps are interpreted by browser like html.
    - It is interpreted by the scripting engine which is part of browser.
2) **Embedded within HTML**
    - It does not require any special or separate editor for programs to be written.
    - It is written in notepad and saved with .html extension.
3) **Minimal syntax –easy to learn**
    - By learning just few commands and simple rules of syntax JavaScript application can build.
4) **Procedural Capabilities**
    - JavaScript support condition checking, looping, branching etc.
5) **Easy debugging and testing**
    - It is interpreted language so it is tested line by line and if any error is there it will display along with line number.
    - So it is easy to locate errors and make changes.
6) **Platform Independence**
    - JavaScript application work on any machine.

## 10.2 Select Developing Environment for JavaScript

With simple text editor such as notepad, you can create JavaScript program.

Following are some other tools also

1) **Microsoft Frontpage**
2) **Adobe Deramweaver**
3) **Adobe GoLive**

## 10.3 HTML and Java Script

There are 3 possibilities to integrate the JavaScript code into the HTML file.

1. Integrated script under the <head> tag
2. Integrated script under the <body> tag
3. Importing the external JavaScript

### 1. Integrated script under the <head> tag

The first possibility to incorporate the source code under the <head> tag of HTML file.

JavaScript in the head section will execute when called.

Example

```
<html>

<head>

<script type=”text/JavaScript”>

…..

…..

</script>

</head>
```

### 2. Integrated script under the <body> tag

When you place JavaScript code under the <body> tag, this generates the contents of the web page.

JavaScript code under the <body> tag executes when the web page loads and go in the body section.

Example

<html>

<head>

</head>

<body>

<script type=”text/JavaScript”>

…..

…..

</script>

</body>

### 3. Importing the external JavaScript

You can import an external JavaScript file, when you want to run the same JavaScript file on several HTML files, without having to write the same JavaScript code on every HTML file.

**External JavaScript file should save with .js extension.**

Example

<html>

<head>

<script src=”abc.js”>

</script>

</head>

**First JavaScript Program**

The javascript code used <script>..</script> tags to start and end the code.

To print something

```
Document.write("hello");
```

## 10.4 Elements of Java Script

### 1. JavaScript Statements

JavaScript statements are embedded in between <script>and </script> tags.

JavaScript statements are the single line codes in between the <script>and </script> tags.

JavaScript statements are separated by **semicolons**.

```
<script type="text/JavaScript">
document.write("hello");
</script>
```

### 2. JavaScript Statement Block

Several statements grouped together in a statement block. The purpose of statements block is to make the sequence of statements execute together.

**The statement block begins and ends with open and close brackets.**

**Example**

```
<script type="text/JavaScript">
{
document.write("<h1>This is a header</h1>");
document.write("<p>This is a paragraph</p>");
document.write("<h2>This is a header2</h2>");
} </script>
```

### 3. JavaScript Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable.

**1. Single-Line Comments**

Single line comments start with //.

Any text between // and the end of the line, will be ignored by JavaScript (will not be executed).

**2. Multi-line Comments**

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored by JavaScript.

## 10.5 Java Script Variables

JavaScript variables are containers for storing data values.

Variables are temporary store data and have a name, value and memory address.

**The general rules** for constructing names for variables (unique identifiers) are:

1. Names can contain letters, digits, underscores, and dollar signs.
2. Names must begin with a letter
3. Names can also begin with $ and _ (but we will not use it in this tutorial)
4. Names are case sensitive (y and Y are different variables)
5. Reserved words (like JavaScript keywords) cannot be used as names

**Declaring Variables**

- Before you can use variable for storing any data, it has to be declared.
- **Syntax**

  Var variable name;

**Assigning values and Accessing Variables**

- You can assign values to variables while declaring them or after declaring them.
- Once you assign value to a variable, you can access the value and use it.

- Syntax

  Var variable name=value;

## 10.6 Types of Data in Java Script

Data types denoted the kind of data.

Depending on the type of data, the data is processed in different ways by JavaScript.

The basic data types available in JavaScript are string, number and Boolean.

**1. String Data type**

- A string (or a text string) is a series of characters like "abc".
- Strings are written with quotes. You can use single or double quotes.
- You can make the use of escape character.

| Code | Outputs |
|---|---|
| \' | single quote |
| \" | double quote |
| \\ | backslash |
| \n | new line |
| \r | carriage return |
| \t | tab |
| \b | backspace |

**You can use quotes inside a string, as long as they don't match the quotes surrounding the string:**

**2. Number Data type**

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals

**Example**

```
var x1 = 34.00;     // Written with decimals
var x2 = 34;        // Written without decimals
```

Extra large or extra small numbers can be written with scientific (exponential) notation:

**Example**

```
var y = 123e5;      // 12300000
var z = 123e-5;     // 0.00123
```

### 3. Boolean Data type

Booleans can only have two values: true or false.

Booleans are often used in conditional testing.

**Example**

```
var x = true;
var y = false;
```

## 10.7 Java Script Operators

JavaScript operators are symbols that are used to perform operations on operands.

JavaScript operators are used to assign values, compare values, perform arithmetic operations, and more.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators

### 1. Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands.

The following operators are known as JavaScript arithmetic operators.

Given that y = 5, the table below explains the arithmetic operators:

| Operator | Description | Example | Result in y | Result in x |
|---|---|---|---|---|
| + | Addition | x = y + 2 | y = 5 | x = 7 |
| - | Subtraction | x = y - 2 | y = 5 | x = 3 |
| * | Multiplication | x = y * 2 | y = 5 | x = 10 |
| / | Division | x = y / 2 | y = 5 | x = 2.5 |
| % | Modulus (Remainder) | x = y % 2 | y = 5 | x = 1 |
| ++ | Increment | x = ++y | y = 6 | x = 6 |
| | | x = y++ | y = 6 | x = 5 |
| -- | Decrement | x = --y | y = 4 | x = 4 |
| | | x = y-- | y = 4 | x = 5 |

## 2. Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that x = 10 and y = 5, the table below explains the assignment operators:

| **Operator** | **Example** | **Same As** | **Result in x** |
|---|---|---|---|
| = | x = y | x = y | x = 5 |
| += | x += y | x = x + y | x = 15 |
| -= | x -= y | x = x - y | x = 5 |
| *= | x *= y | x = x * y | x = 50 |
| /= | x /= y | x = x / y | x = 2 |
| %= | x %= y | x = x % y | x = 0 |

### 3. Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that x = 6 and y = 3, the table below explains the logical operators:

| Operator | Description | Example |
|---|---|---|
| && | and | (x < 10 && y > 1) is true |
| \|\| | or | (x == 5 \|\| y == 5) is false |
| ! | not | !(x == y) is true |

### 4. Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

| **Operator** | **Description** | **Example** |
|---|---|---|
| == | Is equal to | 10==20 = false |
| != | Not equal to | 10!=20 = true |
| !== | Not Identical | 20!==20 = false |
| > | Greater than | 20>10 = true |
| >= | Greater than or equal to | 20>=10 = true |
| < | Less than | 20<10 = false |

### 5. Bitwise Operators

Bit operators work on 32 bits numbers. Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

| Operator | Description | Example |
|---|---|---|
| & | Bitwise AND<br>It performs a Boolean AND operation on each bit of its integer arguments. | (10==20 & 20==33) = false |
| \| | Bitwise OR<br>It performs a Boolean OR operation on each bit of its integer arguments. | (10==20 \| 20==33) = false |
| ^ | Bitwise XOR<br>It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both. | (10==20 ^ 20==33) = false |
| ~ | Bitwise NOT<br>It is a unary operator and operates by reversing all the bits in the operand. | (~10) = -10 |
| << | Bitwise Left Shift<br>It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on. | (10<<2) = 40 |
| >> | Bitwise Right Shift<br>Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand. | (10>>2) = 2 |
| >>> | Bitwise Right Shift with Zero<br>This operator is just like the >> operator, except that the bits shifted in on the left are always zero. | (10>>>2) = 2 |

## 6. Conditional (Ternary) Operator

The conditional operator assigns a value to a variable based on a condition.

| Syntax | Example | Description |
|---|---|---|
| variablename = (condition) ? value1:value2 | voteable = (age < 18) ? "Too young":"Old enough"; | If the variable "age" is a value below 18, the value of the variable "voteable" will be "Too young", otherwise the value of voteable will be "Old enough". |

## 7. The typeof Operator

The **typeof** operator returns the type of a variable, object, function or expression:

**Example**

```
typeof "John"                 // Returns string
typeof 3.14                   // Returns number
typeof false                  // Returns boolean
typeof new Date()             // Returns object
typeof function () {}         // Returns function
typeof myCar                  // Returns undefined (if myCar is not declared)
typeof null                   // Returns object
```

## 8. The delete Operator

The **delete** operator deletes a property from an object

After deletion, the property cannot be used before it is added back again.

The delete operator deletes both the value of the property and the property itself.

**Example**

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
delete person.age;   // or delete person["age"];
```

# 10.8 Java Script Flow Control Statements

## If-else

The JavaScript if-else statement is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

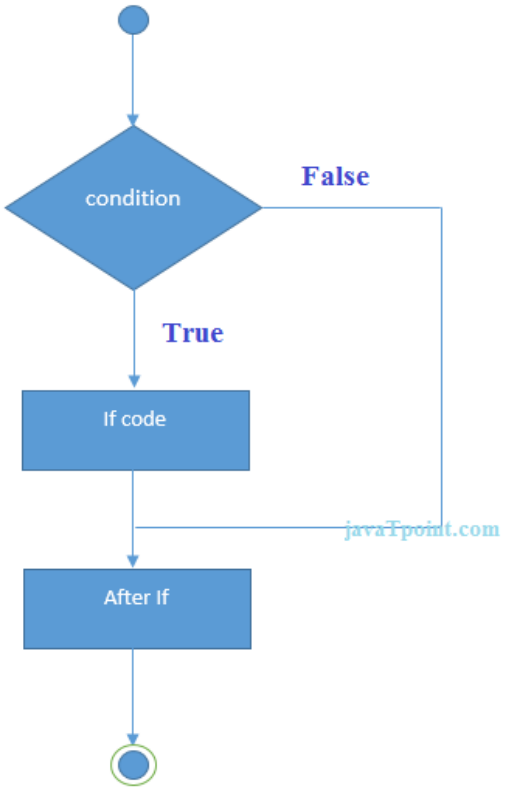
1. If Statement
2. If else statement
3. if else if statement

## JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if(expression){

//content to be evaluated

}
```

The following flow chart shows how the if-else statement works.

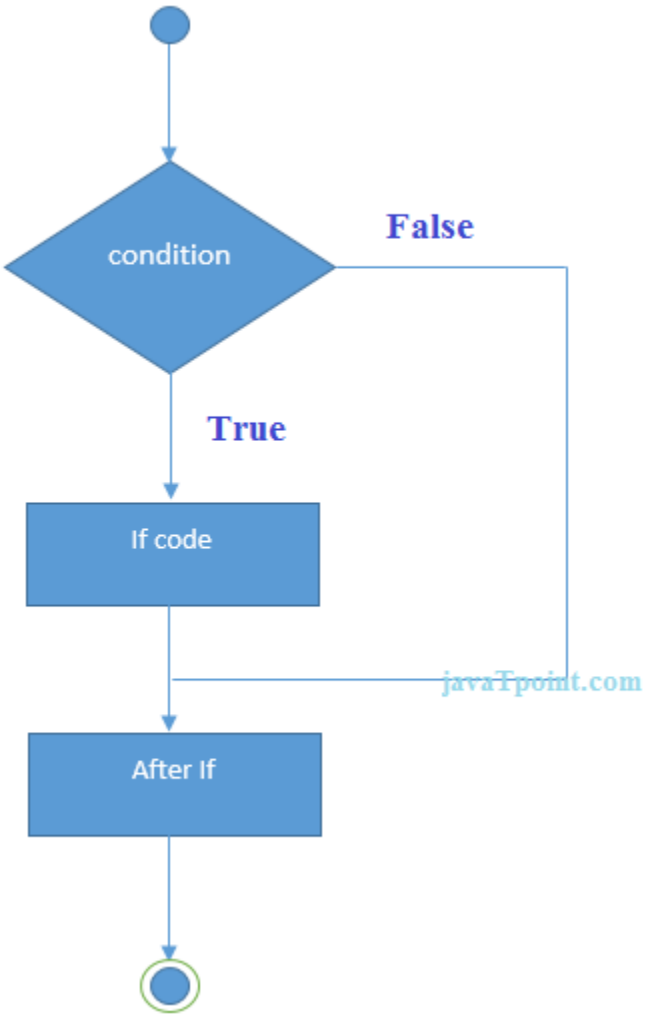


## JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

```
if(expression){

//content to be evaluated if condition is true

}

else{

//content to be evaluated if condition is false

}
```

## Flowchart of JavaScript If...else statement



## JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

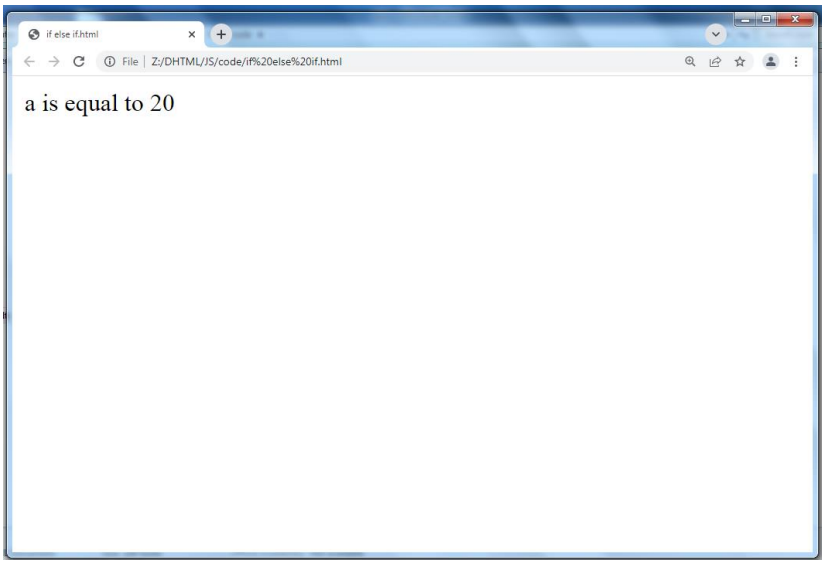
```
if(expression1){

//content to be evaluated if expression1 is true

}

else if(expression2){

//content to be evaluated if expression2 is true

}

else if(expression3){

//content to be evaluated if expression3 is true

}

else{

//content to be evaluated if no expression is true

}
```

Let’s see the simple example of if else if statement in javascript.

```
<script>

var a=20;

if(a==10){

document.write("a is equal to 10");

}

else if(a==15){

document.write("a is equal to 15");

}

else if(a==20){

document.write("a is equal to 20");
```

```
}
else{
document.write("a is not equal to 10, 15 or 20");
}
</script>
```

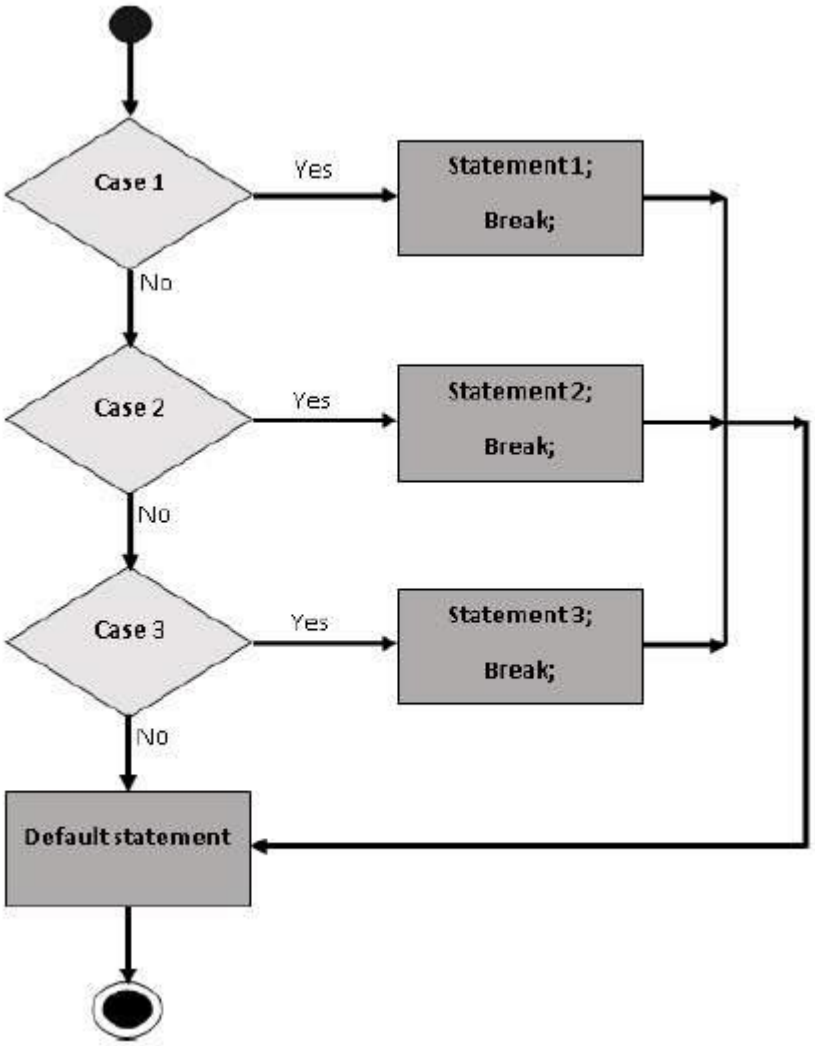
Output for above code




## Switch Case

### Flow Chart

The following flow chart explains a switch-case statement works.




## Syntax

The objective of a switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

```
switch (expression)
{
  case condition 1: statement(s)
  break;

  case condition 2: statement(s)
  break;
  ...

  case condition n: statement(s)
  break;

  default: statement(s)
```

}

The break statements indicate the end of a particular case. If they were omitted, the interpreter would continue executing each statement in each of the following cases.

## While Loops

The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.

**Syntax**

The syntax of while loop in JavaScript is as follows –

```
while (expression){
   Statement(s) to be executed if expression is true
}
```

**The do...while Loop**

The do...while loop is similar to the while loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is false.

**Syntax**

The syntax for do-while loop in JavaScript is as follows –

```
do{
   Statement(s) to be executed;
} while (expression);
```

Note − Don’t miss the semicolon used at the end of the do...while loop.

## For Loop

The 'for' loop is the most compact form of looping. It includes the following three important parts −

- **The loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- **The test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- **The iteration statement** where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

### Syntax

The syntax of for loop is JavaScript is as follows −

```
for (initialization; test condition; iteration statement){
   Statement(s) to be executed if test condition is true
}
```

## Loop Control

JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching at its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the look.

To handle all such situations, JavaScript provides **break** and **continue** statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop respectively.

### The break Statement

The break statement, which was briefly introduced with the switch statement, is used to exit a loop early, breaking out of the enclosing curly braces.

### The continue Statement

The continue statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block.

When a continue statement is encountered, the program flow moves to the loop check expression immediately and if the condition remains true, then it starts the next iteration, otherwise the control comes out of the loop.

### Using Labels to Control the Flow

A label can be used with break and continue to control the flow more precisely. A label is simply an identifier followed by a colon (:) that is applied to a statement or a block of code. We will see two different examples to understand how to use labels with break and continue.

Note − Line breaks are not allowed between the ‘continue’ or ‘break’ statement and its label name. Also, there should not be any other statement in between a label name and associated loop.

## 10.9 Summary

In this chapter you have learned about:

- Introduction to Java Script
- Select Developing Environment for Java Script
- HTML and Java Script
- Elements of Java Script
- Java Script Variables
- Types of Data in Java Script
- Java Script Operators
- Java Script Flow Control Statements

# Unit 11: Arrays, Functions, Events and Dialog boxes in Java Script

## Unit Structure


11.1. Java Script Arrays

11.2. Java Script Functions

11.3. Java Script Popup Boxes

11.4. Java Script Events

11.5. Summary

# 11.1 Java Script Arrays

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

**1) JavaScript array literal**

- The syntax of creating array using array literal is given below:

  var arrayname=[value1,value2.....valueN];

- values are contained inside [ ] and separated by , (comma).

**2) JavaScript Array directly (new keyword)**

The syntax of creating array directly is given below:

var arrayname=new Array();

Here, new **keyword is used to create instance of array.**

**3) JavaScript array constructor (new keyword)**

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

# 11.2 Java Script Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

**Advantage of JavaScript function**

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

Thereare two types of functions

1. Built-in functions
2. User defined functions

**Built-in functions**

JavaScript has five functions built in to the language. They are eval, parseInt, parseFloat, number, and string.

**1. eval**

Evaluates a string and returns a value.

It converts strings to a numeric value.

eval(Expression)

Examples

```
var num = 2;
 eval("num + 200");
```

**2. parseInt**

Parses a string argument and returns an integer and convert number into integer.

**Syntax:**

parseInt(string)

string is a string that represents the value you want to parse.

**3. parseFloat**

Parses a string argument and returns a floating point number.

If string is not begin with a floating point than it returns NaN error.

**Syntax:**

parseFloat(string)

**4. Number**

It converts corresponding value to number or give error NaN

**5. String**

It converts corresponding value to string.

**User defined functions**

**JavaScript Function Syntax**

The syntax of declaring function is given below.

```
function functionName([arg1, arg2, ...argN]){
 //code to be executed
}
```

JavaScript Functions can have 0 or more arguments.

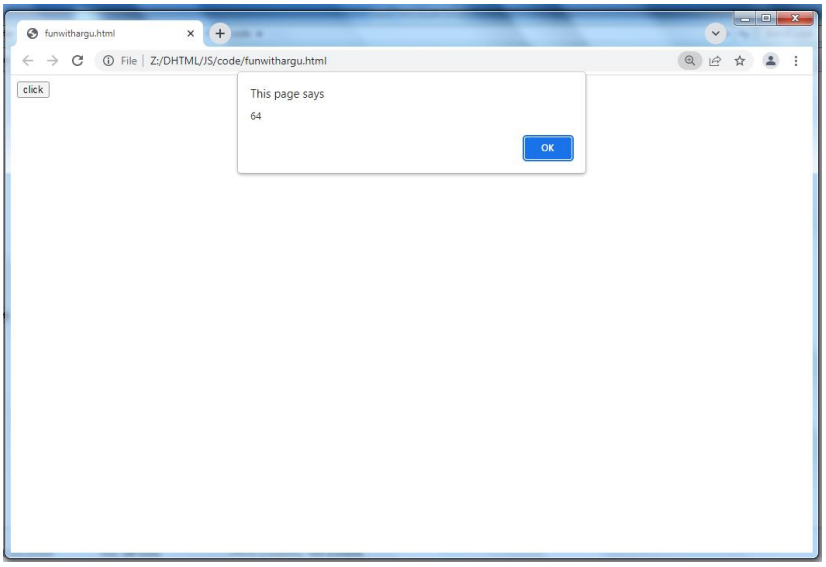
**Function Arguments**

We can call function by passing arguments. Let's see the example of function that has one argument.

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
```

</form>

Output for above code




**Function with Return Value**

We can call function that returns a value and use it in our program.

## 11.3 Java Script Popup Boxes

JavaScript has three kinds of popup boxes: Alert box, Confirm box, and Prompt box.

**Alert Box**

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

**Example**
alert("I am an alert box!");

**Confirm Box**

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

**Prompt Box**

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

**Line Breaks**

To display line breaks inside a popup box, use a back-slash followed by the character n.

Example
alert("Hello\nHow are you?");

## 11.4 Java Script Events

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

| Events | Description |
|---|---|
| onclick | occurs when element is clicked. |

| | |
|---|---|
| ondblclick | occurs when element is double-clicked. |
| onfocus | occurs when an element gets focus such as button, input, textarea etc. |
| onblur | occurs when form looses the focus from an element. |
| onsubmit | occurs when form is submitted. |
| onmouseover | occurs when mouse is moved over an element. |
| onmouseout | occurs when mouse is moved out from an element (after moved over). |
| onmousedown | occurs when mouse button is pressed over an element. |
| onmouseup | occurs when mouse is released from an element (after mouse is pressed). |
| onload | occurs when document, object or frameset is loaded. |
| onunload | occurs when body or frameset is unloaded. |
| onscroll | occurs when document is scrolled. |
| onresized | occurs when document is resized. |
| onreset | occurs when form is reset. |
| onkeydown | occurs when key is being pressed. |
| onkeypress | occurs when user presses the key. |
| onkeyup | occurs when key is released. |

## 11.5 Summary

In this chapter you have learned about:

- Java Script Arrays
- Java Script Functions
- Java Script Popup Boxes
- Java Script Events

Dr. Babasaheb Ambedkar Open University

**BSc(IT)-305**

# Introduction to Web Designing

## BLOCK4: JAVA SCRIPT OBJECTS, JQUERY AND XML

**CHAP 12**

JAVA SCRIPT OBJECTS, METHODS AND PROPERTIES 344

**CHAP 13**

INTRODUCTION TO JQUERY 383

**CHAP 14**

JQUERY SELECTORS, FUNCTIONS, EFFECTS AND EVENTS 388

**CHAP 15**

INTRODUCTION TO XML 490

# BLOCK 4: JAVA SCRIPT OBJECTS, JQUERY AND XML

## Block Introduction

In this block-4 of web technhologies, I have tried to emphasis on: various objects having state and behavior (properties and methods) in java script and also cover jQuery and extensible markup language. Basically, I introduced objects in java script because java script is an object-based language.Along with this I have also include jquery which is java script library simplifies java script programming and XML(extensible markup language) where you can ddefine your own tags and store data .

## Block Objective

The objective of the block is to learn how to create objects in java script, understand properties and methods. Students will also learn about jQuery to simplify HTML DOM traversal and manipulation, event handling, animation and XML to store data in a format that can be stored, searched and shared.

By learning this block of web technology student will learn about adding objects with methods and properties, write code in jquery syntax and describe data in XML. Reader of this block, will know web page development through various objects of java script, jQuery and using XML describes the text in a digital document.

This block servers knowledge of CSS and Java script. I hope, this block will clear the idea of all concepts of objects of javascript, jQuery and XML.

# Unit 12: Java Script Objects, Methods and Properties

## Unit Structure

12.1. Java Script Document Object

12.2. Java Script Array Object

12.3. Java Script String Object

12.4. Java Script Date Object

12.5. Java Script Math Object

12.6. Java Script Window Object

12.7. Summary

# 12.1 Java Script Document Object

The document object represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the root element that represents the html document.

**Methods**

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

| Method | Description |
|---|---|
| write("string") | writes the given string on the doucment. |
| writeln("string") | writes the given string on the doucment with newline character at the end. |
| getElementById() | returns the element having the given id value. |
| getElementsByName() | returns all the elements having the given name value. |
| getElementsByTagName() | returns all the elements having the given tag name. |
| getElementsByClassName() | returns all the elements having the given class name. |

**Accessing the field value by document object**

**Properties**

| Property | Description |
|---|---|
| alinkcolor | The color to be used when displaying activated links in the document |
| anchors | An array of anchor objects present in the document |
| bgColor | The background color of the document |
| fgColor | The foreground color (i.e. the color of the text in the document) |
| forms | An array of the Forms objects contained in the document. |
| images | An Array of Image objects contained the |

| | |
|---|---|
| | document |
| lastModified | The date that the document was last modified |
| linkColor | The color to be used when displaying links in the document |
| links | An array of links contained in the document |
| vlinkColor | The color to be used when displaying links in the document which have been visited by the current user |
| innerHTML | The innerHTML property can be used to modify your document's HTML.When you use innerHTML, you can change the page's content without refreshing the page. |

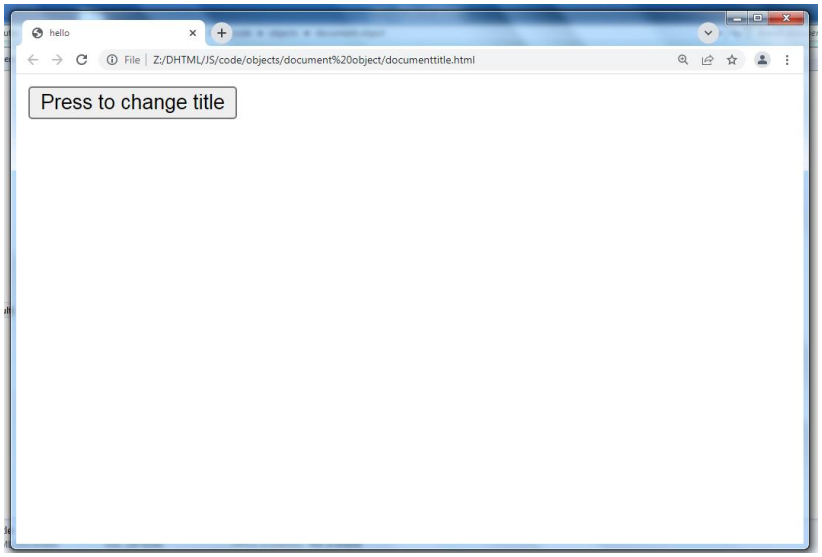
**Changing the Document Title**

Even when the title of a document has been specified using the HTML <title> tag it is possible to dynamically change the current setting using the title property of the document object.

The following example initially sets the document title to JavaScript Document Object Example and provides a button which, when pressed, changes the title to a new value:

```
<html>
<head>
<title>JavaScript Document Object Example</title>
</head>
<body>
<form action="">
<input type=button value="Press to change title" onclick="document.title='hello'"/>
</form>
</body>
</html>
```

Output for above code




**Changing the Document Colors**

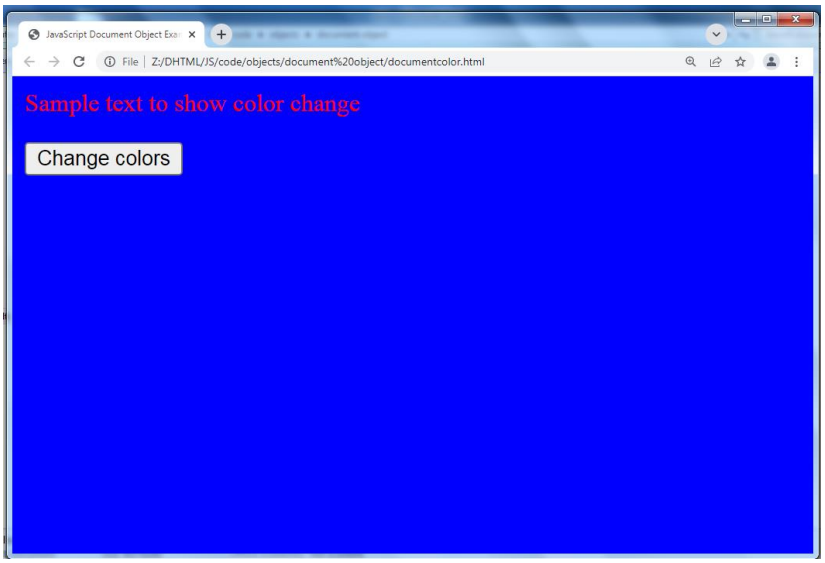
A similar approach can be used to change the colors using in the document.

The following example changes the foreground and background colors when the "Change colors" button is pressed:

```
<html>
<head>
<title>JavaScript Document Object Example</title>
<script language="javascript" type="text/javascript">

function changeColors()
{
	document.fgColor="red";
	document.bgColor="blue";
}
</script>
</head>

<body>
<p>
Sample text to show color change
</p>
<form action="">
<input type=button value="Change colors" onclick="changeColors()"/>
</form>
</body>
```

</html>

Output for above code




**Getting a List of Objects in a Document**

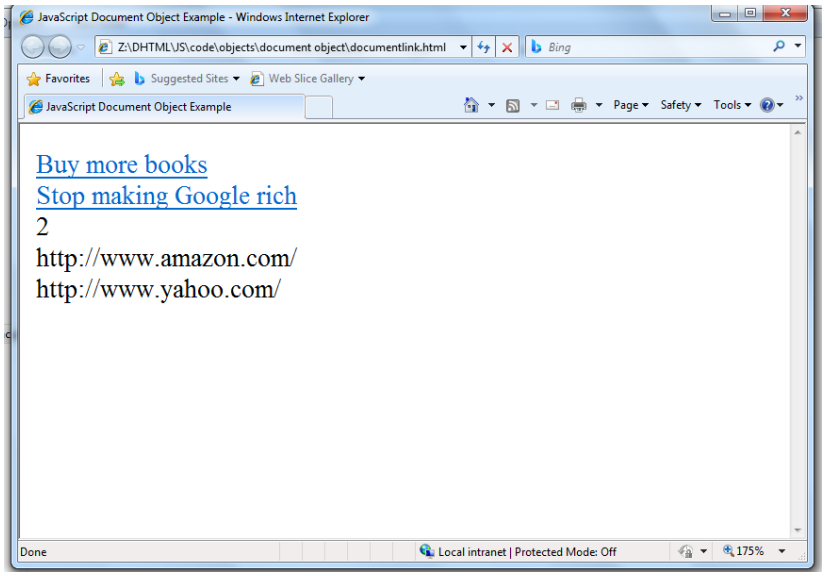
the links property of the document object can be used to obtain a list of Link objects (in other words, links to other web pages) in the current document.

To demonstrate this, the following example consists of a simple HTML page containing two links to other sites (Amazon.com and Yahoo). The J

```
<html>
<head>
<title>JavaScript Document Object Example</title>
</head>

<body>

<a href="http://www.amazon.com">Buy more books</a>
<br>
<a href="http://www.yahoo.com">Stop making Google rich</a>
<br>

<script language="javascript" type="text/javascript">
```

```
for (i in document.links)
{
      document.write( document.links[i] + "<br>" );
}
</script>

</form>
</body>
</html>
```

Output for above code

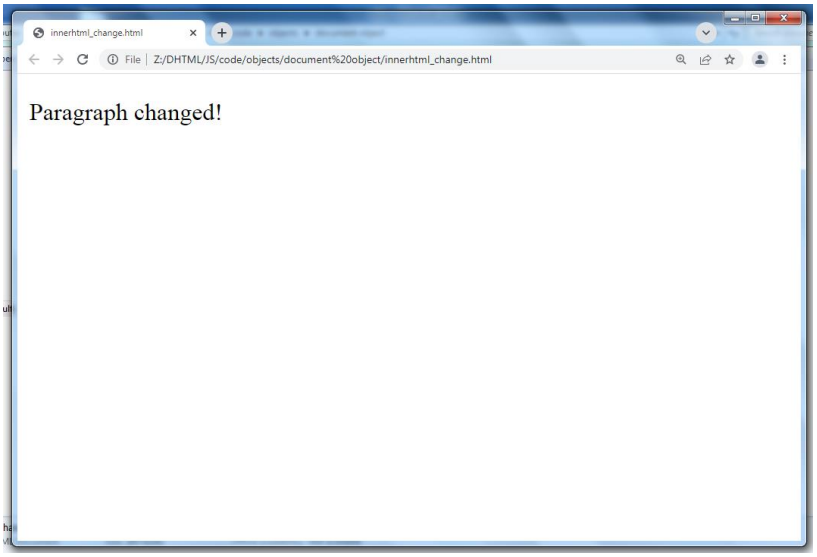



**Inner HTML**

```
<!DOCTYPE html>

<html>

<body>

<p id="demo" onclick="myFunction()">Click me to change my HTML content (innerHTML).</p>

<script>
```

```
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed!";
}
</script>
</body>
</html>
```

Output for above code




## 12.2 Java Script Array Object

Describes the JavaScript array object including parameters, properties, and methods.

**Parameters**

- arrayLength
- elementN - Array element list of values

**Properties**

- index
- input
- length - The quantity of elements in the object.

**Methods**

- **concat()** -Join two arrays:

```
<html>
<head>
<title>JavaScript Array concat Method</title>
</head>

<body>

<script type="text/javascript">
        var alpha = ["a", "b", "c"];
        var numeric = [1, 2, 3];

        var alphaNumeric = alpha.concat(numeric);
        document.write("alphaNumeric : " + alphaNumeric );
</script>

</body>
</html>
```

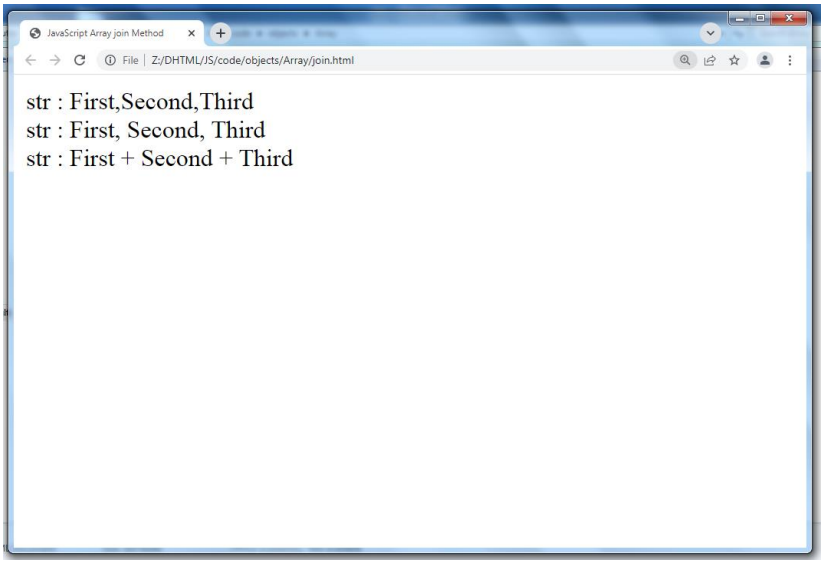
Output for above code

- **join(delimiter)** - Puts all elements in the array into a string, separating each element with the specified delimiter.

```
<html>
<head>
<title>JavaScript Array join Method</title>
</head>
<body>
<script type="text/javascript">
        var arr = new Array("First","Second","Third");
         var str = arr.join();
        document.write("str : " + str );
var str = arr.join(", ");
        document.write("<br />str : " + str );
        var str = arr.join(" + ");
        document.write("<br />str : " + str );
</script>
</body>
```

```
</html>
```

Output for above code

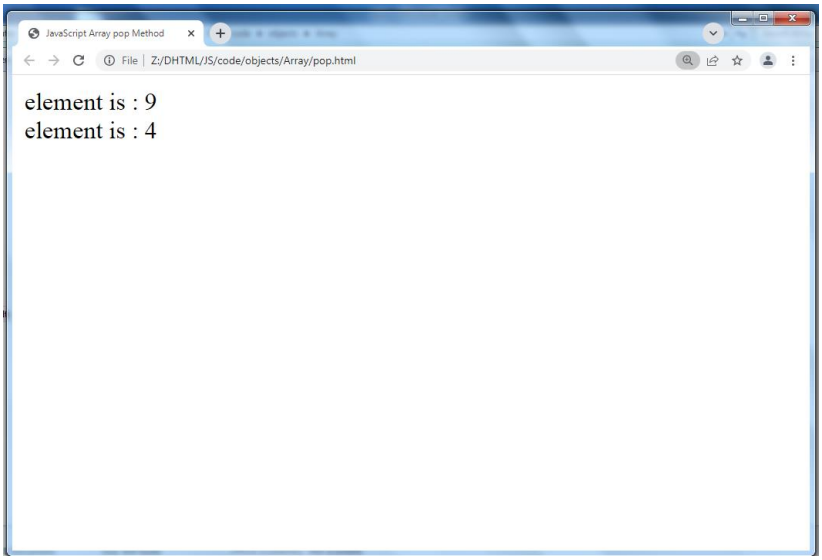



- **pop()** - Pops the last string off the array and returns it.

```
<html>
<head>
<title>JavaScript Array pop Method</title>
</head>
<body>
<script type="text/javascript">
        var numbers = [1, 4, 9];
var element = numbers.pop();
        document.write("element is : " + element );

        var element = numbers.pop();
        document.write("<br />element is : " + element );
```

```
</script>


</body>
</html>
```

Output for above code

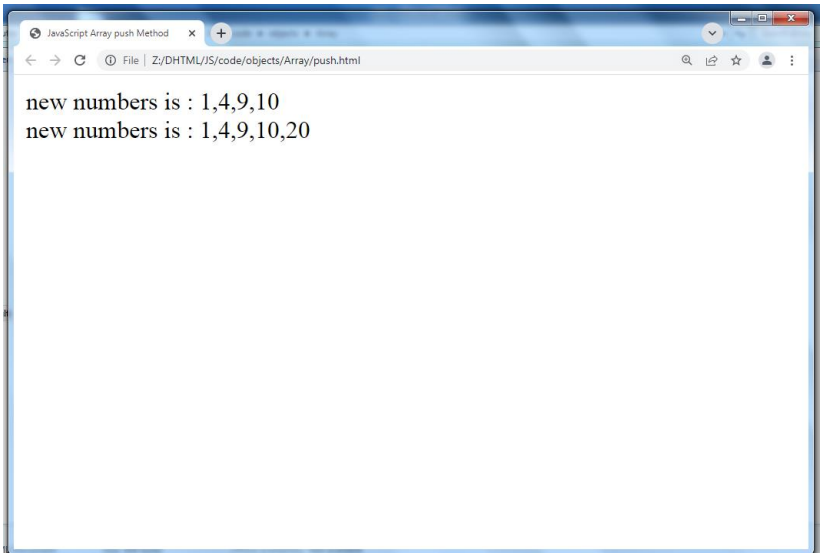



- **push(strings)** - Strings are placed at the end of the array.

```
<html>
<head>
<title>JavaScript Array push Method</title>
</head>
<body>
<script type="text/javascript">
        var numbers = new Array(1, 4, 9);


        var length = numbers.push(10);
        document.write("new numbers is : " + numbers );
```

```
        length = numbers.push(20);
        document.write("<br />new numbers is : " + numbers );
    </script>
    </body>
    </html>
```

Output for above code

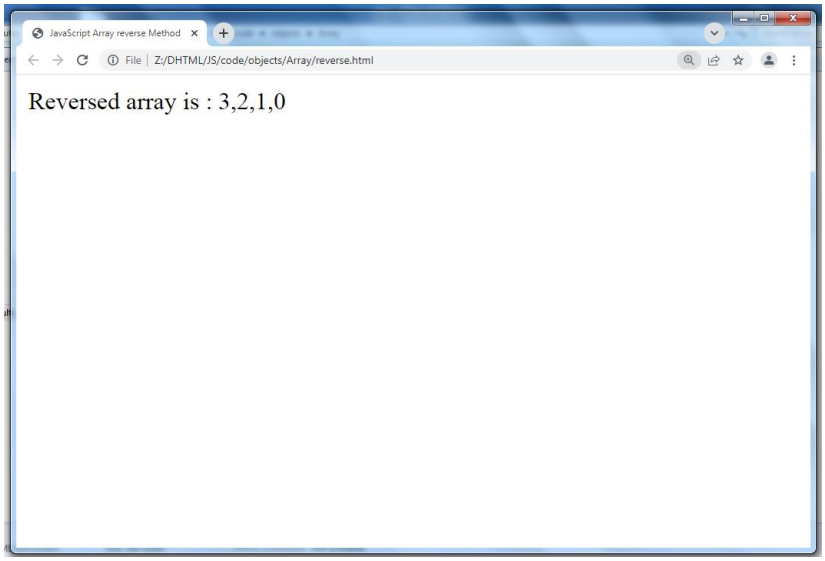



- **reverse()** - Puts array elements in reverse order.

```
<html>
<head>
<title>JavaScript Array reverse Method</title>
</head>
<body>
<script type="text/javascript">
        var arr = [0, 1, 2, 3].reverse();
```

```
    document.write("Reversed array is : " + arr );
</script>
</body>
</html>
```

Output for above code

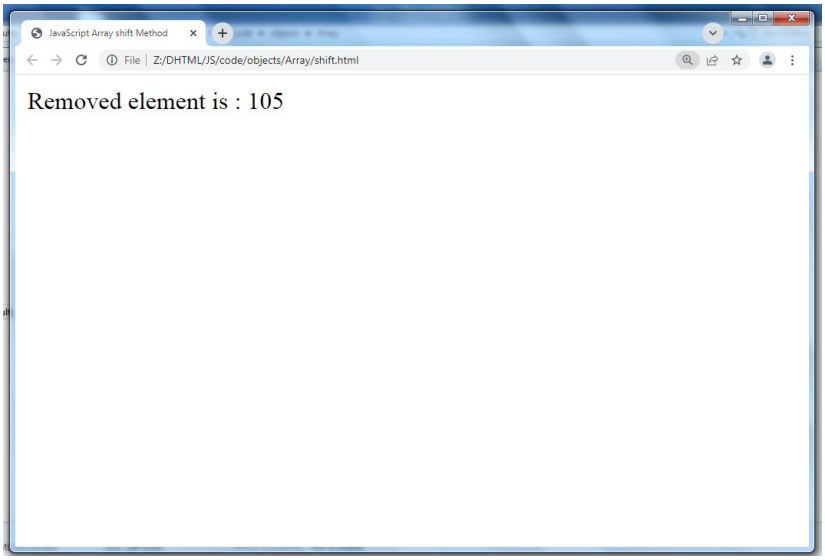



- **shift()** - Decreases array element size by one by shifting the first element off the array and returning it.

```
<html>
<head>
<title>JavaScript Array shift Method</title>
</head>

<body>

<script type="text/javascript">
    var element = [105, 1, 2, 3].shift();
    document.write("Removed element is : " + element );
</script>

</body>
```

```
</html>
```

Output for above code

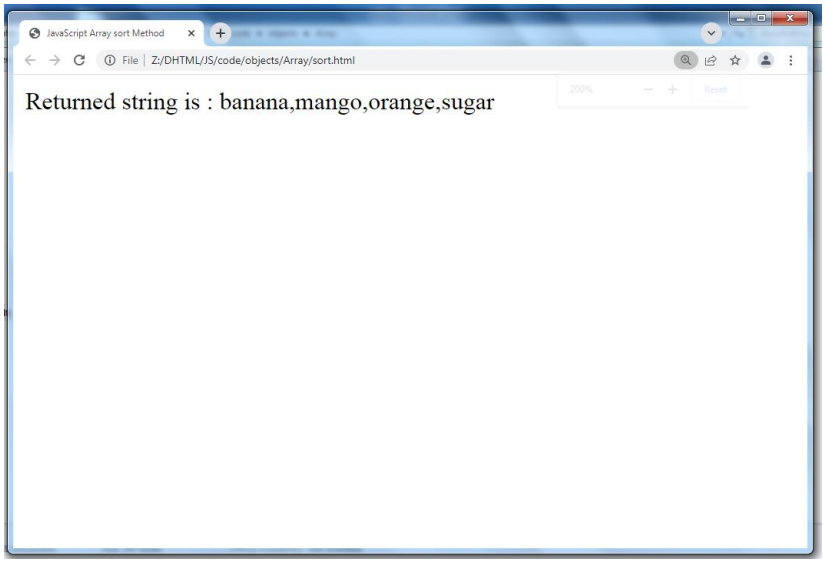



- **sort()** - Sorts the array elements in dictionary order

```
<html>
<head>
<title>JavaScript Array sort Method</title>
</head>
<body>
<script type="text/javascript">
        var arr = new Array("orange", "mango", "banana", "sugar");

        var sorted = arr.sort();
        document.write("Returned string is : " + sorted );
</script>
```

```
</body>

</html>
```

Output for above code

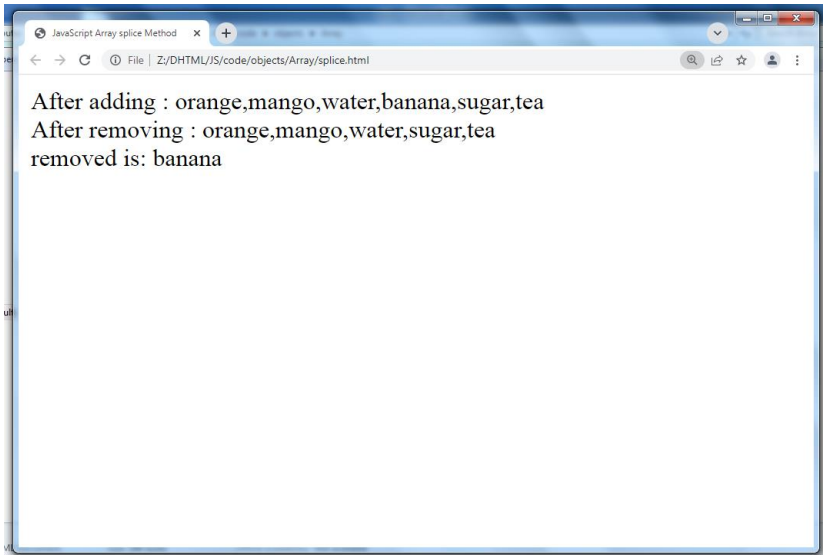



- **splice()** - It is used to take elements out of an array and replace them with those specified. In the below example the element starting at element 3 is removed, two of them are removed and replaced with the specified strings. The value returned are those values that are replaced.

```
<html>

<head>

<title>JavaScript Array splice Method</title>

</head>

<body>

<script type="text/javascript">

    var arr = ["orange", "mango", "banana", "sugar", "tea"];

    var removed = arr.splice(2, 0, "water");

    document.write("After adding : " + arr );
```

```
        // document.write("<br />removed is: " + removed);
         removed = arr.splice(3, 1);
         document.write("<br />After removing : " + arr );
         document.write("<br />removed is: " + removed);
</script>
</body>
</html>
```

Output for above code




- **split(deliimiter)** - Splits a string using the delimiter and returns an array.

```
Var words = new String("limit;lines;finish;complete;In;Out")
var swords = words.split(";")

   document.write(swords);
```

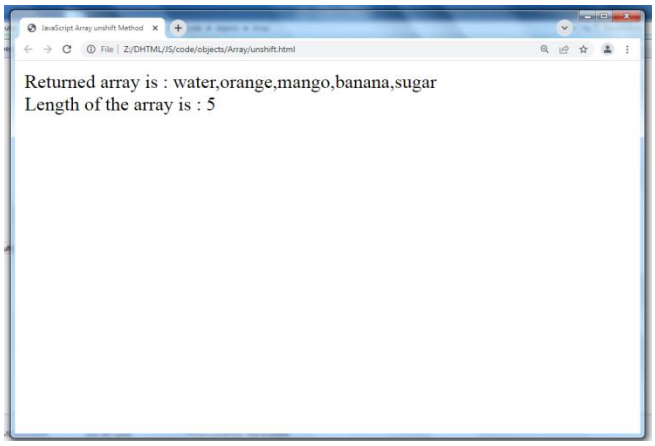
The values in the array swords is:

limit, lines, finish, complete, In, Out

- **unshift()** -method adds one or more elements to the beginning of an array and returns the new length of the array.

```
<html>
<head>
<title>JavaScript Array unshift Method</title>
</head>
<body>
<script type="text/javascript">
    var arr = new Array("orange", "mango", "banana", "sugar");
        var length = arr.unshift("water");
    document.write("Returned array is : " + arr );
    document.write("<br /> Length of the array is : " + arr.length );
</script>
</body>
</html>
```

Output for above code



# 12.3 Java Script String Object

The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

**Syntax**

Use the following syntax to create a String object −

var val = new String(string);

The String parameter is a series of characters that has been properly encoded.

**String Properties**

Here is a list of the properties of String object and their description.

| Property | Description |
|---|---|
| length | Returns the length of the string. |

**String Methods**

Here is a list of the methods available in String object along with their description.

| Method | Description | Example |
|---|---|---|
| charAt() | Returns the character at the specified index. | var str = new String( "This is string" );<br>document.writeln("str.charAt(0) is:" + str.charAt(0));<br>document.writeln("<br/>str.charAt(1) is:" + str.charAt(1));<br>document.writeln("<br/>str.charAt(2) is:" + str.charAt(2));<br>Output<br>str.charAt(0) is:T<br>str.charAt(1) is:h<br>str.charAt(2) is:i |
| concat() | Combines the text of two strings and returns a new string. | <script type="text/javascript"><br>var str1 = new String( "This is string one" );<br>var str2 = new String( "This is string two" );<br>var str3 = str1.concat( str2 );<br>document.write("Concatenated String :" + str3);<br></script><br>**Output**<br>Concatenated String :This is string oneThis is string two. |
| indexOf() | Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found. | <script type="text/javascript"><br>var str1 = new String( "This is string one" );<br>var index = str1.indexOf( "string" );<br>document.write("indexOf found String :" + index );<br>document.write("<br />");<br>var index = str1.indexOf( "one" );<br>document.write("indexOf found String :" + index );<br></script><br>**Output** |

| | | |
|---|---|---|
| | | indexOf found String :8<br>indexOf found String :15 |
| lastIndexOf() | Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found. | <script type="text/javascript"><br>  var str1 = new String( "This is string one and again string" );<br>  var index = str1.lastIndexOf( "string" );<br>  document.write("lastIndexOf found String :" + index );<br><br>  document.write("<br />");<br><br>  var index = str1.lastIndexOf( "one" );<br>  document.write("lastIndexOf found String :" + index );<br></script><br>**Output**<br>lastIndexOf found String :29<br>lastIndexOf found String :15 |
| replace() | Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring. | myString = new String ("This is my string");<br>var newString = myString.replace ("my", "your");<br>Output<br>'This is your string' |
| slice() | Extracts a section of a string and returns a new string. | <script><br>var s1="abcdefgh";<br>var s2=s1.slice(2,5);<br>document.write(s2);<br></script><br></body> |

| | | |
|---|---|---|
| | | **output**<br><br>cde |
| split() | Splits a String object into an array of strings by separating the string into substrings. | <script type="text/javascript"><br>var str = "Apples are round, and apples are juicy.";<br>var splitted = str.split(" ", 3);<br><br>document.write( splitted );<br></script><br><br>**Output**<br>Apples,are,round, |
| substr() | Returns the characters in a string beginning at the specified location through the specified number of characters. | <script type="text/javascript"><br>var str = "Apples are round, and apples are juicy.";<br><br>document.write("(1,2): " + str.substr(1,2));<br>document.write("<br />(1): " + str.substr(1));<br>document.write("<br />(20, 2): " +<br>str.substr(20,2));<br></script><br><br></body><br></html><br><br>**Output**<br>(1,2): pp<br>(1): pples are round, and apples are juicy.<br><br>(20, 2): d |
| substring() | Returns the characters in a string between two indexes into the string. | myString = new String ("This is my string");<br>var partString = myString.substring(5, 10);<br><br>**output**<br><br>'is my' |
| toLowerCase() | Returns the calling string value | <script><br><br>var s1="JavaScript toLowerCase Example"; |

| | | |
|---|---|---|
| | converted to lower case. | var s2=s1.toLowerCase();<br><br>document.write(s2);<br><br></script> |
| toString() | Returns a string representing the specified object. | <script type="text/javascript"><br>var str = "Apples are round, and Apples are Juicy.";<br>document.write(str.toString( ));<br></script><br><br>Output<br>Apples are round, and Apples are Juicy. |
| toUpperCase() | Returns the calling string value converted to uppercase. | <script><br><br>var s1="JavaScript toUpperCase Example";<br><br>var s2=s1.toUpperCase();<br><br>document.write(s2);<br><br></script> |
| Trim() | This method removes leading and trailing whitespaces from the string. | <script><br><br>var s1="  javascript trim  ";<br><br>var s2=s1.trim();<br><br>document.write(s2);<br><br></script> |

## 12.4 Java Script Date Object

The JavaScript date object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

**Constructor**

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

Here is a description of the parameters −

- **No Argument** − With no arguments, the Date() constructor creates a Date object set to the current date and time.
- **milliseconds** − When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned by the getTime() method. For example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70.
- **datestring** − When one string argument is passed, it is a string representation of a date, in the format accepted by the Date.parse() method.( Javascript date.parse() method takes a date string and returns the number of milliseconds since midnight of January 1, 1970.)

**Example**

```
<script>
var msecs = Date.parse( "Aug 28, 2008 23:30:00" );
document.write( "Number of milliseconds from 1970: " + msecs );
</script>

</body>
</html>
```

**Output**

Number of milliseconds from 1970: 1219946400000

7 agruments − To use the last form of the constructor shown above. Here is a description of each argument:

- **year** − Integer value representing the year., you should always specify the year in full; use 1998, rather than 98.
- **month** − Integer value representing the month, beginning with 0 for January to 11 for December.
- **date** − Integer value representing the day of the month.
- **hour** − Integer value representing the hour of the day (24-hour scale).
- **minute** − Integer value representing the minute segment of a time reading.
- **second** − Integer value representing the second segment of a time reading.
- **millisecond** − Integer value representing the millisecond segment of a time reading.

**JavaScript Date Methods**

The important methods of date object are as follows:

| Method | Description |
|---|---|
| getFullYear() | returns the year in 4 digit e.g. 2015. It is a new method and suggested than getYear() which is now deprecated. |
| getMonth() | returns the month in 2 digit from 1 to 31. |
| getDate() | returns the date in 1 or 2 digit from 1 to 31. |
| getDay() | returns the day of week in 1 digit from 0 to 6. |
| getHours() | returns all the elements having the given name value. |
| getMinutes() | returns all the elements having the given class name. |
| getSeconds() | returns all the elements having the given class name. |
| getMilliseconds() | returns all the elements having the given tag name. |

**Example:**

<HTML>

```
<HEAD>
	<TITLE>Welcome Message</TITLE>
</HEAD>
<BODY>
	<SCRIPT language="JavaScript">
	var t = new Date();
	var d = t.getDate();
	var m = t.getMonth()+1;
	var y = t.getFullYear();
	var h = t.getHours();
	if(h<12)
	{
		document.write("Good Morning!!!");
	}
	else if(h>=12 && h<18)
	{
		document.write("Good Afternoon!!!");
	}
	else if(h>=18)
	{
		document.write("Good Evening!!!");
	}
	else if(h>=22)
	{
		document.write("Good Night!!!");
```

```
        }
        document.write("<BR><BR>Today is " + d + " - " + m + " - " + y);
        </SCRIPT>
    </BODY>
</HTML>
```

## 12.5 Java Script Math Object

**Methods**

The following table lists the methods available with the Math object:

| Method | Description | Example |
|---|---|---|
| abs | Absolute value | <script type="text/javascript"><br>var value = Math.abs(-1);<br>document.write("First Test Value : " + value );<br><br>var value = Math.abs(null);<br>document.write("<br />Second Test Value : " + value );<br><br>var value = Math.abs(20);<br>document.write("<br />Third Test Value : " + value );<br><br>var value = Math.abs("string");<br>document.write("<br />Fourth Test Value : " + value );<br></script><br><br>**Output**<br>First Test Value : 1 |

| | | |
|---|---|---|
| | | Second Test Value : 0<br>Third Test Value : 20<br>Fourth Test Value : NaN |
| sin, cos, tan | Standard trigonometric functions; argument in radians | <script type="text/javascript"><br>var value =Math.cos(90);<br>document.write("First Test Value : " + value );<br><br>var value =Math.cos(30);<br>document.write("<br />Second Test Value : " + value );<br>var value = Math.sin( 90 );<br>document.write("<br />Second Test Value : " + value );<br>var value = Math.tan( 45 );<br>document.write("<br />Third Test Value : " + value );<br></script> |
| exp, log | Exponential and natural logarithm, base e | <script type="text/javascript"><br>var value = Math.exp(1);<br>document.write("First Test Value : " + value );<br>var value = Math.log(10);<br>document.write("First Test Value : " + value );<br></script> |
| ceil | Returns least integer greater than or equal to argument | <body><br><br><br>Ceil of 4.6 is:<br><br><script><br><br>document.write(Math.ceil(4.6));<br><br></script><br><br>Output |

| | | |
|---|---|---|
| | | Ceil of 4.6 is: 5 |
| floor | Returns greatest integer less than or equal to argument | <body><br>Floor of 4.6 is:<br><script><br>document.write(Math.floor(4.6));<br></script><br></body><br>Output<br>Floor of 4.6 is: 4 |
| min, max | Returns greater or lesser (respectively) of two arguments | var value = Math.min(10, 20, -1, 100);<br>document.write("First Test Value : " + value );<br><br>var value = Math.min(-1, -3, -40);<br>document.write("<br />Second Test Value : " + value );<br>Output<br>First Test Value : -1<br>Second Test Value : -40<br><br><script type="text/javascript"><br>var value = Math.max(10, 20, -1, 100);<br>document.write("First Test Value : " + value );<br><br>var value = Math.max(-1, -3, -40);<br>document.write("<br />Second Test Value : " + value );<br>Output |

| | | |
|---|---|---|
| | | First Test Value : 100<br>Second Test Value : -1 |
| pow | Exponential; first argument is base, second is exponent | <body><br>3 to the power of 4 is:<br><script><br>document.write(Math.pow(3,4));<br></script><br></body><br>Output<br>3 to the power of 4 is: 81 |
| random | Returns a random number between 0 and 1. | <body><br>Random Number is:<br><script><br>document.write(Math.random());<br></script><br></body> |
| round | Rounds argument to nearest integer | <script type="text/javascript"><br><br>var value = Math.round( 20.7 );<br>document.write("<br />Second Test Value : " + value );<br><br>var value = Math.round( 20.3 );<br>document.write("<br />Third Test Value : " + value );<br><br>var value = Math.round( -20.3 );<br>document.write("<br />Fourth Test Value : " + value );<br></script> |

| | | |
|---|---|---|
| | | </body><br></html><br><br>Output<br>Second Test Value : 21<br>Third Test Value : 20<br>Fourth Test Value : -20 |
| sqrt | Square root | <body><br><br>Square Root of 17 is:<br><br><script><br><br>document.write(Math.sqrt(17));<br><br></script><br><br></body><br><br>Output<br>Square Root of 17 is:<br>4.123105625617661 |

## 12.6 Java Script Window Object

**Methods**

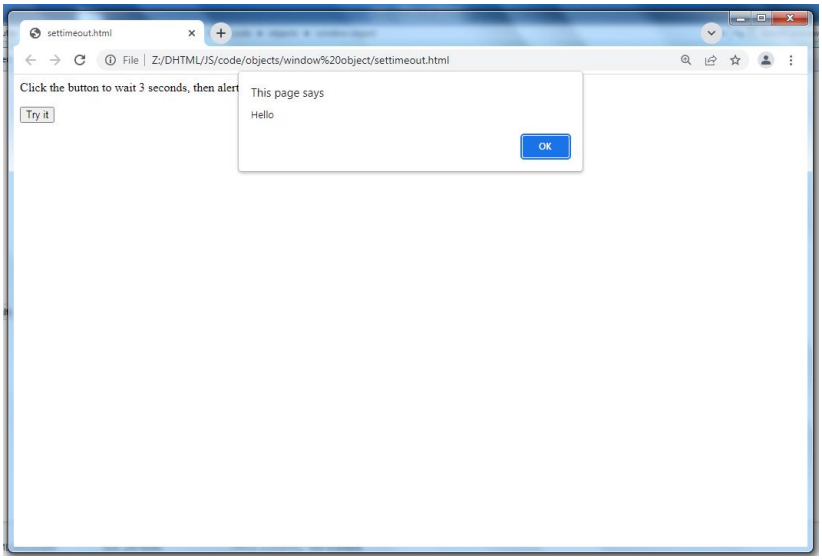
| Method | Description |
|---|---|
| alert() | Displays the alert box containing message with ok button. |
| confirm() | Displays the confirm dialog box containing message with ok and cancel button. |
| prompt() | Displays a dialog box to get input from the user. |

| | |
|---|---|
| open() | Opens the new window. |
| close() | Closes the current window. |
| setTimeout() | It calls a function or evaluates an expression after a specified number of milliseconds. |

**Example**

```
<html>
<body>
<p>Click the button to wait 3 seconds, then alert "Hello".</p>
<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    setTimeout(function(){ alert("Hello"); }, 3000);
}
</script>
</body>
</html>
```

Output for the above code

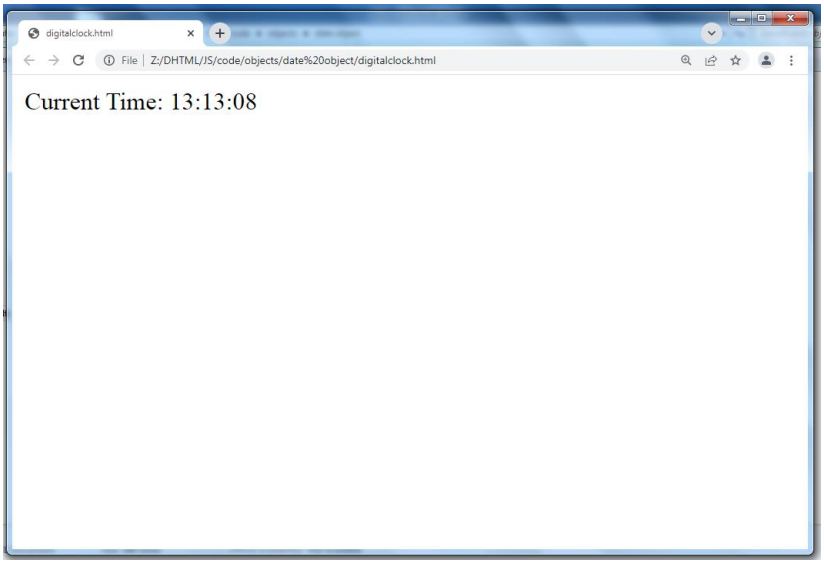



**Example of creating Digital Clock**

```
<html>
<body>
Current Time: <span id="txt"></span>
<script>
window.onload=function(){getTime();}
function getTime(){
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();

// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
```

```
setTimeout(function(){getTime()},1000);
}
//setInterval("getTime()",1000);//another way
function checkTime(i){
if (i<10){
  i="0" + i;
 }
return i;
}
</script>
</body>
</html>
```

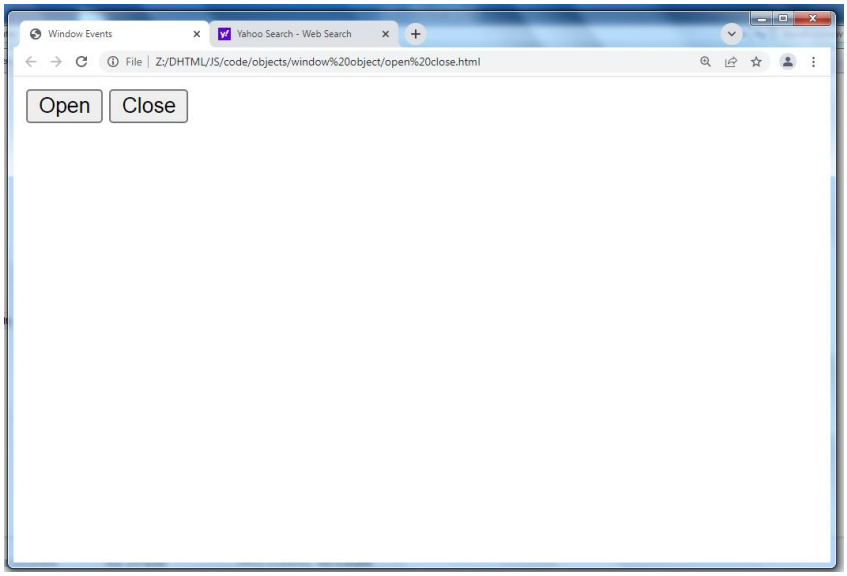
Output for the above code




**Example:**

```
<HTML>
	<HEAD>
		<TITLE>Window Events</TITLE>
	</HEAD>
	<BODY>
		<FORM name="windowEvent">
<INPUT type="button" value="Open" onClick='window.open("http://www.yahoo.com")'>

<INPUT type="button" value="Close" onClick="window.close();">
		</FORM>
	</BODY>
</HTML>
```

Output for the above code



## 12.7 Summary

In this chapter you have learned about:

- Java Script Document Object
- Java Script Array Object
- Java Script String Object
- Java Script Date Object
- Java Script Math Object
- Java Script Window Object

# Unit 13: Introduction to JQuery

## Unit Structure

13.1. What is JQuery

13.2. Features and Advantages of JQuery

13.3. Using JQuery

13.4. JQuery Syntax

13.5. Connecting JQuery to Load Event(Document Ready Event)

13.6. Summary

# 13.1 What is JQuery?

JQuery is a fast, small and feature-rich JavaScript library included in a single .js file.

JQuery makes a web developer's life easy. It provides many built-in functions using which you can accomplish various tasks easily and quickly.

# 13.2 Features and Advantages of JQuery

**DOM Selection:**jQuery provides Selectors to retrieve DOM element based on different criteria like tag name, id, css class name, attribute name, value, nth child in hierarchy etc.

**DOM Manipulation:**You can manipulate DOM elements using various built-in jQuery functions. For example, adding or removing elements, modifying html content, css class etc.

**Special Effects:** You can apply special effects to DOM elements like show or hide elements, fade-in or fade-out of visibility, sliding effect, animation etc.

**Events:** jQuery library includes functions which are equivalent to DOM events like click, dblclick, mouseenter, mouseleave, blur, keyup, keydown etc. These functions automatically handle cross-browser issues.

**Ajax:** jQuery also includes easy to use AJAX functions to load data from servers without reloading whole page.

**Cross-browser support:** jQuery library automatically handles cross-browser issues, so the user does not have to worry about it. jQuery supports IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+.

**Advantages of jQuery:**

1. Easy to learn: jQuery is easy to learn because it supports same JavaScript style coding.
2. Write less do more: jQuery provides a rich set of features that increase developers' productivity by writing less and readable code.

3. Excellent API Documentation: jQuery provides excellent online API documentation.
4. Cross-browser support: jQuery provides excellent cross-browser support without writing extra code.
5. Unobtrusive: jQuery is unobtrusive which allows separation of concerns by separating html and jQuery code.

## 13.3 Using JQuery

**You can obtain jQuery from http://www.jquery.com/.**

### 1. The Two jQuery Downloads

On the jQuery home page, two downloads are available: **a production version and a development version.** Unless you're planning to develop jQuery plug-ins, or need to look at the internals of jQuery, you should download and use the minified production version.

As another viable option, especially for working through this chapter, you could use a content delivery network (CDN) to access a hosted version of jQuery. Google hosts jQuery and other libraries through its API website. This means that you can include jQuery in your webpage and JavaScript programs without having to host the file locally on your server. See http://code.google.com/apis/libraries/devguide.html for more information.

### 2. Including jQuery

You include jQuery in a webpage in the same manner as you would any other external JavaScript file—with a <script> tag pointing to the source file. Consider the code in Listing

**Example**

Including jQuery in a webpage.

```
<!DOCTYPE HTML >
<html>
<head>
<title>Adding jQuery</title>
```

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
</head>
<body>
</body>

</html>
```

## 13.4 JQuery Syntax

The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).

**Basic syntax is:** $(selector).action()

- A $ sign to define/access jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

**Examples:**

$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

## 13.5 Connecting JQuery to Load Event (Document Ready)

One of the most common ways to work with jQuery is by connecting to elements during the load (or onload) event of the page.

In jQuery, you do this through the .ready() utility function of the document element.

**Example**

```
$(document).ready
```

```
$(document).ready(function(){

        // jQuery methods go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

```
$(function(){

  // jQuery methods go here...

});
```

## 13.6 Summary

In this chapter you have learned about:

- What is JQuery
- Features and Advantages of JQuery
- Using JQuery
- JQuery Syntax
- Connecting JQuery to Load Event(Document Ready Event)

# Unit 14: JQuery Selectors, Functions, Effects and Events

## Unit Structure


14.1. JQuery Selectors

14.2. JQuery Traversing

14.3. JQuery Attributes

14.4. JQuery Effects

14.5. JQuery Events

14.6. Summary

# 14.1 JQuery Selectors

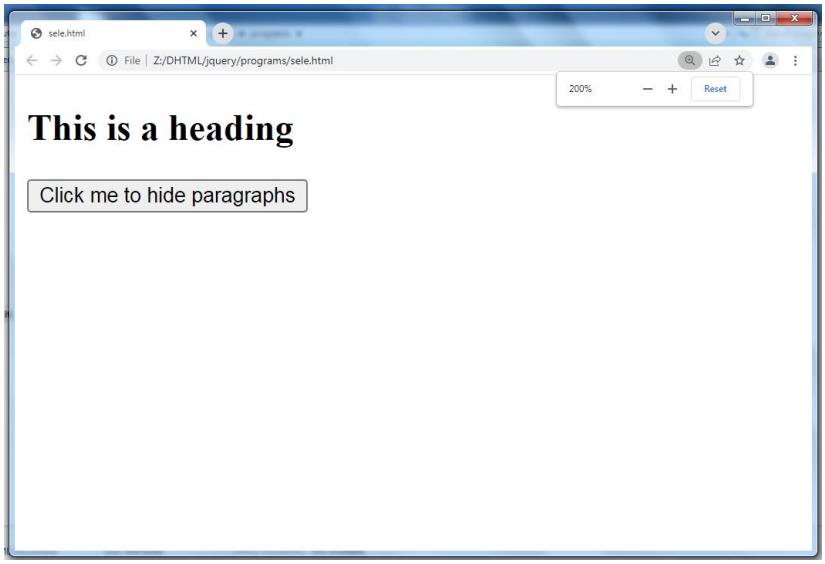
jQuery selectors allow you to select and manipulate HTML element(s).jQuery selectors are used to "find" (or select) HTML elements based

**All selectors in jQuery start with the dollar sign and parentheses: $().**

```
<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<script type="text/javascript">

$(document).ready(function(){

    $("button").click(function(){

        $("p").hide();

    });

});

</script>

</head>
<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me to hide paragraphs</button>
</body>
```

</html>

**Output for above code**




## 1. Selecting element by id ( #id Selector)

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

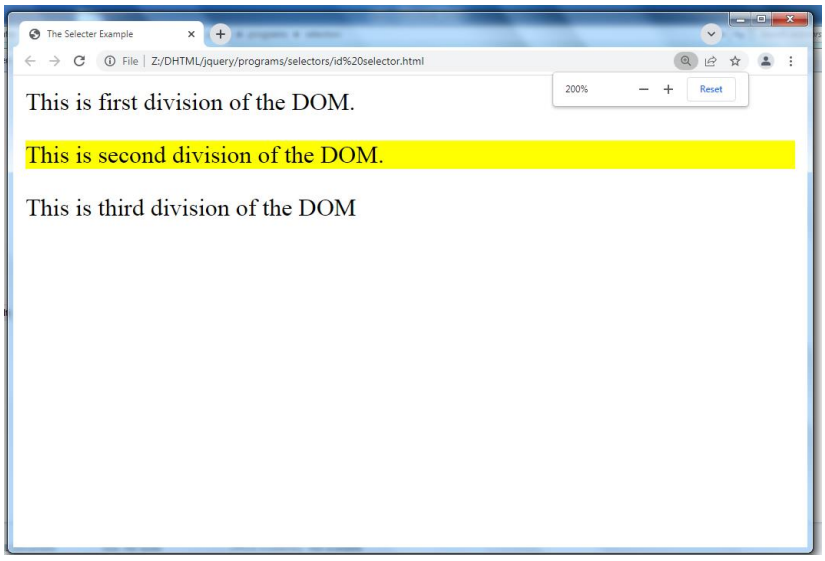
**To find an element with a specific id, write a hash character, followed by the id of the HTML element:**

**Example:** $("#test")

```
<html>
<head>
<title>The Selecter Example</title>
<script src="jquery-3.3.1.js"></script>
<script type="text/javascript">
        $(document).ready(function() {
```

```
        /* This would select second division only*/
        $("#div2").css("background-color", "yellow");
    });
</script>
</head>

<body>
<div class = "big" id = "div1">
<p>This is first division of the DOM.</p>
</div>

<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>

<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>
```

**Output for above code**



## 2. Selecting element by class(The .class Selector)

The jQuery class selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

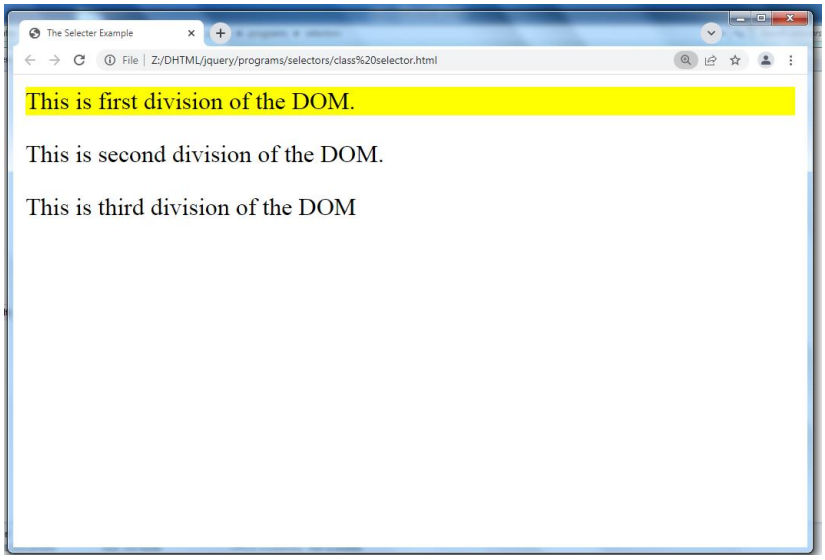
```
$(".test")
```

**Example**

```
<html>
<head>
<title>The Selecter Example</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
/* This would select second division only*/
$(".big").css("background-color", "yellow");
});
</script>
</head>
<body>
<div class = "big" id="div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
```

```
<p>This is second division of the DOM.</p>

</div>

<div class = "small" id = "div3">

<p>This is third division of the DOM</p>

</div>

</body>

</html>
```

**Output for above code**




### 3. Selecting element by type

The element type selector selects all the elements that have a tag name of it.

**Syntax**

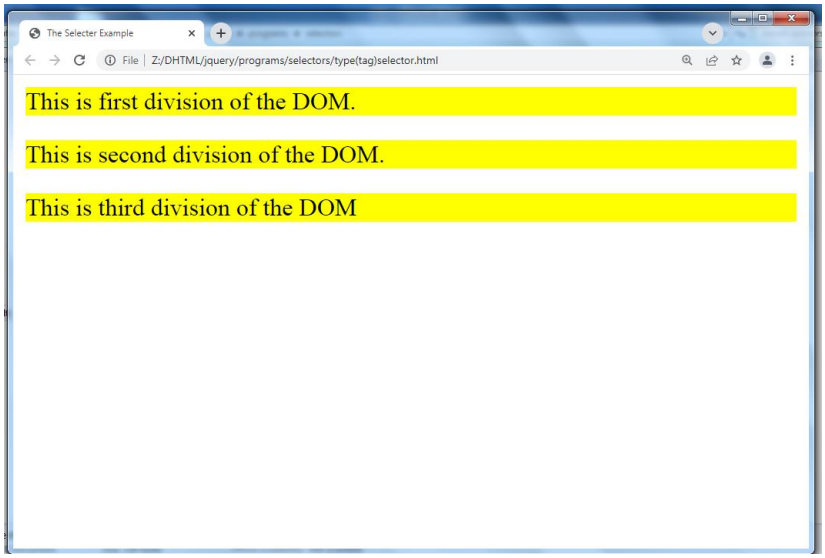
Here is the simple syntax to use this selector −

```
$('tagname')
```

**Example:** $('div'), $('a')

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
</head>
<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me to hide paragraphs</button>
</body>
</html>
```

**Output for above code**




### 4. Selecting element by hierarchy

This hierarchy Elements selector selects the combined results of all the specified selectors like div, p etc.

You can specify any number of selectors to combine into a single result. Here order of the DOM elements in the jQuery object isn't necessarily identical.

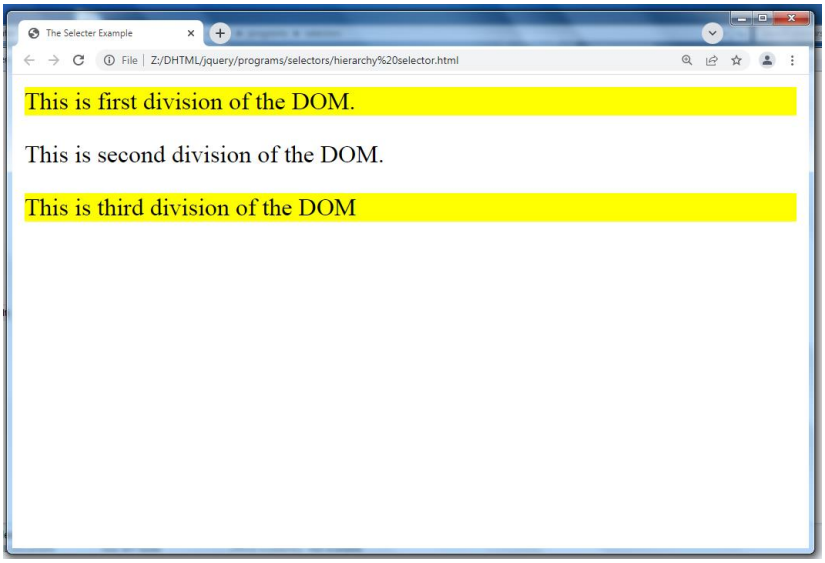
**Example**

- $('div, p') − selects all the elements matched by div or p.

```
<html>
<head>
<title>The Selecter Example</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
  $(document).ready(function() {
$(".big, #div3").css("background-color", "yellow");
});
```

```
</script>
</head>
<body>
<div class = "big" id = "div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>
<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>
```

**Output for above code**



### 5. Selecting element by Attribute

The attribute selector selects the attribute of all the specified selectors like div, p etc.

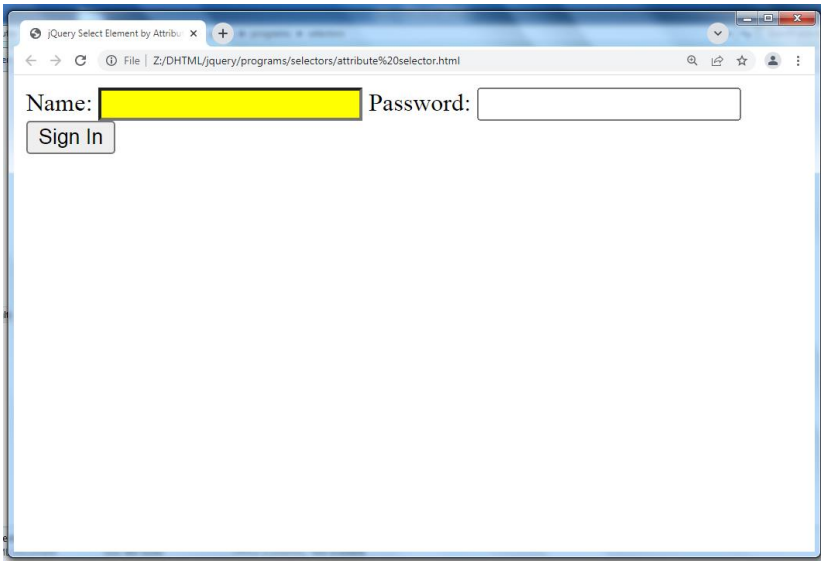
**Example**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery Select Element by Attribute</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    // Highlight paragraph elements
    $(":text").css("background-color", "yellow");
});
</script>
</head>
<body>
<form>
<label>Name:
<input type="text"></label>
<label>Password:
<input type="password"></label>
<input type="submit" value="Sign In">
</form>
```

</body>

</html>

**Output for above code**




**Basic Selectors**

| Selector | Example | Description |
|---|---|---|
| * | $("*") | Selects all elements. |
| *#id* | $("#foo") | Selects all elements with the id="foo". |
| *.class* | $(".bar") | Selects all elements with the class="bar". |
| element | $("p") | Selects all <p> elements. |
| selector1, selector2, selectorN | $("h1,p.bar,span") | Selects all <h1> and , but only that <p>elements that has the class="bar". |

**Basic Filter Selectors**

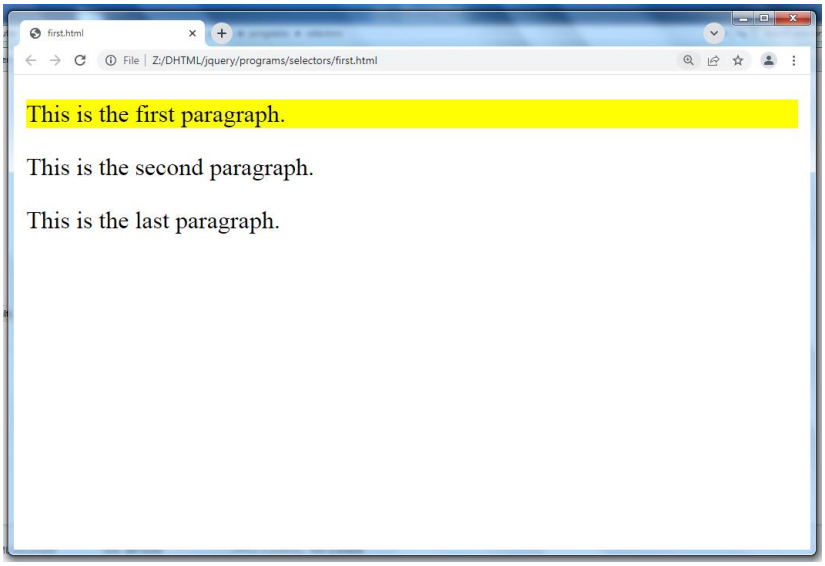
| Selector | Example | Description |
|---|---|---|
| :first | $("p:first") | Selects the first <p> element. |
| :last | $("p:last") | Selects the last <p> element. |
| :even | $("tr:even") | Selects all even <tr> elements, zero-indexed. |
| :odd | $("tr:odd") | Selects all odd <tr> elements, zero-indexed. |
| :eq() | $("tr:eq(1)") | Select the 2nd <tr> element within the matched set, zero-based index. |
| :not() | $("p:not(:empty)") | Select all <p> elements that are not empty. |
| :lt() | $("ul li:lt(3)") | Select all <li> elements at an index less than three within the matched set (i.e. selects 1st, 2nd, 3rd list elements), zero-based index. |
| :gt() | $("ul li:gt(3)") | Select all <li> elements at an index greater than three within the matched set (i.e. selects 4th, 5th, ... list elements), zero-based index. |
| :header | $(":header") | Selects all elements that are headers,<br>like <h1>, <h2>,<h3> and so on. |
| :lang() | $(":lang(en)") | Selects all elements that have a language value "en"<br>i.e.lang="en", lang="en-us" etc. |

**Example of First selector**

```
<!DOCTYPE html>
<html>
<head>
```

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("p:first").css("background-color", "yellow");
});
</script>
</head>
<body>

<p>This is the first paragraph.</p>
<p>This is the second paragraph.</p>
<p>This is the last paragraph.</p>

</body>
</html>
```

**Output for above code**




**Example of even selector**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("tr:even").css("background-color", "yellow");
```

```
});
</script>
</head>
<body>

<h1>Welcome to My Web Page</h1>

<table border="1">
<tr>
<th>Company</th>
<th>Country</th>
</tr>
<tr>
<td>Alfreds Futterkiste</td>
<td>Germany</td>
</tr>
<tr>
<td>Berglunds snabbköp</td>
<td>Sweden</td>
</tr>
<tr>
<td>Centro comercial Moctezuma</td>
<td>Mexico</td>
</tr>
<tr>
<td>Ernst Handel</td>
<td>Austria</td>
</tr>
<tr>
<td>Island Trading</td>
<td>UK</td>
</tr>
</table>

</body>
</html>
```

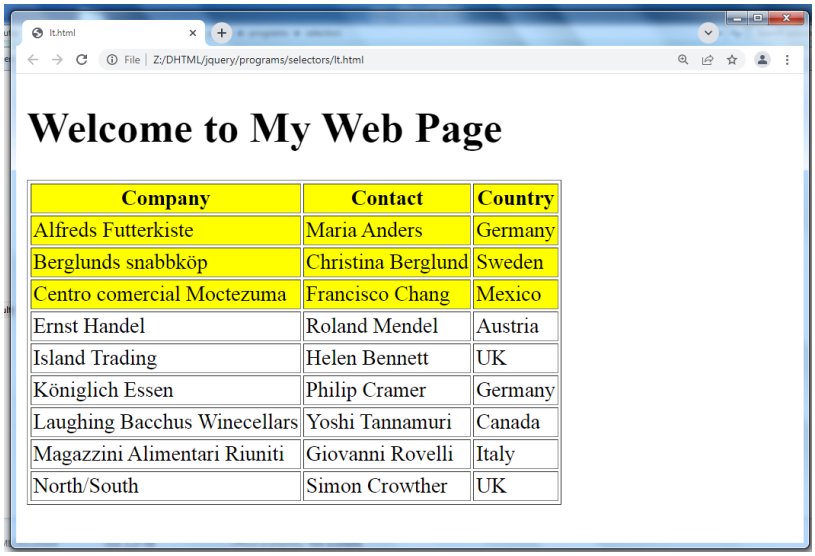
**Output for above code**

**Example of lt selector**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("tr:lt(4)").css("background-color", "yellow");
});
</script>
</head>
<body>

<h1>Welcome to My Web Page</h1>

<table border="1">
<tr>
<th>Company</th>
<th>Contact</th>
<th>Country</th>
</tr>
<tr>
<td>Alfreds Futterkiste</td>
<td>Maria Anders</td>
<td>Germany</td>
</tr>
<tr>
<td>Berglunds snabbköp</td>
```

```
<td>Christina Berglund</td>
<td>Sweden</td>
</tr>
<tr>
<td>Centro comercial Moctezuma</td>
<td>Francisco Chang</td>
<td>Mexico</td>
</tr>
<tr>
<td>Ernst Handel</td>
<td>Roland Mendel</td>
<td>Austria</td>
</tr>
<tr>
<td>Island Trading</td>
<td>Helen Bennett</td>
<td>UK</td>
</tr>
<tr>
<td>Königlich Essen</td>
<td>Philip Cramer</td>
<td>Germany</td>
</tr>
<tr>
<td>Laughing Bacchus Winecellars</td>
<td>Yoshi Tannamuri</td>
<td>Canada</td>
</tr>
<tr>
<td>Magazzini Alimentari Riuniti</td>
<td>Giovanni Rovelli</td>
<td>Italy</td>
</tr>
<tr>
<td>North/South</td>
<td>Simon Crowther</td>
<td>UK</td>
</tr>
</table>

</body>
</html>
```

**Output for above code**

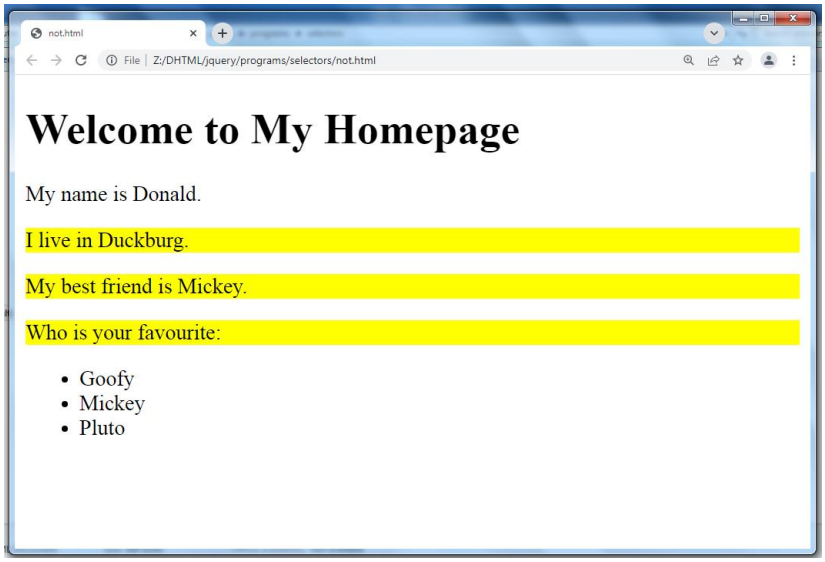



**Example of not selector**

```
<!DOCTYPE html>
<html>
<head>
<script

src="https://code.jquery.com/jquery-

3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("p:not(.intro)").css("background-

color", "yellow");
});
</script>
</head>
<body>

<h1>Welcome to My Homepage</h1>

<p class="intro">My name is Donald.</p>
<p>I live in Duckburg.</p>
<p>My best friend is Mickey.</p>

<p>Who is your favourite:</p>
```

```
<ul id="choose">
<li>Goofy</li>
<li>Mickey</li>
<li>Pluto</li>
</ul>

</body>
</html>
```

**Output for above code**




**Child Filter Selectors**

| Selector | Example | Description |
|---|---|---|
| :first-child | $("p:first-child") | Selects all <p> elements that are the first child of their parent. |
| :last-child | $("p:last-child") | Selects all <p> elements that are the last child of their parent. |
| :nth-child(n) | $("p:nth-child(3)") | Selects all <p> elements that are the 3rd child of their parent. |

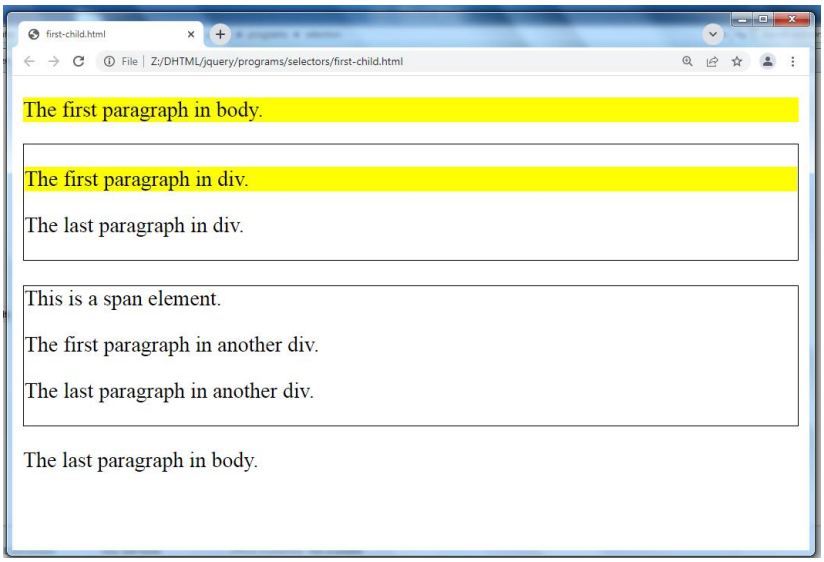
| | | |
|---|---|---|
| :only-child | $("p:only-child") | Selects all <p> elements that are the only child of their parent. |
| :first-of-type | $("p:first-of-type") | Selects all <p> elements that are the first<p> element of their parent. |
| :last-of-type | $("p:last-of-type") | Selects all <p> elements that are the last<p> element of their parent. |
| :only-of-type | $("p:only-of-type") | Selects all <p> elements that have no siblings with the same element name. |
| :nth-last-child(n) | $("p:nth-last-child(3)") | Selects all <p> elements that are the 3rd-child of their parent, counting from the last element to the first. |
| :nth-of-type(n) | $("p:nth-of-type(2)") | Selects all <p> elements that are the 2nd<p> element of their parent |

**Example of First-child selector**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("p:first-child").css("background-color", "yellow");
```

```
});
</script>
</head>
<body>

<p>The first paragraph in body.</p>

<div style="border:1px solid;">
<p>The first paragraph in div.</p>
<p>The last paragraph in div.</p>
</div><br>

<div style="border:1px solid;">
<span>This is a span element.</span>
<p>The first paragraph in another div.</p>
<p>The last paragraph in another div.</p>
</div>

<p>The last paragraph in body.</p>

</body>
</html>
```

**Output for above code**

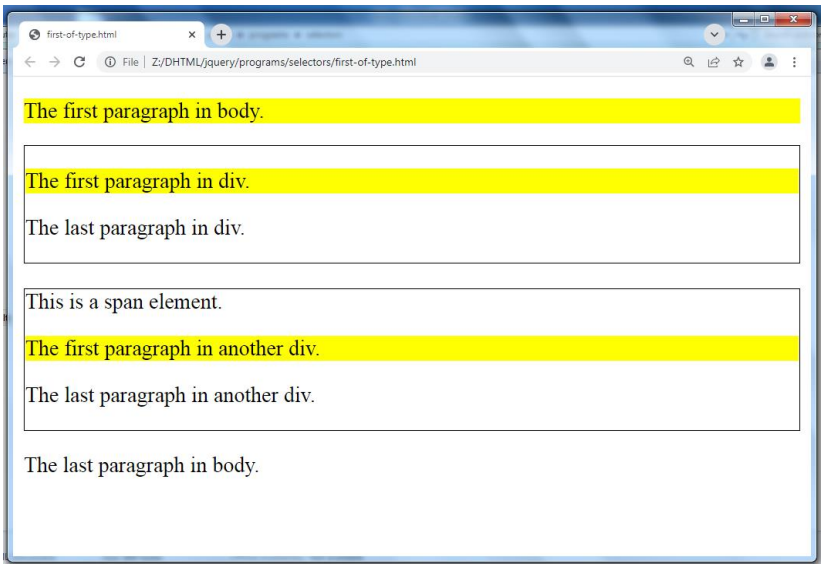



**Example of First-of-type selector**

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.3.1.js"></script>
<script>
$(document).ready(function(){
  $("p:first-of-type").css("background-color", "yellow");
});
</script>
</head>
<body>

<p>The first paragraph in body.</p>
```

```
<div style="border:1px solid;">
<p>The first paragraph in div.</p>
<p>The last paragraph in div.</p>
</div><br>

<div style="border:1px solid;">
<span>This is a span element.</span>
<p>The first paragraph in another div.</p>
<p>The last paragraph in another div.</p>
</div>
<p>The last paragraph in body.</p>
</body>
</html>
```

**Output for above code**

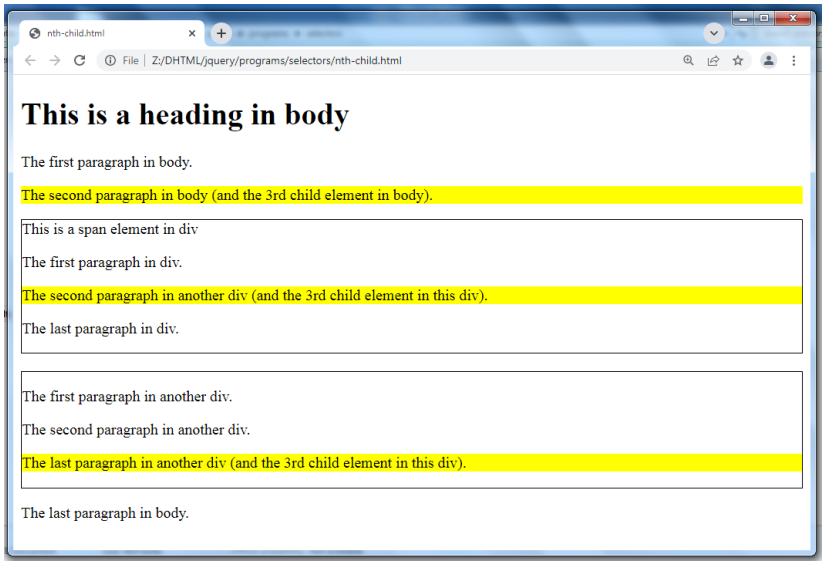


**Example of N$^{th}$-child Selector**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("p:nth-child(3)").css("background-color", "yellow");
});
</script>
</head>
<body>

<h1>This is a heading in body</h1>

<p>The first paragraph in body.</p>
<p>The second paragraph in body (and the 3rd child element in body).</p>

<div style="border:1px solid;">
<span>This is a span element in div</span>
<p>The first paragraph in div.</p>
<p>The second paragraph in another div (and the 3rd child element in this div).</p>
<p>The last paragraph in div.</p>
</div><br>
```

```
<div style="border:1px solid;">
<p>The first paragraph in another div.</p>
<p>The second paragraph in another div.</p>
<p>The last paragraph in another div (and the 3rd child element in this div).</p>
</div>
<p>The last paragraph in body.</p>
</body>
</html>
```

**Output for above code**




**Content Filter Selectors**

| Selector | Example | Description |
|---|---|---|
| :contains() | $('p:contains("Hello")') | Selects all <p> elements that contains the text "Hello". |
| :empty | $("td:empty") | Selects all <td> elements that are empty i.e that have no children including text. |

| | | |
|---|---|---|
| :has() | $("p:has(img)") | Selects all <p> elements which contain at least one<img> element. |
| :parent | $(":parent") | Select all elements that have at least one child node either an element or text. |

**Form Selectors**

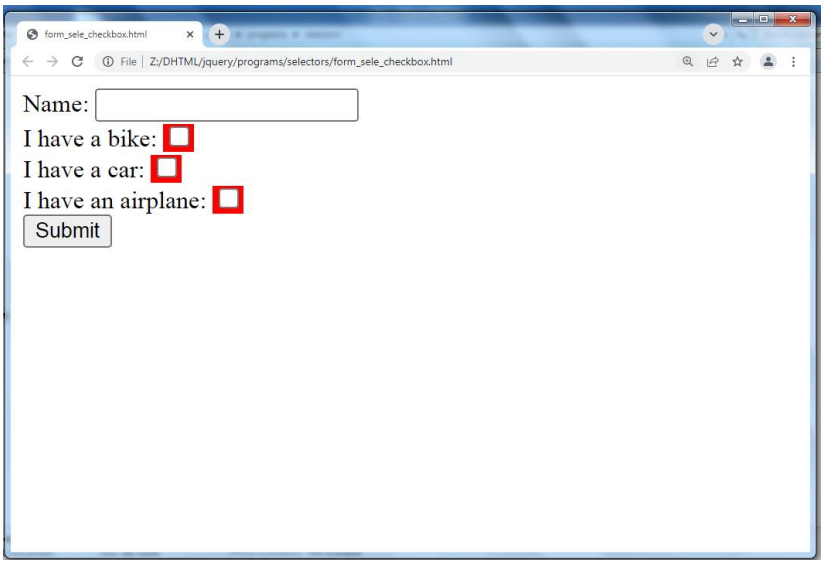
| Selector | Example | Description |
|---|---|---|
| :input | $(":input") | Selects all input, textarea, select and button elements (basically selects all form controls). |
| :text | $(":text") | Selects all input elements with type="text". |
| :password | $(":password") | Selects all input elements with type="password". |
| :radio | $(":radio") | Selects all input elements with type="radio". |
| :checkbox | $(":checkbox") | Selects all input elements with type="checkbox". |
| :button | $(":button") | Selects all input and button elements withtype="button". |
| :submit | $(":submit") | Selects all input and button elements withtype="submit". |
| :reset | $(":reset") | Selects all input and button elements withtype="reset". |
| :image | $(":image") | Selects all input elements with type="image". |
| :file | $(":file") | Selects all input elements with type="file". |
| :selected | $(":selected") | Selects all elements that are selected, only works for<option> elements. |
| :checked | $(":checked") | Selects all elements that are checked or selected, works for checkboxes, radio buttons, and select elements. |

| | | |
|---|---|---|
| :focus | $(":focus") | Selects element if it is currently focused. |

**Examle of input type=”checkbox”**

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.3.1.js"></script>
<script>
$(document).ready(function(){
  $(":checkbox").wrap("<span style='background-color:red'>");
});
</script>
</head>
<body>

<form action="">
  Name: <input type="text" name="user"><br>
  I have a bike: <input type="checkbox" name="vehicle" value="Bike"><br>
  I have a car: <input type="checkbox" name="vehicle" value="Car"><br>
  I have an airplane: <input type="checkbox" name="vehicle" value="Airplane"><br>
<input type="submit">
</form>
</body>
</html>
```

**Output for above code**

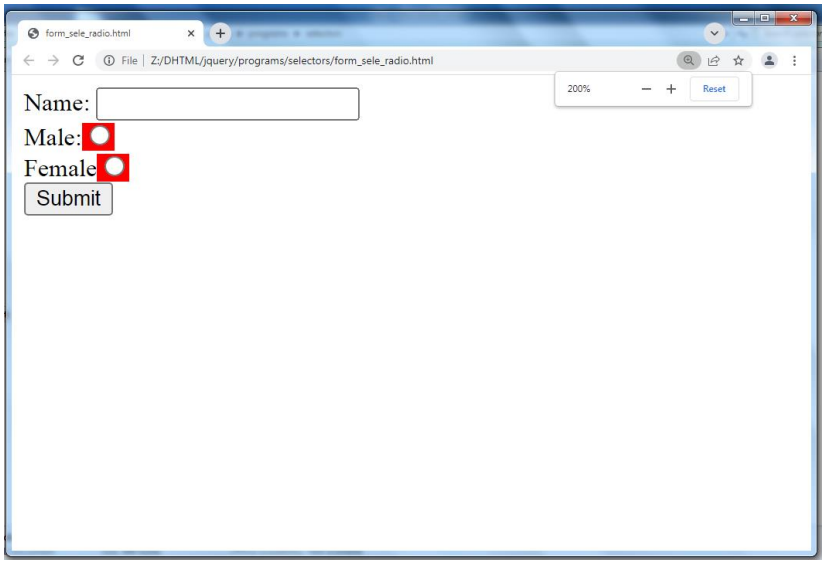



**Example of input type=”radio”**

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.3.1.js"></script>
<script>
$(document).ready(function(){
$(":radio").wrap("<span style='background-color:red'>");
});
</script>
</head>
<body>
```

```
<form action="">
Name: <input type="text" name="user"><br>
Male:<input type="radio" name="sex" value="m"><br>
Female<input type="radio" name="sex" value="f"><br>
<input type="submit">
</form>
</body>
</html>
```

**Output for the above code**

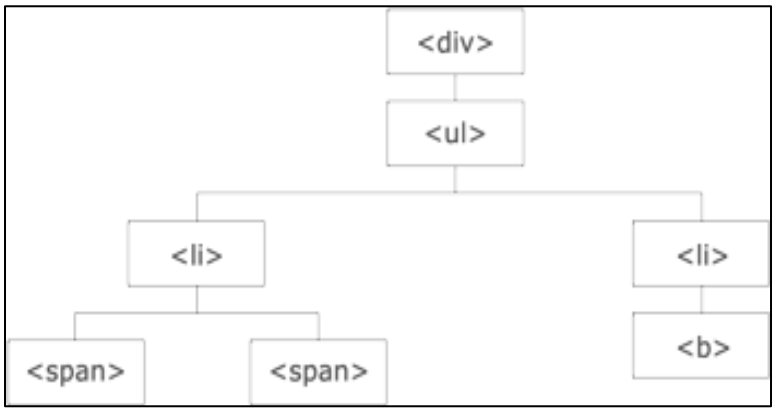



## 14.2 JQuery Traversing

Query traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

The image below illustrates an HTML page as a tree (DOM tree). With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM tree.

Let consider following tree




- The  element is the **parent** of <ul>, and an **ancestor** of everything inside of it
- The <ul> element is the **parent** of both <li> elements, and a **child** of 
- The left <li> element is the **parent** of , **child** of <ul> and a**descendant** of 
- The  element is a **child** of the left <li> and a **descendant** of <ul> and 
- The two <li> elements are **siblings** (they share the same parent)
- The right <li> element is the **parent** of <b>, **child** of <ul> and a**descendant** of 
- The <b> element is a **child** of the right <li> and a **descendant** of <ul> and 

An ancestor is a parent, grandparent, great-grandparent, and so on

A descendant is a child, grandchild, great-grandchild, and so on.
Siblings share the same parent.

## Traversing the DOM

jQuery provides a variety of methods that allow us to traverse the DOM.

The largest category of traversal methods are tree-traversal.

## jQuery Traversing - Ancestors

An ancestor is a parent, grandparent, great-grandparent, and so on.

With jQuery you can traverse up the DOM tree to find ancestors of an element.

## Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

- parent()
- parents()
- parentsUntil()

## jQuery parent() Method

The parent() method returns the direct parent element of the selected element.

This method only traverse a single level up the DOM tree.

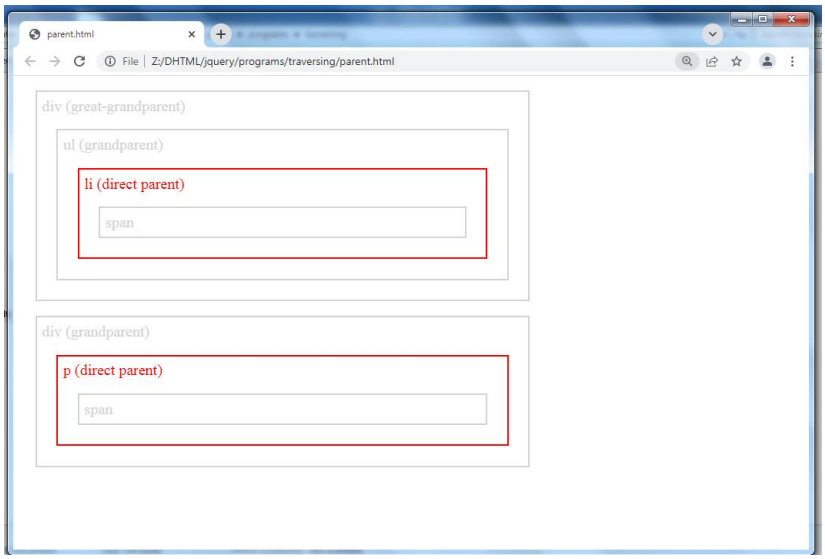
**Example**

```
<!DOCTYPE html>

<html>

<head>

<style>

.ancestors * {

   display: block;
```

```
    border: 2px solid lightgrey;
    color: lightgrey;
    padding: 5px;
    margin: 15px;
}
</style>
<script src="jquery-3.3.1.js"></script>
<script>
$(document).ready(function(){
    $("span").parent().css({"color": "red", "border": "2px solid red"});
});
</script>
</head>
<body>
<div class="ancestors">
<div style="width:500px;">div (great-grandparent)
<ul>ul (grandparent)
<li>li (direct parent)
<span>span</span>
</li>
</ul>
```

```
</div>

<div style="width:500px;">div (grandparent)

<p>p (direct parent)

<span>span</span>

</p>

</div>

</div>

</body>

</html>
```

**Output for above code**




## jQuery parents() Method

The parents() method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).

```
<!DOCTYPE html>

<html>

<head>

<script src="jquery-3.3.1.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("li").each(function(){

            alert($(this).text())

        });

    });

});

</script>

</head>

<body>

<button>Alert the value of each list item</button>

<ul>

<li>Coffee</li>

<li>Milk</li>
```

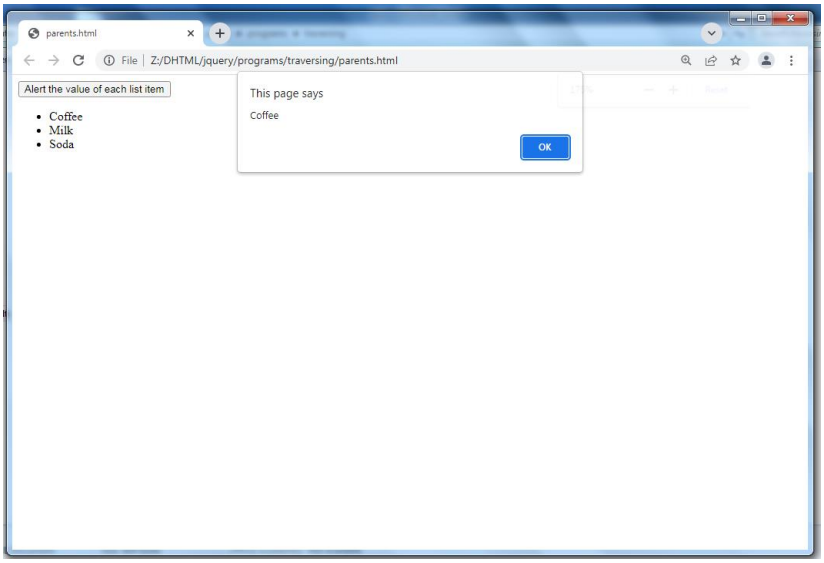
<li>Soda</li>

</ul>

</body>

</html>

**Output for above code**




## jQuery parentsUntil() Method

The parentsUntil() method returns all ancestor elements between two given arguments.

**Example**

<!DOCTYPE html>

<html>

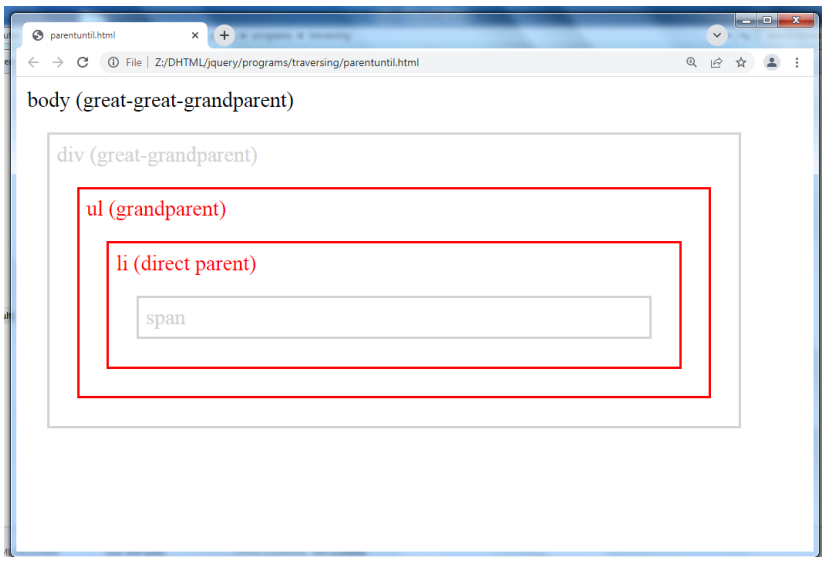
<head>

<style>

.ancestors * {

```
    display: block;
    border: 2px solid lightgrey;
    color: lightgrey;
    padding: 5px;
    margin: 15px;
}
</style>
<script src="jquery-3.3.1.js"></script>
<script>
$(document).ready(function(){
    $("span").parentsUntil("div").css({"color": "red", "border": "2px solid red"});
});
</script>
</head>

<body class="ancestors"> body (great-great-grandparent)
<div style="width:500px;">div (great-grandparent)
<ul>ul (grandparent)
<li>li (direct parent)
<span>span</span>
</li>
</ul>
</div>
</body>
</html>
```

### Output for above code




## jQuery Traversing - Descendants

A descendant is a child, grandchild, great-grandchild, and so on.

With jQuery you can traverse down the DOM tree to find descendants of an element.

### Traversing Down the DOM Tree

Two useful jQuery methods for traversing down the DOM tree are:

- children()
- find()

### jQuery children() Method

The children() method returns all direct children of the selected element.

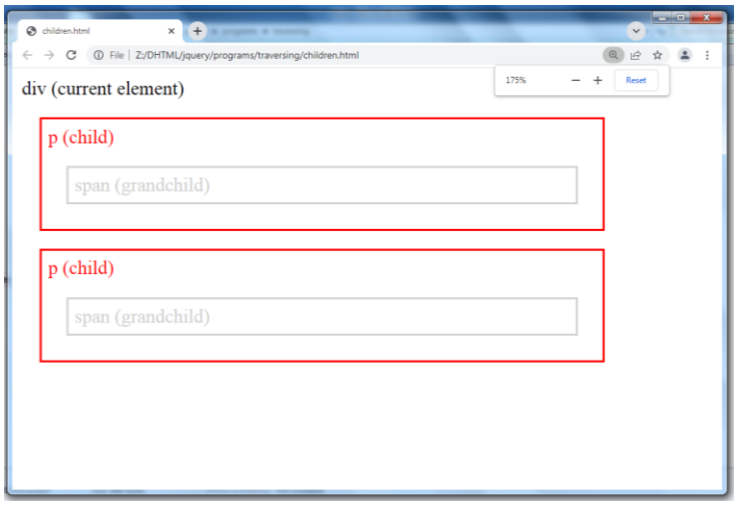
This method only traverse a single level down the DOM tree.

**Example**

```
<!DOCTYPE html>
<html>
<head>
<style>
.descendants * {
    display: block;
    border: 2px solid lightgrey;
    color: lightgrey;
    padding: 5px;
    margin: 15px;
}
</style>
<script src="jquery-3.3.1.js"></script>
<script>
$(document).ready(function(){
    $("div").children().css({"color": "red", "border": "2px solid red"});
});
</script>
</head>
```

```
<body>

<div class="descendants" style="width:500px;">div (current element)

<p>p (child)

<span>span (grandchild)</span>

</p>

<p>p (child)

<span>span (grandchild)</span>

</p>

</div>

</body>

</html>
```

**Output for above code**




## jQuery find() Method

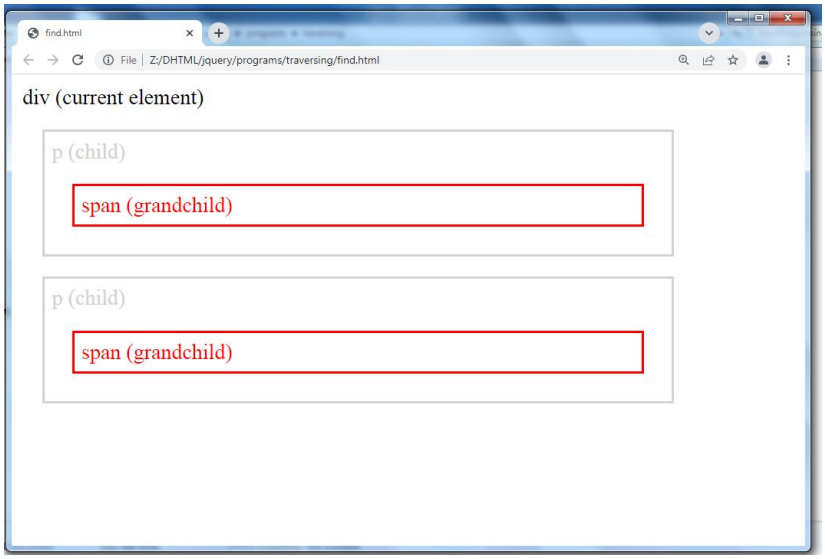
The find() method returns descendant elements of the selected element, all the way down to the last descendant.

**Example**

```
<!DOCTYPE html>
<html>
<head>
<style>
.descendants * {
    display: block;
    border: 2px solid lightgrey;
    color: lightgrey;
    padding: 5px;
    margin: 15px;
}
</style>
<script src="jquery-3.3.1.js"></script>
<script>
$(document).ready(function(){
    $("div").find("span").css({"color": "red", "border": "2px solid red"});
});
</script>
</head>
<body>
<div class="descendants" style="width:500px;">div (current element)
<p>p (child)
<span>span (grandchild)</span>
```

```
</p>
<p>p (child)
<span>span (grandchild)</span>
</p>
</div>
</body>
</html>
```

**Output for above code**




**jQuery siblings() Method**

The jQuery siblings() method is used to get the sibling elements of the selected element.

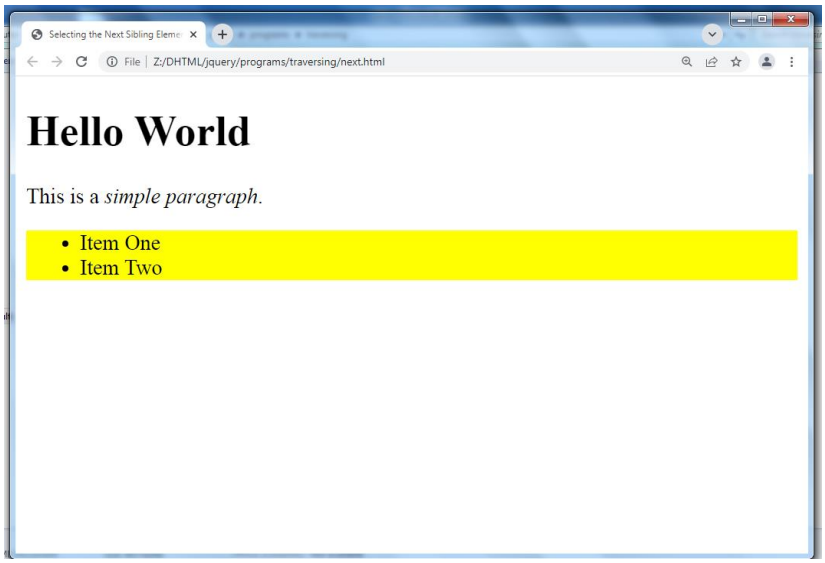
**1. next() Method**

The jQuery next() method is used to get the immediately following sibling i.e. the next sibling element of the selected element.

```
<!DOCTYPE html>
```

```
<html>
<head>

<title>Selecting the Next Sibling Element in jQuery</title>
<style type="text/css">
    .highlight{
        background: yellow;
    }
</style>
<script src="jquery-3.3.1.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("p").next().addClass("highlight");
});
</script>
</head>
<body>
<div>
<h1>Hello World</h1>
<p>This is a <em>simple paragraph</em>.</p>
<ul>
<li>Item One</li>
<li>Item Two</li>
</ul>
</div>
</body>
</html>
```

**Output for above code**




## 2. nextAll() Method

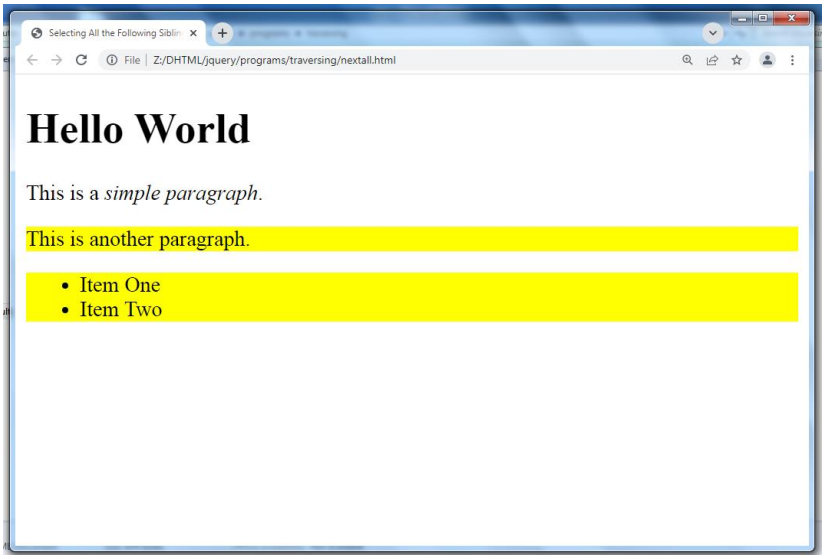
The jQuery nextAll() method is used to get all following siblings of the selected element.

**Example**

```
<!DOCTYPE html>
<html>
<head>

<title>Selecting All the Following Sibling Elements in jQuery</title>
<style type="text/css">
    .highlight{
        background: yellow;
    }
</style>
<script src="jquery-3.3.1.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("p").nextAll().addClass("highlight");
});
</script>
</head>
<body>
```

```
<div >
<h1>Hello World</h1>
<p>This is a <em>simple paragraph</em>.</p>
<p>This is another paragraph.</p>
<ul>
<li>Item One</li>
<li>Item Two</li>
</ul>
</div>
</body>
</html>
```

**Output for above code**




### 3. nextUntil() Method

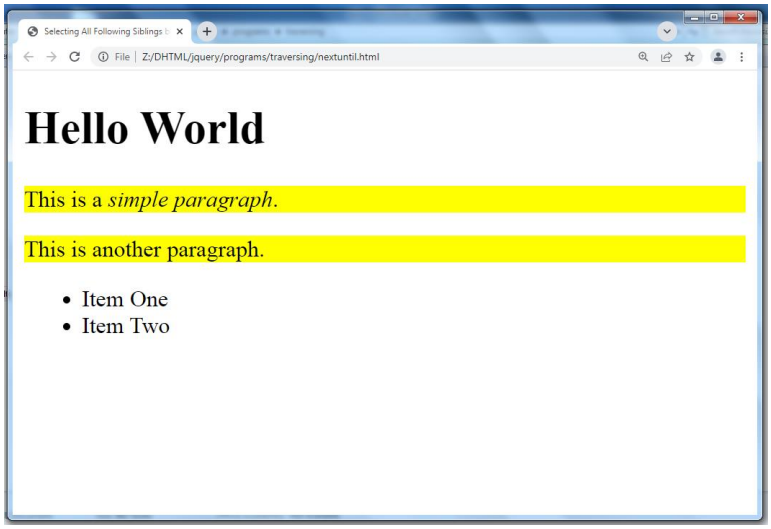
The jQuery nextUntil() method is used to get all the following siblings up to but not including the element matched by the selector. In simple words we can say it returns all the next siblings elements between two given elements in a DOM hierarchy.

**Example**

```
<!DOCTYPE html>
```

```
<html>
<head>

<title>Selecting All Following Siblings between Two Elements in jQuery</title>
<style type="text/css">
   .highlight{
      background: yellow;
   }
</style>
<script src="jquery-3.3.1.js"></script>
<script type="text/javascript">
$(document).ready(function(){
   $("h1").nextUntil("ul").addClass("highlight");
});
</script>
</head>
<body>
<div>
<h1>Hello World</h1>
<p>This is a <em>simple paragraph</em>.</p>
<p>This is another paragraph.</p>
<ul>
<li>Item One</li>
<li>Item Two</li>
</ul>
</div>
</body>
</html>
```

**Output for above code**




### 4. prev() Method

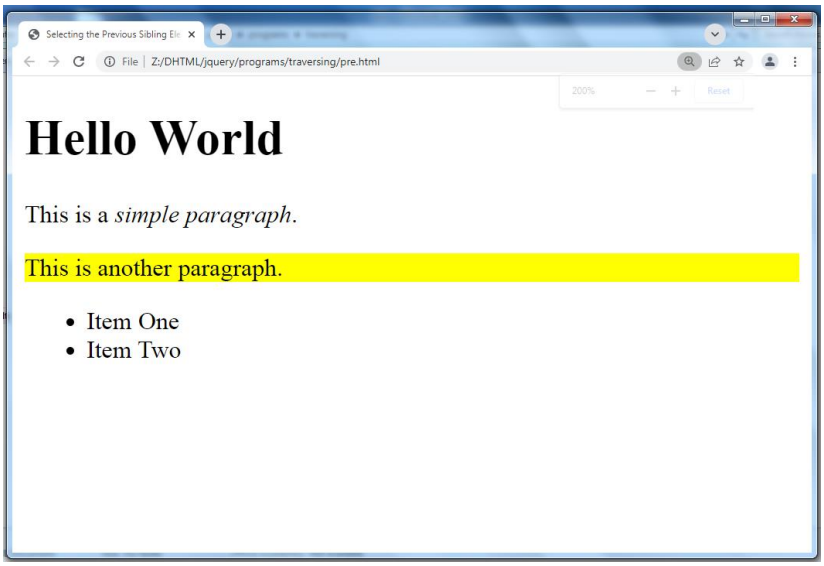
The jQuery prev() method is used to get the immediately preceding sibling i.e. the previous sibling element of the selected element.

**Example**

```
<!DOCTYPE html>
<html >
<head>
<meta charset="utf-8">
<title>Selecting the Previous Sibling Element in jQuery</title>
<style type="text/css">
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("ul").prev().addClass("highlight");
});
</script>
</head>
<body>
```

```
<div>
<h1>Hello World</h1>
<p>This is a <em>simple paragraph</em>.</p>
<p>This is another paragraph.</p>
<ul>
<li>Item One</li>
<li>Item Two</li>
</ul>
</div>
</body>
</html>
```

**Output for above code**




## 5. prevAll() Method

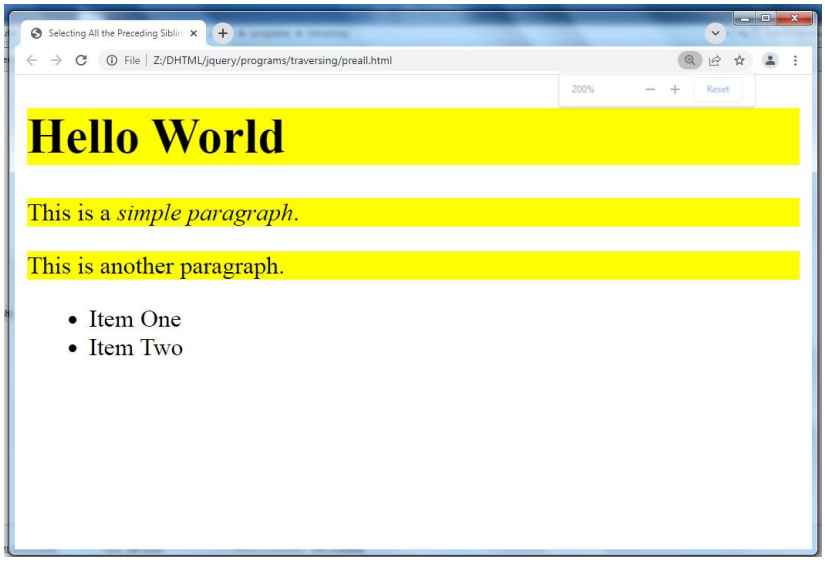
The jQuery prevAll() method is used to get all preceding siblings of the selected element.

**Example:**

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
<meta charset="utf-8">
<title>Selecting All the Preceding Sibling Elements in jQuery</title>
<style type="text/css">
   .highlight{
      background: yellow;
   }
</style>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
   $("ul").prevAll().addClass("highlight");
});
</script>
</head>
<body>
<div class="container">
<h1>Hello World</h1>
<p>This is a <em>simple paragraph</em>.</p>
<p>This is another paragraph.</p>
<ul>
<li>Item One</li>
<li>Item Two</li>
</ul>
</div>
</body>
</html>
```

**Output for above code**



### 6. prevUntil() Method

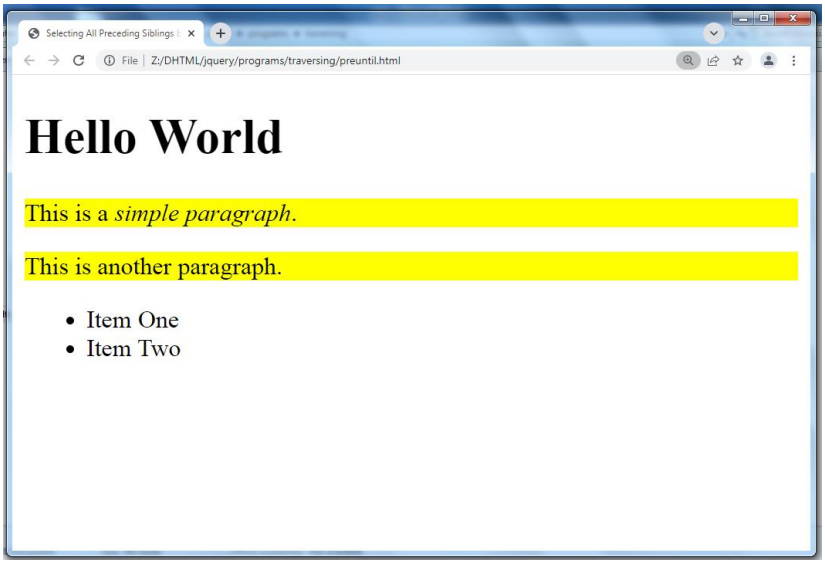
The jQuery prevUntil() method is used to get all the preceding siblings up to but not including the element matched by the selector. In simple words we can say it returns all the previous siblings elements between two given elements in a DOM hierarchy.

**Example**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Selecting All Preceding Siblings between Two Elements in jQuery</title>
<style type="text/css">
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("ul").prevUntil("h1").addClass("highlight");
});
</script>
</head>
<body>
<div class="container">
<h1>Hello World</h1>
```

```
<p>This is a <em>simple paragraph</em>.</p>

<p>This is another paragraph.</p>

<ul>

<li>Item One</li>

<li>Item Two</li>

</ul>

</div>

</body>

</html>
```

**Output for above code**




## 14.3 JQuery Attributes

jQuery gives us the means to easily manipulate an element's attributes and gives us access to the element so that we can also change its properties.

### Get Attribute Value

The **attr()** method can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.

### Applying Styles

The **addClass( classes )** method can be used to apply defined style sheets onto all the matched elements. You can specify multiple classes separated by space.

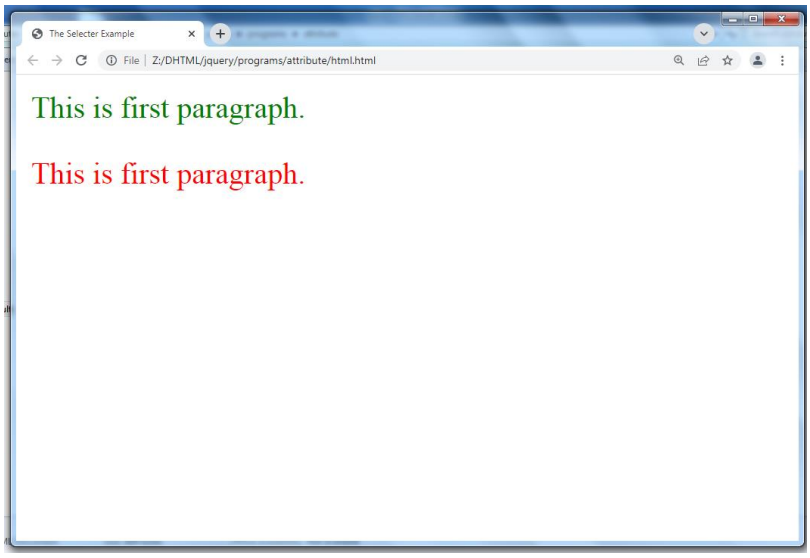
### Attribute Methods

| | |
|---|---|
| **html( )** | The html( ) method gets the html contents of the first matched element.<br>Syntax<br>selector.html( ) |
| **text( )** | Get the combined text contents of all matched elements.<br>Syntax<br>selector.text( ) |
| **val( )** | gets the input value of the first matched element. |
| **addClass()** | method adds one or more classes to the selected elements. |
| **attr()** | method to either get the value of an element's attribute or set one or more attributes for the selected element. |
| **append()** | method is used to insert content to the end of the selected elements. |
| **prepend()** | method is used to insert content to the beginning of the selected elements. |
| **before()** | method is used to insert content before the selected elements. |
| **after()** | method is used to insert content after the selected elements. |
| **empty()** | method removes all child elements as well as other descendant elements and the text content within the selected elements from the DOM. |
| **remove()** | method removes the selected elements from the DOM as well as everything inside it. |

**Example HTML Method**

```
<html>
<head>
<title>The Selecter Example</title>
<script src="jquery-3.3.1.js"></script>

<script type = "text/javascript" language="javascript">
        $(document).ready(function() {
           var content = $("p").html();
           $("#pid2").html( content );
        });
</script>
   <style>
        .red { color:red; }
        .green { color:green; }
</style>
</head>
   <body>
<p class = "green" id = "pid1">This is first paragraph.</p>
<p class = "red" id = "pid2">This is second paragraph.</p>
</body>
</html>
```

**Output for above code**

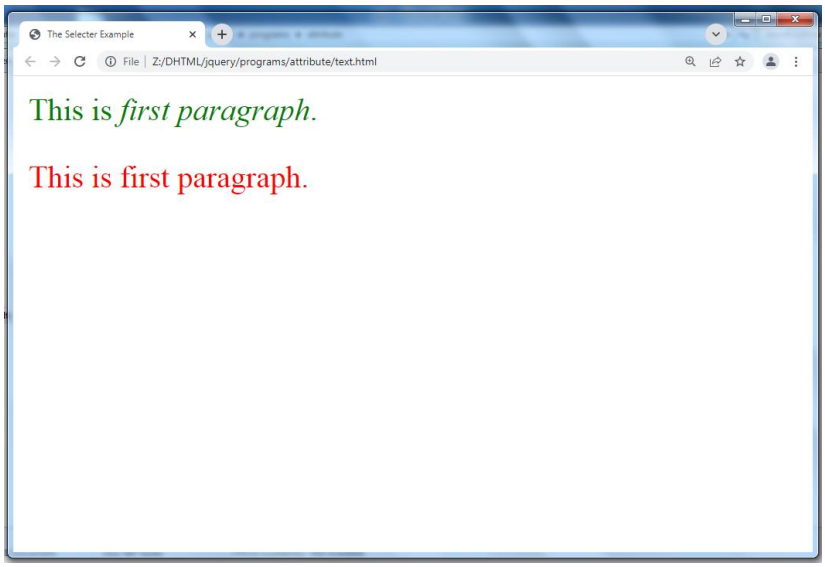



**Example of Text Method**

```
<html>
<head>
<title>The jQuery Example</title>
<script type = "text/javascript"
        src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>

<script type = "text/javascript" language = "javascript">
        $(document).ready(function() {
           var title = $("em").attr("title");
           $("#divid").text(title);
        });
</script>
</head>
```

```
<body>
<div>
<em title = "Bold and Brave">This is first paragraph.</em>
<p id = "myid">This is second paragraph.</p>
<div id = "divid"></div>
</div>
</body>
</html>
```

**Output for above code**



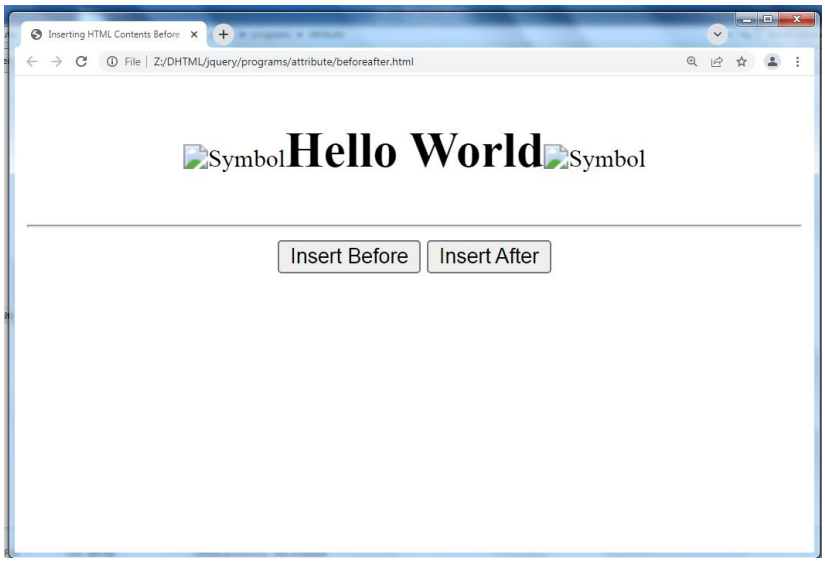

**Example of Before-after Method**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Inserting HTML Contents Before or After the Elements in jQuery</title>
```

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){

    // Add content after heading on button click
    $("button.insert-after").click(function(){
        $("h1").after('<img src="Z:\DHTML\jquery\programs\attribute\arrow.png" alt="Symbol">');
    });

    // Add content before heading on button click
    $("button.insert-before").click(function(){
        $("h1").before('<img src="Z:\DHTML\jquery\programs\attribute\arrow2.png" alt="Symbol">');
    });
});
</script>
<style type="text/css">
    h1{
        display: inline-block; /* To place marker image and heading in one line */
    }
    body{
        text-align: center;
    }
</style>
</head>
```

```
<body>
<h1>Hello World</h1>
<hr>
<button type="button" class="insert-before">Insert Before</button>
<button type="button" class="insert-after">Insert After</button>
</body>
</html>
```

**Output for above code**

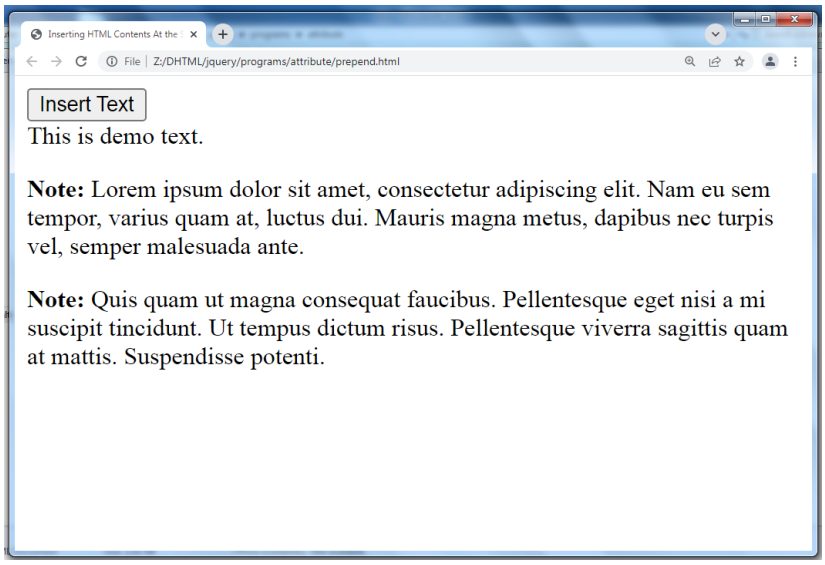



**Example of Prepend Method**

```
<!DOCTYPE html>
<html lang="en">
<head>

<title>Inserting HTML Contents At the Start of the Elements in jQuery</title>
```

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    // Prepend all paragraphs on document ready
    $("p").prepend("<strong>Note:</strong> ");

    // Prepend a div container on button click
    $("button").click(function(){
       $("#container").prepend("This is demo text.");
    });
});
</script>
</head>
<body>
<button type="button">Insert Text</button>
<div id="container">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam eu sem tempor, varius quam at, luctus dui. Mauris magna metus, dapibus nec turpis vel, semper malesuada ante.</p>
<p>Quis quam ut magna consequat faucibus. Pellentesque eget nisi a mi suscipit tincidunt. Ut tempus dictum risus. Pellentesque viverra sagittis quam at mattis. Suspendisse potenti.</p>
</div>
</body>
</html>
```

**Output for above code**

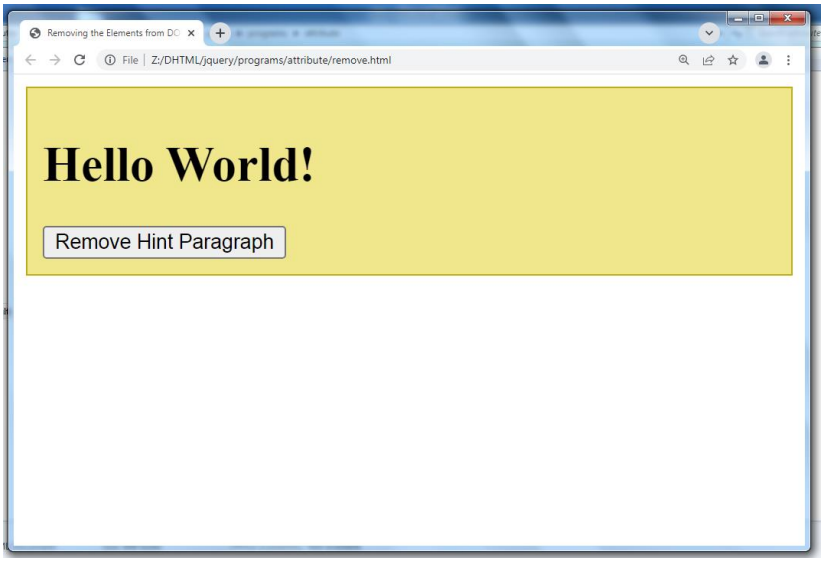



**Example of Remove Method**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Removing the Elements from DOM in jQuery</title>
<style type="text/css">
.container{
    padding: 10px;
    background: #f0e68C;
    border: 1px solid #bead18;
}
</style>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
```

```
<script type="text/javascript">
$(document).ready(function(){
    // Removes paragraphs with class "hint" from DOM on button click
    $("button").click(function(){
       $("p.hint").remove();
    });
});
</script>
</head>
<body>
<div class="container">
<h1>Hello World!</h1>
<p class="hint"><strong>Note:</strong> If you click the following button it will remove this paragraph.</p>
<button type="button">Remove Hint Paragraph</button>
</div>
</body>
</html>
```

**Output for above code**




## 14.4 JQuery Effects

**Show () or show (nanoseconds) and hide () or hide (nanoseconds) Methods**

You can show and hide HTML elements using the jQuery show() and hide() methods.

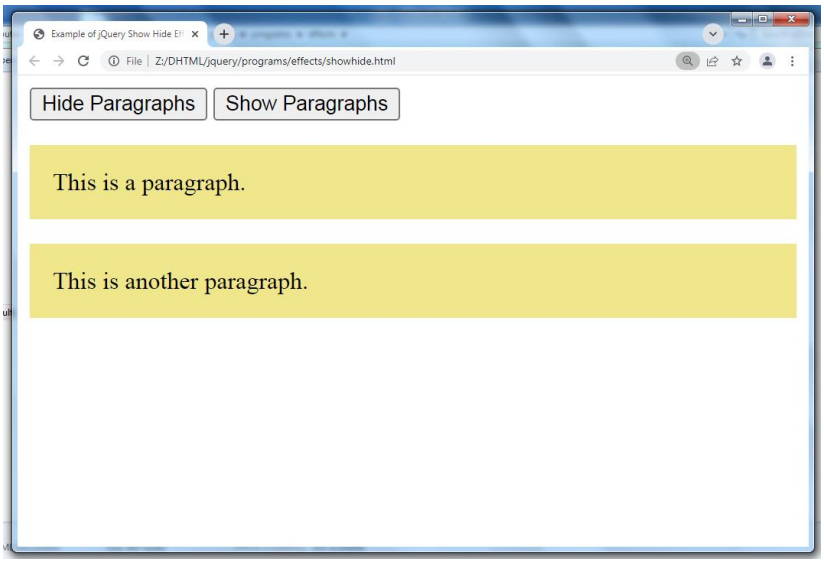
The hide() method simply sets the inline style display:none for the selected elements. Conversely, the show() method restores the display properties

**Example**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Show Hide Effects</title>
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<style type="text/css">
   p{
```

```
        padding: 15px;
        background: #F0E68C;
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    // Hide displayed paragraphs
    $(".hide-btn").click(function(){
        $("p").hide();
    });

    // Show hidden paragraphs
    $(".show-btn").click(function(){
        $("p").show();
    });
});
</script>
</head>
<body>
<button type="button" class="hide-btn">Hide Paragraphs</button>
<button type="button" class="show-btn">Show Paragraphs</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

**Output for above code**




**Example-2**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Animated Show Hide Effects</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 15px;
      background: #F0E68C;
   }
</style>
```

```
<script type="text/javascript">
$(document).ready(function(){
    // Hide displayed paragraphs with different speeds
    $(".hide-btn").click(function(){
        $("p.normal").hide();
        $("p.fast").hide("fast");
        $("p.slow").hide("slow");
        $("p.very-fast").hide(50);
        $("p.very-slow").hide(2000);
    });

    // Show hidden paragraphs with different speeds
    $(".show-btn").click(function(){
        $("p.normal").show();
        $("p.fast").show("fast");
        $("p.slow").show("slow");
        $("p.very-fast").show(50);
        $("p.very-slow").show(2000);
    });
});
</script>
</head>
<body>
<button type="button" class="hide-btn">Hide Paragraphs</button>
<button type="button" class="show-btn">Show Paragraphs</button>
```

This paragraph will show/hide with very fast speed.

This paragraph will show/hide with default speed.

This paragraph will show/hide with fast speed.

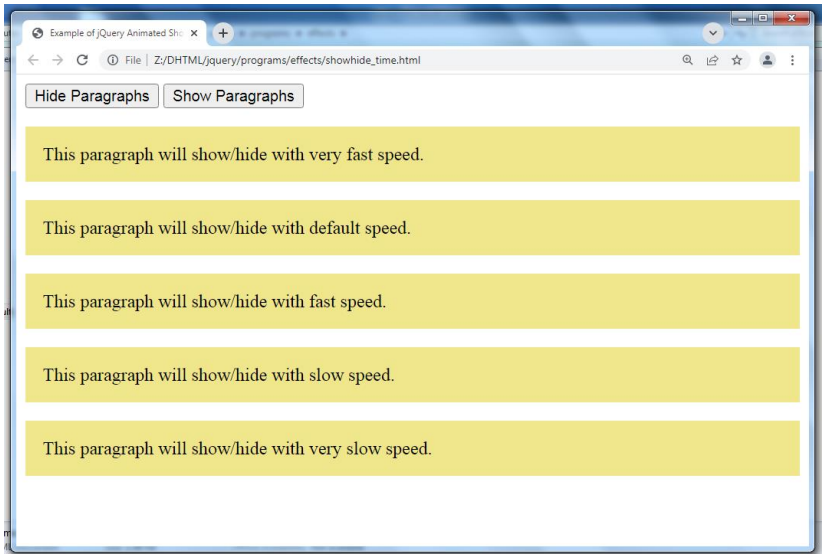
This paragraph will show/hide with slow speed.

This paragraph will show/hide with very slow speed.

</body>

</html>

**Output for above code**




**toggle() Method**

The jQuery toggle() method show or hide the elements in such a way that if the element is initially displayed, it will be hidden; if hidden, it will be displayed (i.e. toggles the visibility).

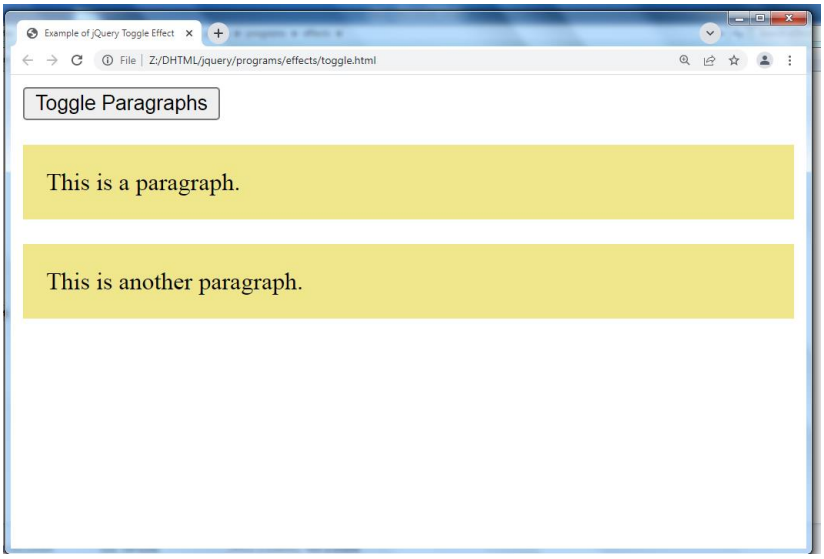
**Example**

<!DOCTYPE html>

```
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Toggle Effect</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
    p{
        padding: 15px;
        background: #F0E68C;
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    // Toggles paragraphs display
    $(".toggle-btn").click(function(){
        $("p").toggle();
    });
});
</script>
</head>
<body>
<button type="button" class="toggle-btn">Toggle Paragraphs</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
```

</html>

**Output for above code**

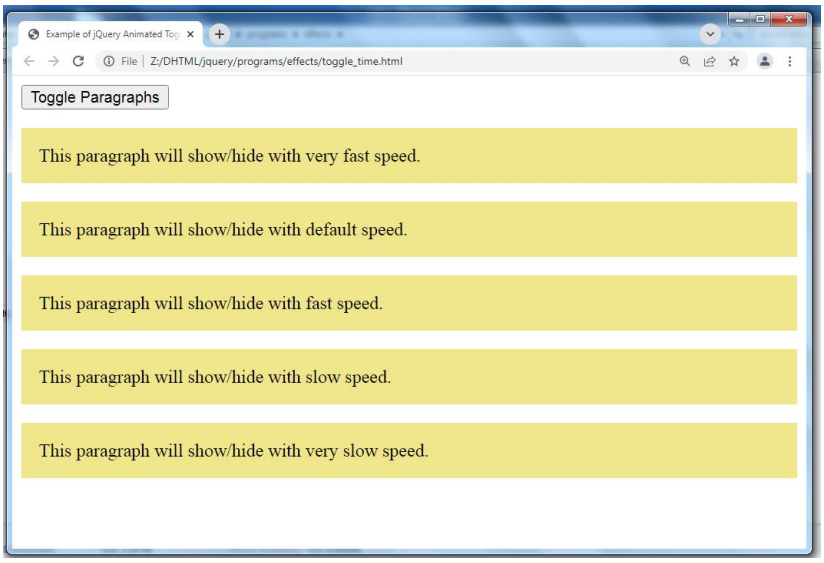



**Example-2**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Animated Toggle Effect</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 15px;
      background: #F0E68C;
   }
</style>
```

```
<script type="text/javascript">
$(document).ready(function(){
    // Toggles paragraphs with different speeds
    $(".toggle-btn").click(function(){
        $("p.normal").toggle();
        $("p.fast").toggle("fast");
        $("p.slow").toggle("slow");
        $("p.very-fast").toggle(50);
        $("p.very-slow").toggle(2000);
    });
});
</script>
</head>
<body>
<button type="button" class="toggle-btn">Toggle Paragraphs</button>
<p class="very-fast">This paragraph will show/hide with very fast speed.</p>
<p class="normal">This paragraph will show/hide with default speed.</p>
<p class="fast">This paragraph will show/hide with fast speed.</p>
<p class="slow">This paragraph will show/hide with slow speed.</p>
<p class="very-slow">This paragraph will show/hide with very slow speed.</p>
</body>
</html>
```

**Output for above code**




**fadeIn() (show)and fadeOut()(hide) Methods**

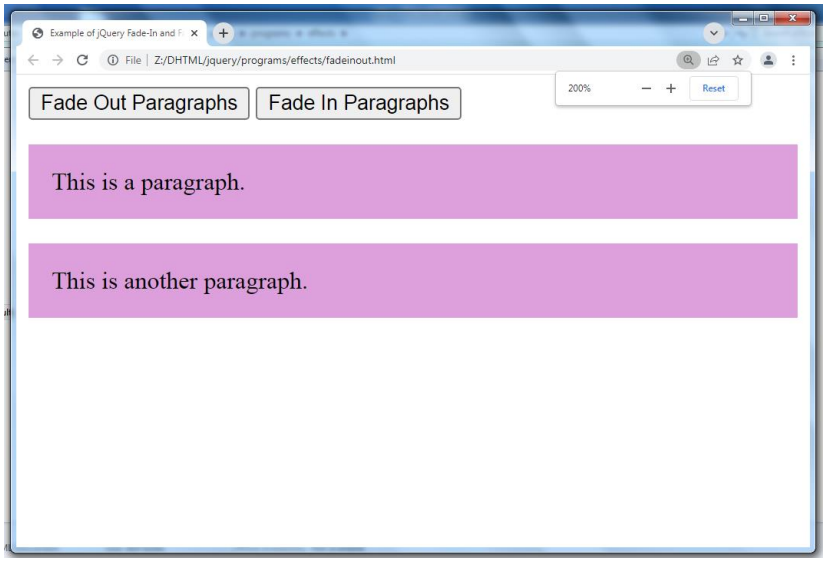
You can use the jQuery fadeIn() and fadeOut() methods to display or hide the HTML elements by gradually increasing or decreasing their opacity.

**Example**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Fade-In and Fade-Out Effects</title>
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
<style type="text/css">
   p{
      padding: 15px;
      background: #DDA0DD;
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
   // Fadeing out displayed paragraphs
   $(".out-btn").click(function(){
      $("p").fadeOut();
   });

   // Fading in hidden paragraphs
```

```
    $(".in-btn").click(function(){
        $("p").fadeIn();
    });
});
</script>
</head>
<body>
<button type="button" class="out-btn">Fade Out Paragraphs</button>
<button type="button" class="in-btn">Fade In Paragraphs</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

**Output for above code**

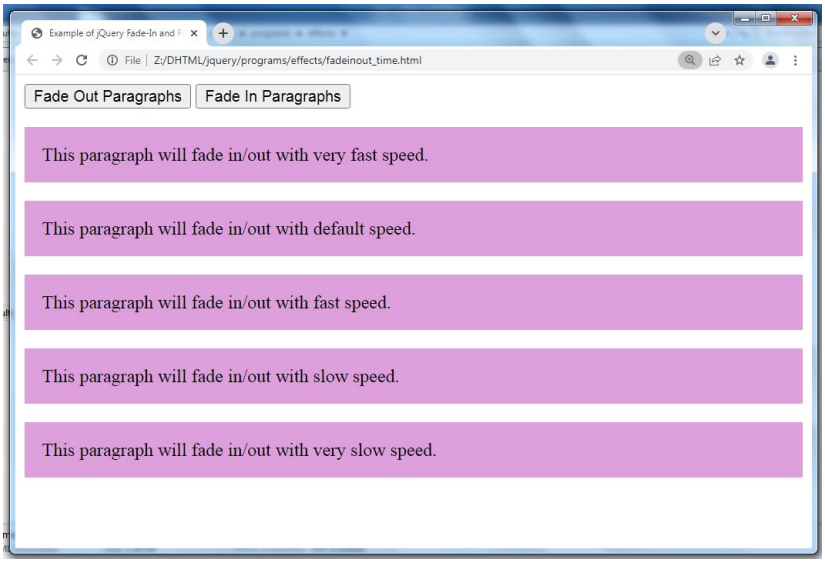



**Example-2**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Fade-In and Fade-Out Effects with Different Speeds</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
    p{
```

```
        padding: 15px;
        background: #DDA0DD;
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    // Fading out displayed paragraphs with different speeds
    $(".out-btn").click(function(){
        $("p.normal").fadeOut();
        $("p.fast").fadeOut("fast");
        $("p.slow").fadeOut("slow");
        $("p.very-fast").fadeOut(50);
        $("p.very-slow").fadeOut(2000);
    });

    // Fading in hidden paragraphs with different speeds
    $(".in-btn").click(function(){
        $("p.normal").fadeIn();
        $("p.fast").fadeIn("fast");
        $("p.slow").fadeIn("slow");
        $("p.very-fast").fadeIn(50);
        $("p.very-slow").fadeIn(2000);
    });
});
</script>
</head>
<body>
<button type="button" class="out-btn">Fade Out Paragraphs</button>
<button type="button" class="in-btn">Fade In Paragraphs</button>
<p class="very-fast">This paragraph will fade in/out with very fast speed.</p>
<p class="normal">This paragraph will fade in/out with default speed.</p>
<p class="fast">This paragraph will fade in/out with fast speed.</p>
<p class="slow">This paragraph will fade in/out with slow speed.</p>
<p class="very-slow">This paragraph will fade in/out with very slow speed.</p>
</body>
</html>
```

**Output for above code**




**fadeToggle() Method**

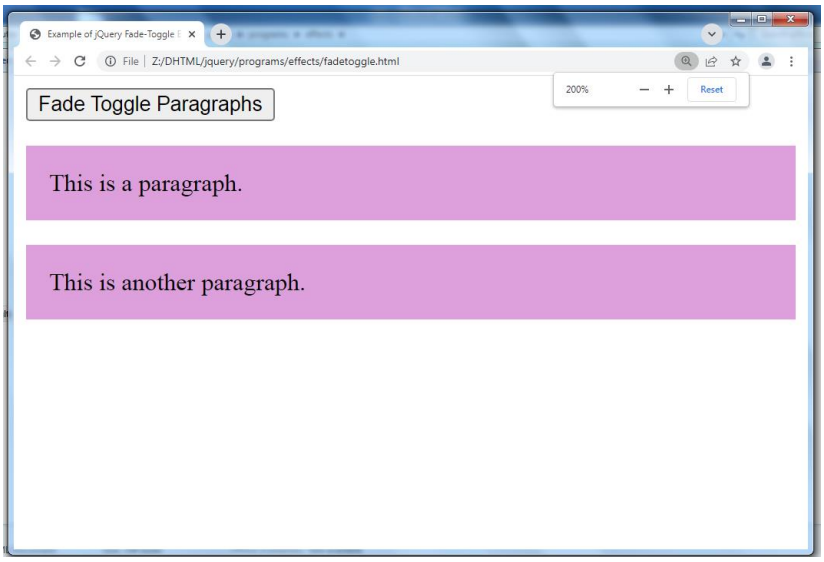
The jQuery fadeToggle() method display or hide the selected elements by animating their opacity in such a way that if the element is initially displayed, it will be fade out; if hidden, it will be fade in (i.e. toggles the fading effect).

**Example**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Fade-Toggle Effect</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 15px;
      background: #DDA0DD;
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
   // Toggles paragraphs display with fading
   $(".toggle-btn").click(function(){
```

```
        $("p").fadeToggle();
    });
});
</script>
</head>
<body>
<button type="button" class="toggle-btn">Fade Toggle Paragraphs</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

**Output for above code**




**slideUp()(close) and slideDown()(open) Methods**

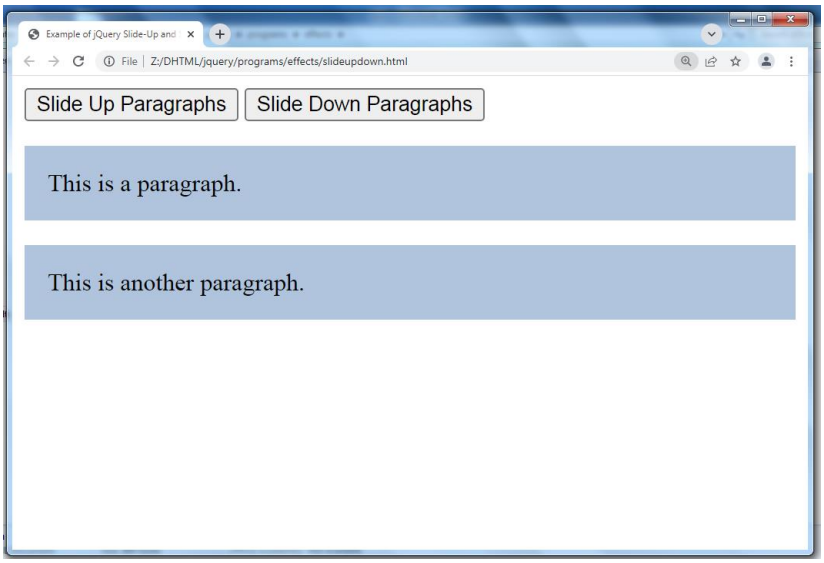
The jQuery slideUp() and slideDown() methods is used to hide or show the HTML elements by gradually decreasing or increasing their height (i.e. by sliding them up or down).

**Example**

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Slide-Up and Slide-Down Effects</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 15px;
      background: #B0C4DE;
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
   // Slide up displayed paragraphs
   $(".up-btn").click(function(){
      $("p").slideUp();
   });

   // Slide down hidden paragraphs
   $(".down-btn").click(function(){
      $("p").slideDown();
   });
});
</script>
</head>
<body>
<button type="button" class="up-btn">Slide Up Paragraphs</button>
<button type="button" class="down-btn">Slide Down Paragraphs</button>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

**Output for above code**




**Example-2**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Example of jQuery Slide-Up and Slide-Down Effects with Different Speeds</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 15px;
      background: #B0C4DE;
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
   // Sliding up displayed paragraphs with different speeds
   $(".up-btn").click(function(){
      $("p.normal").slideUp();
      $("p.fast").slideUp("fast");
      $("p.slow").slideUp("slow");
      $("p.very-fast").slideUp(50);
```

```
        $("p.very-slow").slideUp(2000);
    });

    // Sliding down hidden paragraphs with different speeds
    $(".down-btn").click(function(){
        $("p.normal").slideDown();
        $("p.fast").slideDown("fast");
        $("p.slow").slideDown("slow");
        $("p.very-fast").slideDown(50);
        $("p.very-slow").slideDown(2000);
    });
});
</script>
</head>
<body>
<button type="button" class="up-btn">Slide Up Paragraphs</button>
<button type="button" class="down-btn">Slide Down Paragraphs</button>
<p class="very-fast">This paragraph will fade in/out with very fast speed.</p>
<p class="normal">This paragraph will fade in/out with default speed.</p>
<p class="fast">This paragraph will fade in/out with fast speed.</p>
<p class="slow">This paragraph will fade in/out with slow speed.</p>
<p class="very-slow">This paragraph will fade in/out with very slow speed.</p>
</body>
</html>
```

**Output for above code**

**animate() Method**

The jQuery animate() method is used to create custom animations.
The animate() method is typically used to animate numeric CSS properties, for example, width, height, margin, padding,opacity, top, left, etc. but the non-numeric properties such as color or background-color cannot be animated using the basic jQuery functionality.

**Syntax**
The basic syntax of the jQuery animate() method can be given with:

**$(selector).animate({ properties }, duration, callback);**

The parameters of the animate() method have the following meanings:

**The required properties** parameter defines the CSS properties to be animated.

**The optional duration** parameter specifies how long the animation will run. Durations can be specified either using one of the predefined string 'slow' or 'fast', or in a number of milliseconds; higher values indicate slower animations.

**The optional callback** parameter is a function to call once the animation is complete.

**stop() Method**

The jQuery stop() method is used to stop the jQuery animations or effects currently running on the selected elements before it completes.

**syntax**
**$(selector).stop(stopAll, goToEnd);**

The parameters in the above syntax have the following meanings:

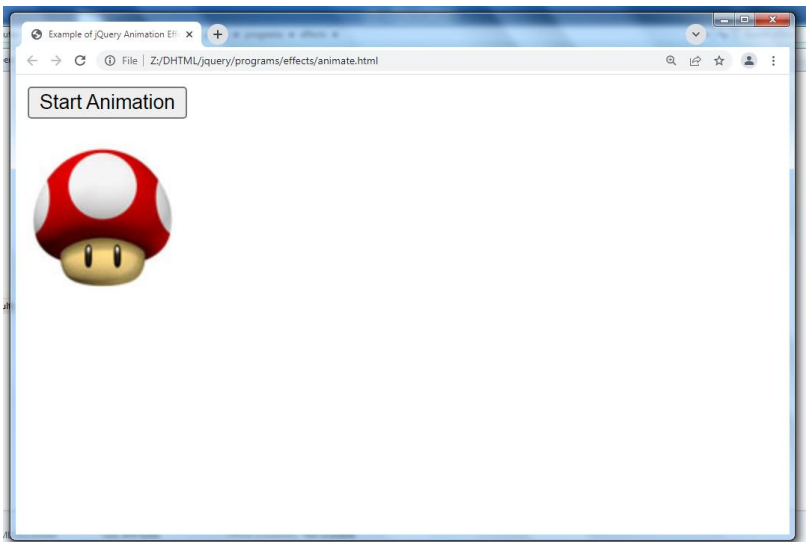
- **The optional stopAll** Boolean parameter specifies whether to remove queued animation or not. Default value is false, that means only the current animation will be stopped, rest of the animations in the queue will run afterwards.
- **The optional goToEnd** Boolean parameter specifies whether to complete the current animation immediately. Default value is false.

**Example-1**

```
<!DOCTYPE html>
<html>
<head>

<title>Example of jQuery Animation Effects</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   img{
      position: relative; /* Required to move element */
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
   $("button").click(function(){
```

```
        $("img").animate({
            left: 300
        });
    });
});
</script>
</head>
<body>
<button type="button">Start Animation</button>
<p>
        <img src="Z:\DHTML\jquery\programs\effects\images\mushroom.jpg"
alt="Mushroom">
</p>
</body>
</html>
```

**Output for above code**

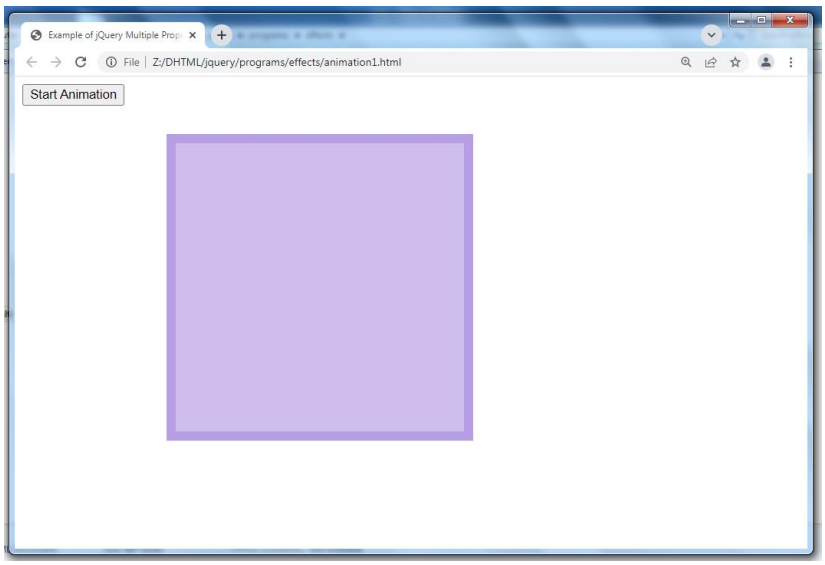


**Example-2**

```
<!DOCTYPE html>
<html>
<head>

<title>Example of jQuery Multiple Properties Animation</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
    .box{
        width: 100px;
        height: 100px;
        background: #9d7ede;
        margin-top: 30px;
        border-style: solid; /* Required to animate border width */
        border-color: #6f40ce;
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    $("button").click(function(){
        $(".box").animate({
            width: "300px",
            height: "300px",
            marginLeft: "150px",
            borderWidth: "10px",
```

```
            opacity: 0.5
        });
    });
});
</script>
</head>
<body>
<button type="button">Start Animation</button>
<div class="box"></div>
</body>
</html>
```

**Output for above code**



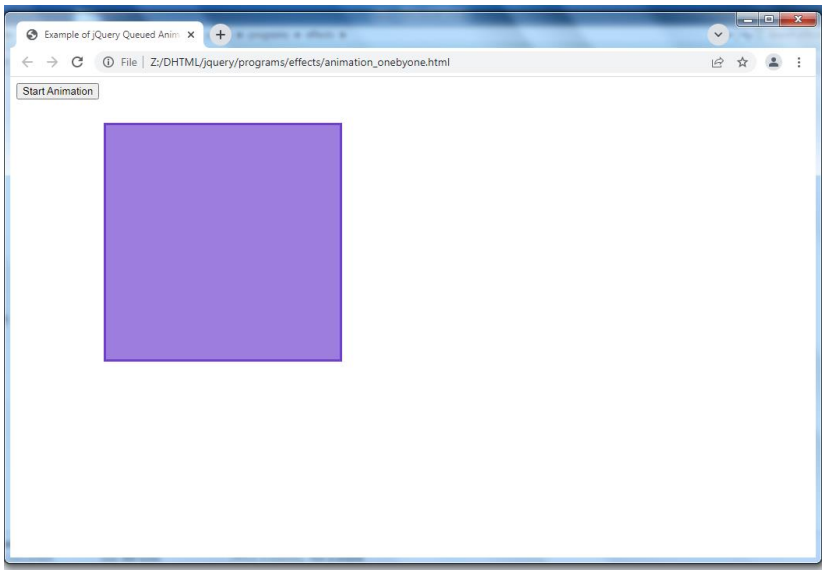

**Example-3**

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Example of jQuery Queued Animation</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
    .box{
        width: 100px;
        height: 100px;
        background: #9d7ede;
        margin-top: 30px;
        border-style: solid; /* Required to animate border width */
        border-color: #6f40ce;
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    $("button").click(function(){
        $(".box")
            .animate({width: "300px"})
            .animate({height: "300px"})
            .animate({marginLeft: "150px"})
            .animate({borderWidth: "10px"})
            .animate({opacity: 0.5});
    });
});
</script>
```

```
</head>
<body>
<button type="button">Start Animation</button>
<div class="box"></div>
</body>
</html>
```

**Output for above code**




**Example-4**

```
<!DOCTYPE html>
<html>
<head>

<title>Example of jQuery Animation with Relative Values</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
```

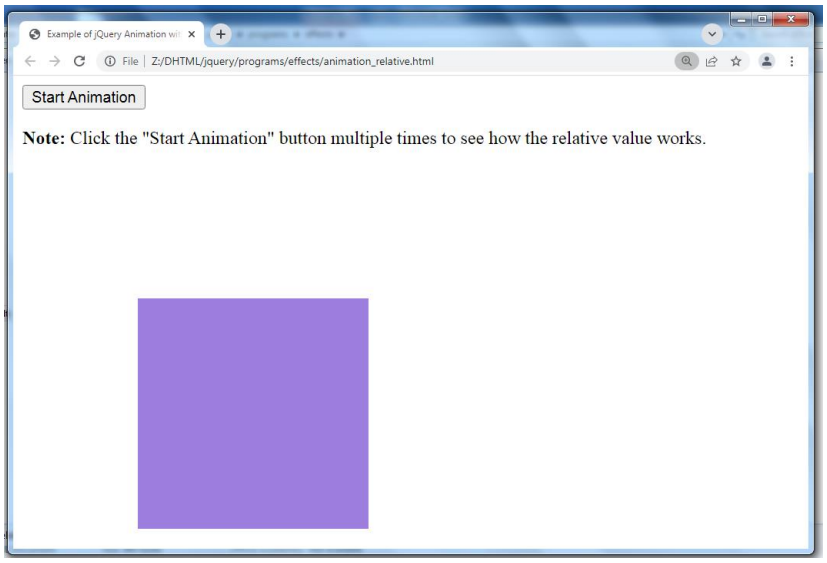
```
<style type="text/css">
    .box{
        width: 100px;
        height: 100px;
        background: #9d7ede;
        margin-top: 30px;
        position: relative; /* Required to move element */
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    $("button").click(function(){
        $(".box").animate({
            top: "+=50px",
            left: "+=50px",
            width: "+=50px",
            height: "+=50px"
        });
    });
});
</script>
</head>
<body>
<button type="button">Start Animation</button>
```

Note: Click the "Start Animation" button multiple times to see how the relative value works.

</body>

</html>

**Output for above code**




## 14.5 JQuery Events

Events are often triggered by the user's interaction with the web page, such as when a link or button is clicked, text is entered into an input box or textarea, selection is made in a select box, key is pressed on the keyboard, the mouse pointer is moved etc.

In some cases, the Browser itself can trigger the events, such as the page load and unload events.

jQuery enhances the basic event-handling mechanisms by offering the events methods

**jQuery Event Methods** Order by Alphabet

This section contains a comprehensive list of event methods belonging to the latest jQuery JavaScript library. All the methods are grouped into categories.

**Mouse Events**

| Method | Description |
|---|---|
| click() | Bind an event handler to be fired when the element is clicked, or trigger that handler on an element. |
| dblclick() | Bind an event handler to be fired when the element is double-clicked, or trigger that event on an element. |
| hover() | Bind one or two handlers to the selected elements, to be executed when the mouse pointer enters and leaves the elements. |
| mousedown() | Bind an event handler to be fired when the mouse button is pressed within the element, or trigger that event on an element. |
| mouseenter() | Bind an event handler to be fired when the mouse enters an element, or trigger that handler on an element. |
| mouseleave() | Bind an event handler to be fired when the mouse leaves an element, or trigger that handler on an element. |
| mousemove() | Bind an event handler to be fired when the mouse pointer moves within the element, or trigger that event on an element. |
| mouseout() | Bind an event handler to be fired when the mouse pointer leaves the element, or trigger that event on an element. |
| mouseover() | Bind an event handler to be fired when the mouse pointer enters the element, or trigger that event on an element. |

| | |
|---|---|
| mouseup() | Bind an event handler to be fired when the mouse button is released within the element, or trigger that event on an element. |

**Keyboard Events**

| Method | Description |
|---|---|
| keydown() | Bind an event handler to be fired when a key is pressed and the element has keyboard focus, or trigger that event on an element. |
| keypress() | Bind an event handler to be fired when a keystroke occurs and the element has keyboard focus, or trigger that event on an element. |
| keyup() | Bind an event handler to be fired when a key is released and the element has keyboard focus, or trigger that event on an element. |

**Form Events**

| **Method** | **Description** |
|---|---|
| blur() | Bind an event handler to be fired when the element loses keyboard focus, or trigger that event on an element. |
| change() | Bind an event handler to be fired when the element's value changes, or trigger that event on an element. |
| focus() | Bind an event handler to be fired when the element gains keyboard focus, or trigger that event on an element. |
| focusin() | Bind an event handler to be fired when the element, or a descendant, gains keyboard |

| | |
|---|---|
| | focus. |
| focusout() | Bind an event handler to be fired when the element, or a descendant, loses keyboard focus. |
| select() | Bind an event handler to be fired when text in the element is selected, or trigger that event on an element. |
| submit() | Bind an event handler to be fired when the form element is submitted, or trigger that event on an element. |

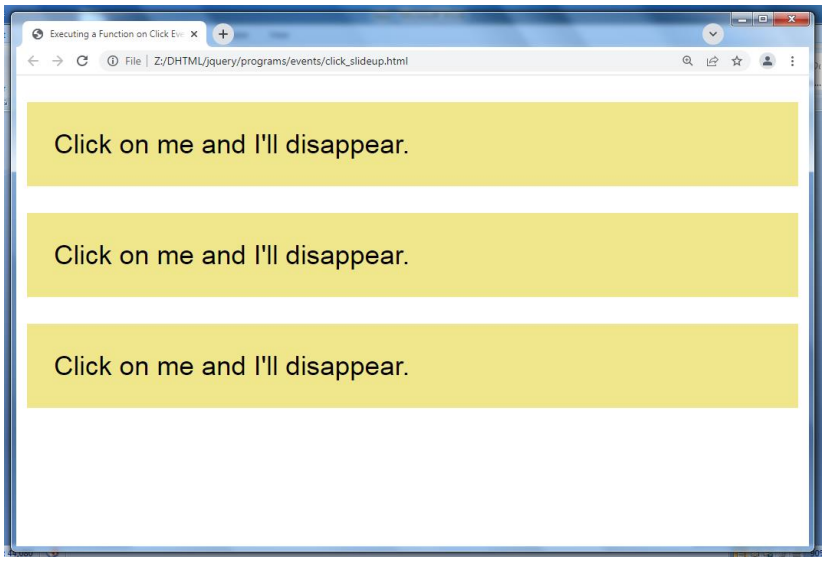
**Document/Browser Events**

| Method | Description |
|---|---|
| load() | Bind an event handler to be fired when the element finishes loading. |
| ready() | Bind an event handler to be fired when the DOM is fully loaded. |
| resize() | Bind an event handler to be fired when the element is resized, or trigger that event on an element. |
| scroll() | Bind an event handler to be fired when the window's or element's scroll position changes, or trigger that event on an element. |
| unload() | Bind an event handler to be fired when the user navigates away from the page. |
| load() | Bind an event handler to be fired when the element finishes loading. |
| ready() | Bind an event handler to be fired when the DOM is fully loaded. |
| resize() | Bind an event handler to be fired when the element is resized, or trigger that event on an element. |

**Example-1**

```
<!DOCTYPE html>
<html>
<head>
<title>Executing a Function on Click Event in jQuery</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 20px;
      font: 20px sans-serif;
      background: khaki;
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
   $("p").click(function(){
      $(this).slideUp();
   });
});
</script>
</head>
<body>
<p>Click on me and I'll disappear.</p>
<p>Click on me and I'll disappear.</p>
<p>Click on me and I'll disappear.</p>
</body>
</html>
```

**Output for above code**

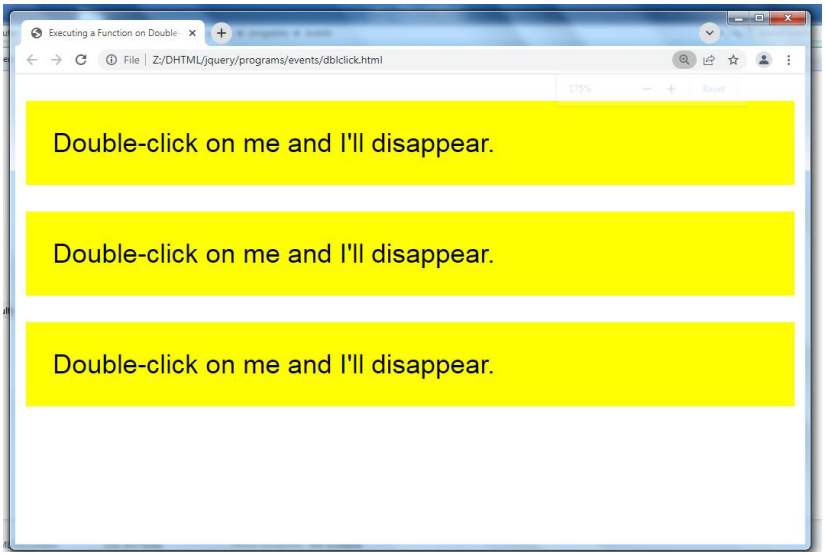



**Example-2**

```
<!DOCTYPE html>
<html>
<head>

<title>Executing a Function on Double-click Event in jQuery</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
    p{
        padding: 20px;
        font: 20px sans-serif;
        background: yellow;
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    $("p").dblclick(function(){
        $(this).slideUp();
    });
```

```
});
</script>
</head>
<body>
<p>Double-click on me and I'll disappear.</p>
<p>Double-click on me and I'll disappear.</p>
<p>Double-click on me and I'll disappear.</p>
</body>
</html>
```

**Output for above code**

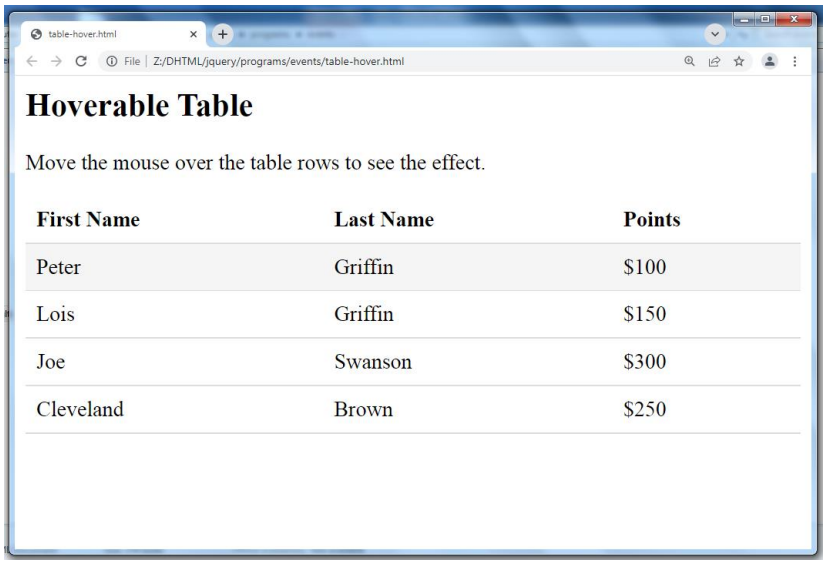



**Example-3**

```
<!DOCTYPE html>
<html>
<head>

<title>Executing a Function on Hover Event in jQuery</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
```

```
    p{
        padding: 20px;
        font: 20px sans-serif;
        background: #f2f2f2;
    }
    p.highlight{
        background: yellow;
    }
</style>
<script type="text/javascript">
$(document).ready(function(){
    $("p").hover(function(){
        $(this).addClass("highlight");
    }, function(){
        $(this).removeClass("highlight");
    });
});
</script>
</head>
<body>
<p>Place mouse pointer on me.</p>
<p>Place mouse pointer on me.</p>
<p>Place mouse pointer on me.</p>
</body>
</html>
```

**Output for above code**

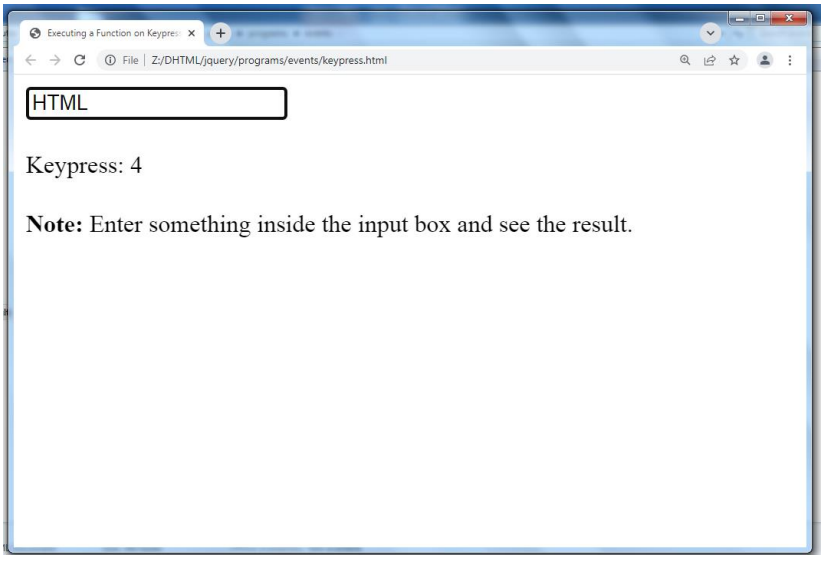



**Example-4**

```
<!DOCTYPE html>
<html>
<head>

<title>Executing a Function on Keypress Event in jQuery</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 10px;
      background: lightgreen;
      display: none;
   }
   div{
      margin: 20px 0;
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
```

```
    var i = 0;
    $('input[type="text"]').keypress(function(){
        $("span").text(i += 1);

    });
});
</script>
</head>
<body>
<input type="text">
<div>Keypress: <span>0</span></div>
        <div><strong>Note:</strong> Enter something inside the input box and see the result.</div>

</body>
</html>
```

**Output for above code**



**Example-5**

```
<!DOCTYPE html>
<html>
<head>

<title>Executing a Function on Mouseenter Event in jQuery</title>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<style type="text/css">
   p{
      padding: 20px;
      font: 20px sans-serif;
      background: #f2f2f2;
   }
   p.highlight{
      background: yellow;
   }
</style>
<script type="text/javascript">
$(document).ready(function(){
   $("p").mouseenter(function(){
      $(this).addClass("highlight");
   });
   $("p").mouseleave(function(){
      $(this).removeClass("highlight");
   });
});
</script>
</head>
<body>
<p>Place mouse pointer on me.</p>
```

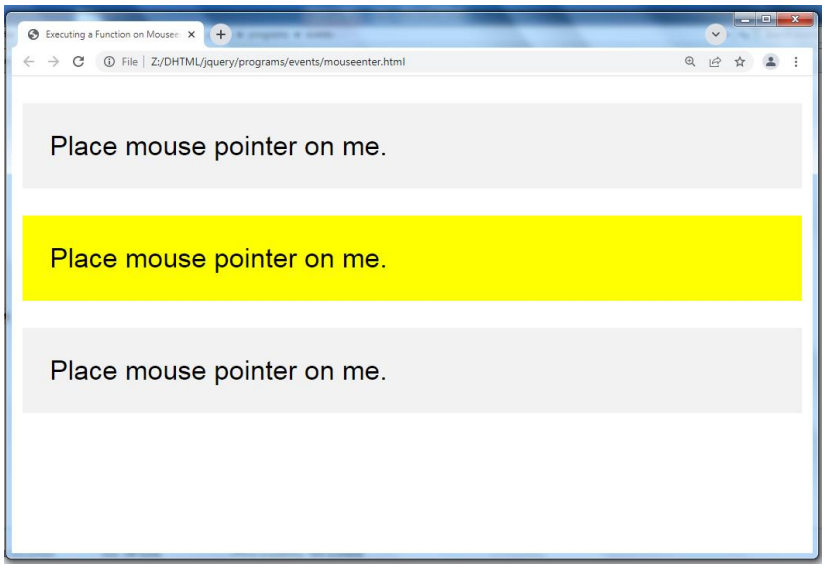
Place mouse pointer on me.

Place mouse pointer on me.

</body>

</html>

**Output for above code**




**bind() Method**

The bind() method attaches one or more event handlers for selected elements, and specifies a function to run when the event occurs.

Syntax

$(*selector*).bind(*event,data,function*)

**Example**

<!DOCTYPE html>

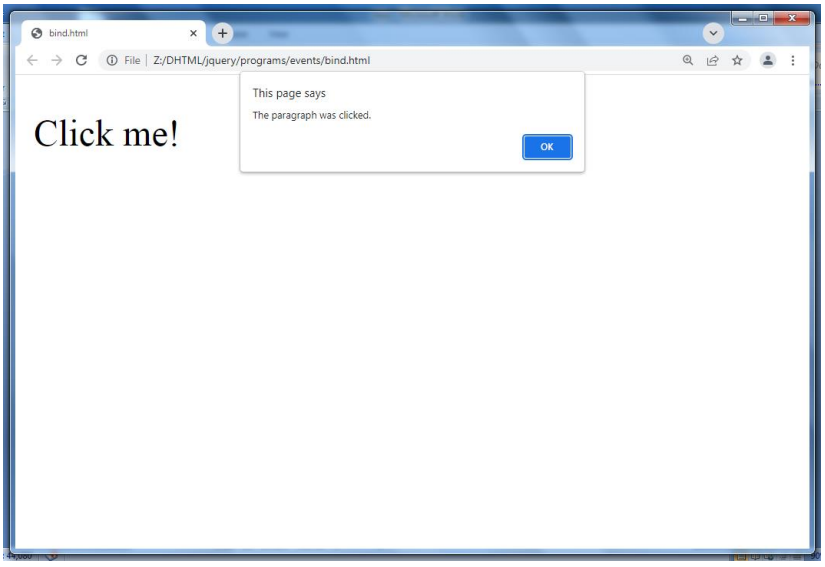
<html>

<head>

<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

```
<script>
$(document).ready(function(){
    $("p").bind("click", function(){
        alert("The paragraph was clicked.");
    });
});
</script>
</head>
<body>

<p>Click me!</p>
</body>
</html>
```

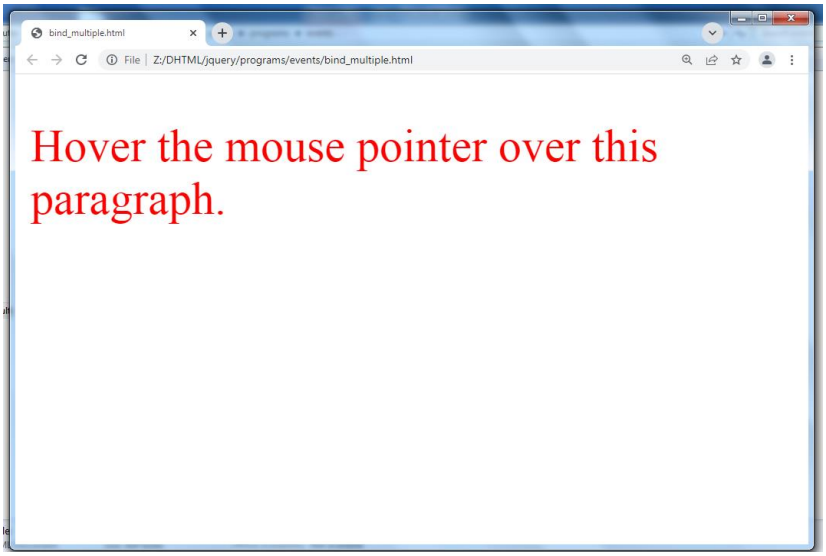
**Output for above code**



**Example-2**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
    $("p").bind("mouseover mouseout", function(){
        $("p").toggleClass("intro");
    });
});
</script>
<style>
.intro {
    font-size: 150%;
    color: red;
}
</style>
</head>
<body>

<p>Hover the mouse pointer over this paragraph.</p>
</body>
</html>
```

**Output for above code**




**unbind() Method**

The unbind() method removes event handlers from selected elements.

This method can remove all or selected event handlers, or stop specified functions from running when the event occurs.

Syntax

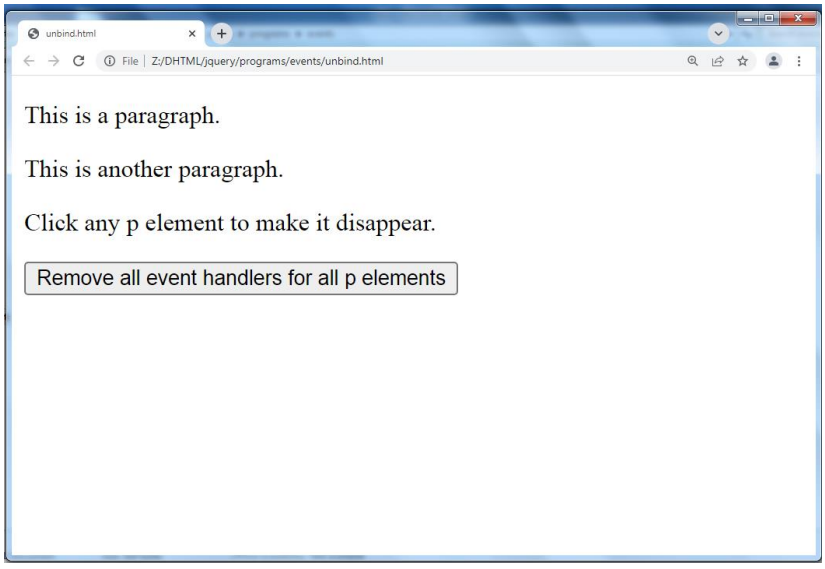
$(*selector*).unbind(*event,function,eventObj*)

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).slideToggle();
```

```
    });
    $("button").click(function(){
        $("p").unbind();
    });
});
</script>
</head>
<body>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<p>Click any p element to make it disappear.</p>
<button>Remove all event handlers for all p elements</button>
</body>
</html>
```

**Output for above code**



## 14.6 Summary

In this chapter you have learned about:

- JQuery Selectors
- JQuery Traversing
- JQuery Attributes
- JQuery Effects
- JQuery Events

# Unit 15: Introduction to XML

## Unit Structure

15.1. What is XML

15.2. XML Verses HTML

15.3. XML Syntax

15.4. XML References

15.5. XML Declaration

15.6. XML Comments

15.7. XML Terminologies

15.8. XML Namespace

15.9. Summary

# 15.1 What is XML?

XML stands for Extensible Markup Language.

**Extensible**

XML is extensible. It lets you define your own tags, the order in which they occur, and how they should be processed or displayed.

**Markup**

The most recognizable feature of XML is its tags, or elements.

In fact, the elements you'll create in XML will be very similar to the elements you've already been creating in your HTML documents. However, XML allows you to define your own set of tags.

**Language**

- XML is a language that's very similar to HTML. It's much more flexible than HTML because it allows you to create your own custom tags.
- However, it's important to realize that XML is not just a language.
- XML is a meta-language: a language that allows us to create or define other languages.
- XML was designed to describe data.
- XML tags are not predefined in XML. You must define your own tags.
- XML is self describing.
- XML uses a DTD (Document Type Definition) to formally describe the data.

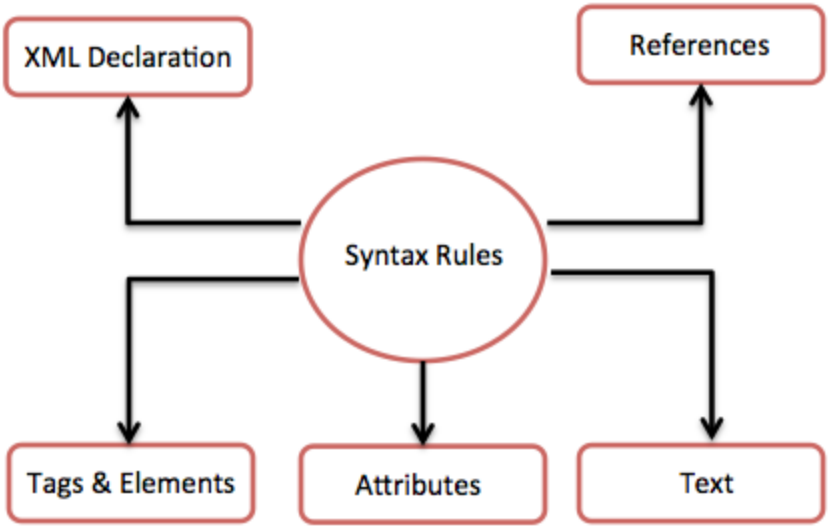
The way XML works is that programmers mark-up a text-based document with tags (similar to HTML tags) that tell what each word, number or group of words represent. For example, the tag <invoice number> might be used to describe the number of an invoice.

## 15.2 XML Verses HTML

| Sr.No | HTML | XML |
|---|---|---|
| 1 | HTML is an abbreviation for Hypertext Markup Language. | XML stands for Extensible Markup Language. |
| 2 | HTML was designed to display data with focus on how data looks. | XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is. |
| 3 | HTML is a markup language itself. | XML provides a framework for defining markup languages. |
| 4 | HTML is a presentation language. | XML is neither a programming language nor a presentation language. |
| 5 | HTML is case insensitive. | XML is case sensitive. |
| 6 | HTML is used for designing a web-page to be rendered on the client side. | XML is used basically to transport data between the application and the database. |
| 7 | HTML has it own predefined tags. | While what makes XML flexible is that custom tags can be defined and the tags are invented by the author of the XML document. |
| 8 | HTML is not strict if the user does not use the closing tags. | XML makes it mandatory for the user the close each tag that has been used. |
| 9 | HTML does not preserve white space. | XML preserves white space. |
| 10 | HTML is about displaying data,hence static. | XML is about carrying information,hence dynamic. |

## 15.3 XML Syntax

The following diagram depicts the syntax rules to write different types of markup and text in an XML document.




Let us see each component of the above diagram in detail:

XML Declaration

The XML document can optionally have an XML declaration. It is written as below:

<?xml version="1.0" encoding="UTF-8"?>

Where version is the XML version and encoding specifies the character encoding used in the document.

**XML Naming Rules**

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces
- Any name can be used, no words are reserved (except xml).

**Syntax Rules for XML declaration**

- The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case.
- If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.
- An HTTP protocol can override the value of encoding that you put in the XML declaration.

**Tags and Elements**

An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. XML-elements' names are enclosed by triangular brackets <> as shown below:

<element>

An element can contain:

- text
- attributes
- other elements
- or a mix of the above

**Syntax Rules for Tags and Elements**

**Element Syntax**: Each XML-element needs to be closed either with start or with end elements as shown below:

<element>....</element>

or in simple-cases, just this way:

<element/>

**Nesting of elements**: An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

Following example shows incorrect nested tags:

```
<?xml version="1.0"?>
<contact-info>
<company>TutorialsPoint
<contact-info>
</company>
```

Following example shows correct nested tags:

```
<?xml version="1.0"?>
<contact-info>
<company>TutorialsPoint</company>
<contact-info>
```

**Root element:** An XML document can have only one root element. For example, following is not a correct XML document, because both the x and y elements occur at the top level without a root element:

```
<x>...</x>
<y>...</y>
```

The following example shows a correctly formed XML document:

```
<root>
<x>...</x>
<y>...</y>
</root>
```

Case sensitivity: The names of XML-elements are case-sensitive. That means the name of the start and the end elements need to be exactly in the same case.

For example <contact-info> is different from <Contact-Info>.

**Empty XML Elements**

An element with no content is said to be empty.

In XML, you can indicate an empty element like this:

```
<element></element>
```

You can also use a so called self-closing tag:

```
<element />
```

**Attributes**

An element can have multiple unique attributes. Attribute gives more information about XML elements. Syntax Rules for XML Attributes

**Syntax**

An XML attribute has following syntax:

<element-name attribute1 attribute2 >
....content..
< /element-name>

where attribute1 and attribute2 has the following form:

name = "value"

value has to be in double (" ") or single (' ') quotes. Here, attribute1 and attribute2 are unique attribute labels.

- Same attribute cannot have two values in a syntax. The following example shows incorrect syntax because the attribute b is specified twice:

  <a b="x" c="y" b="z">....</a>

- Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks. Following example demonstrates incorrect xml syntax:

  <a b=x>....</a>

  In the above syntax, the attribute value is not defined in quotation marks.

## 15.4 XML References

References usually allow you to add or include additional text or markup in an XML document. References always begin with the symbol "&" ,which is a reserved character and end with the symbol ";". XML has two types of references:

Entity References: An entity reference contains a name between the start and the end delimiters. For example & where amp is name. The name refers to a predefined string of text and/or markup.

Character References: These contain references, such as A, contains a hash mark (“#”) followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refers to alphabet "A".

**XML Text**

The names of XML-elements and XML-attributes are case-sensitive, which means the name of start and end elements need to be written in the same case.

To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files.

Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored.

Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly. To use them, some replacement-entities are used, which are listed below:

| not allowed character | replacement-entity | character description |
|---|---|---|
| < | < | less than |
| > | > | greater than |
| & | & | ampersand |
| ' | ' | apostrophe |
| " | " | quotation mark |

This will generate an XML error:

<message>salary < 1000</message>

To avoid this error, replace the "<" character with an entity reference:

<message>salary < 1000</message>

# 15.5 XML Declarations

XML declaration contains details that prepare an XML processor to parse the XML document.

It is optional, but when used, it must appear in first line of the XML document.

**Syntax**

Following syntax shows XML declaration:

```
<?xml
  version="version_number"
```

```
  encoding="encoding_declaration"
  standalone="standalone_status"
?>
```

Each parameter consists of a parameter name, an equals sign (=), and parameter value inside a quote

| Parameter | Parameter_value | Parameter_description |
|---|---|---|
| Version | 1.0 | Specifies the version of the XML standard used. |
| Encoding | UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP | It defines the character encoding used in the document. UTF-8 is the default encoding used. |
| Standalone | yes or no. | It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to no. Setting it to yes tells the processor there are no external declarations required for parsing the document. |

**All XML Elements Must Have a Closing Tag**

**XML Tags are Case Sensitive**

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.

Opening and closing tags must be written with the same case:

```
<Message>This is incorrect</message>
<message>This is correct</message>
```

**XML Elements Must be Properly Nested**

In HTML, you might see improperly nested elements:

This text is bold and italic

In XML, all elements must be properly nested within each other:

This text is bold and italic

**XML Attribute Values Must be Quoted**

XML elements can have attributes in name/value pairs just like in HTML.

In XML, the attribute values must always be quoted.

```
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

## 15.6 XML Comments

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

**XML Comments Rules**

Following rules are needed to be followed for XML comments:

- Comments cannot appear before XML declaration.
- Comments may appear anywhere in a document.
- Comments must not appear within attribute values.
- Comments cannot be nested inside the other comments.

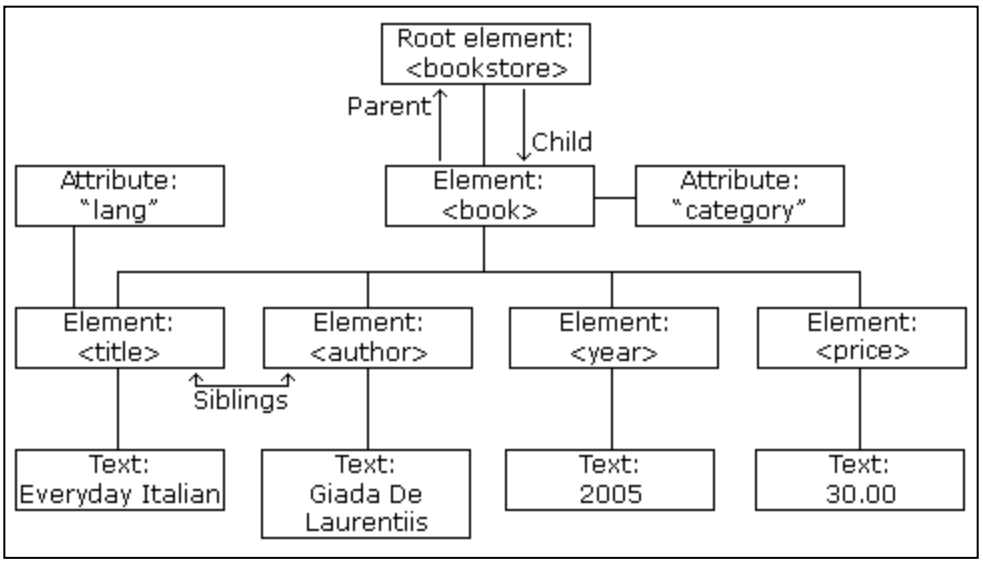
**White-space is preserved in XML**

XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):

XML: Hello Tove
HTML: Hello Tove

# 15.7 XMLTerminologies

XML Tree Structure




**XML Tree Structure**

XML documents are formed as element trees.

An XML tree starts at a root element and branches from the root to child elements.

All elements can have sub elements (child elements):

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

The terms parent, child, and sibling are used to describe the relationships between elements.

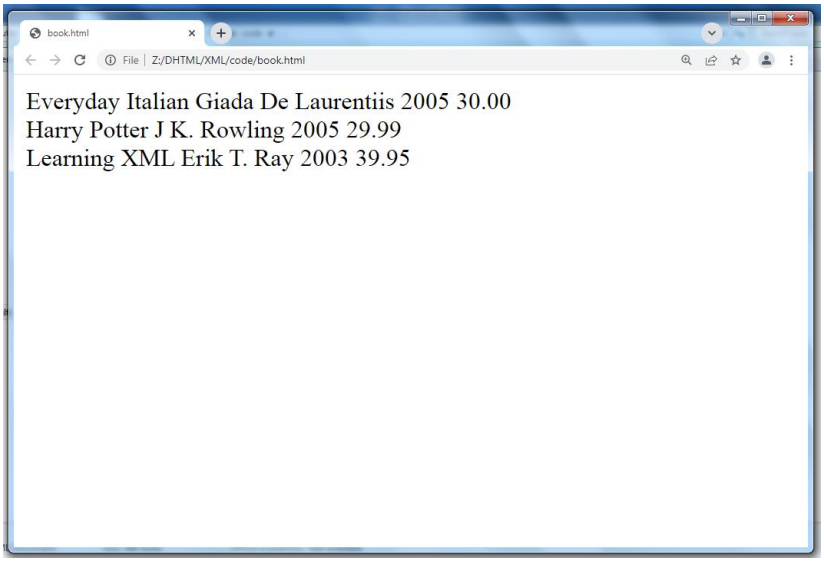
Parents have children. Children have parents. Siblings are children on the same level (brothers and sisters).

All elements can have text content (Harry Potter) and attributes (category="cooking").

### An Example XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

### Output for above code



### Self-Describing Syntax

XML uses a much self-describing syntax.

**A prolog** defines the XML version and the character encoding:

<?xml version="1.0" encoding="UTF-8"?>

The next line is **the root element** of the document:

<bookstore>

The next line starts a <book> element:

<book category="cooking">

The <book> elements have **4 child elements**: <title>,< author>, <year>, <price>.

<title lang="en">Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>

The next line ends the book element:

</book>

## 15.8 XML Namespace

XML Namespaces provide a method to avoid element name conflicts.

### Namespace Declaration

A Namspace is declared using reserved attributes. Such an attribute name must either be xmlns or begin with xmlns: shown as below:

<element xmlns:name="description">

### Syntax

- The Namespace starts with the keyword xmlns.
- The word name is the Namespace prefix.

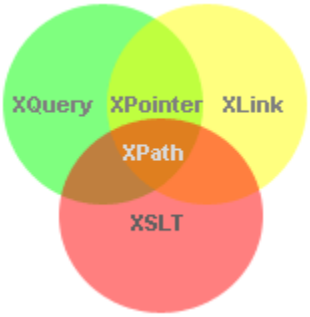
- The URL is the Namespace identifier.

**Example**

```
<?xml version="1.0" encoding="UTF-8"?>

<cont:contact xmlns:cont="contactdetails">

<cont:name>Tanmay Patil</cont:name>

<cont:company>TutorialsPoint</cont:company>

<cont:phone>(011) 123-4567</cont:phone>

</cont:contact><br>


<cont1:contact xmlns:cont1="contactdetails">

<cont1:name>patel</cont1:name>

<cont1:company>tp</cont1:company>

<cont1:phone>s123-4567</cont1:phone>

</cont1:contact>
```

**XLink and XPointer**

XLink is used to create hyperlinks in XML documents.




XLink is used to create hyperlinks within XML documents

Any element in an XML document can behave as a link

**XLink Syntax**

In HTML, the <a> element defines a hyperlink. However, this is not how it works in XML.

In XML documents, you can use whatever element names you want

To get access to the XLink features we must declare the XLink namespace. The XLink namespace is: "http://www.w3.org/1999/xlink".

**Example**

```
<?xml version="1.0" encoding="UTF-8"?>

<homepages xmlns:xlink="http://www.w3.org/1999/xlink">
  <homepage xlink:type="simple" xlink:href="http://www.w3schools.com">Visit
W3Schools</homepage>
  <homepage xlink:type="simple" xlink:href="http://www.w3.org">Visit
W3C</homepage>
</homepages>
```

**XPointer**

XPointer allows links to point to specific parts of an XML document.

XPointer uses XPath expressions to navigate in the XML document.

XPointer allows you to link to specific parts of the document. To link to a specific part of a page, add a number sign (#) and an XPointer expression after the URL in the xlink:href attribute

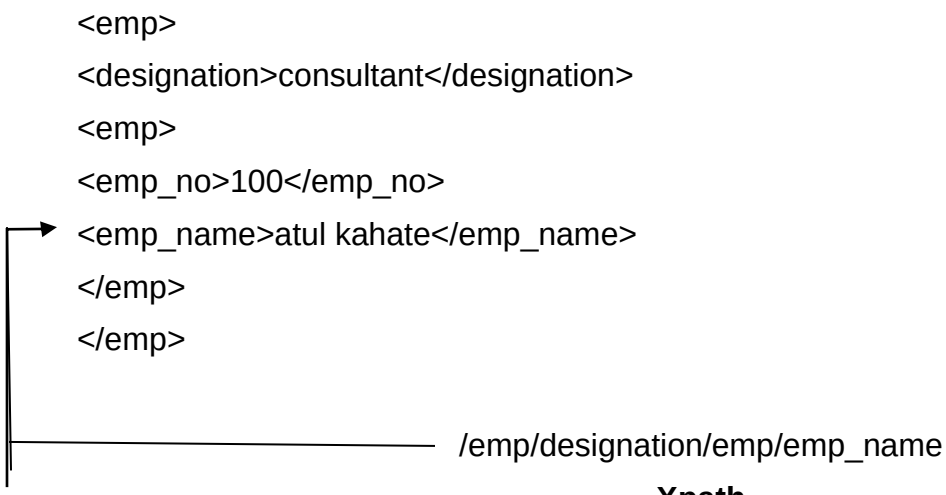
**Example**

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
<book pubyear="2004">
<book_title>Computer networks</book_title>
<author  id=”AST”>Tanenbaum</author>
</books>
<?xml version="1.0" encoding="UTF-8"?>
<myauthor type=”simple” href=”http://www.test.com/books.xml#AST”>
```

**XPath**

XPath is a syntax for defining parts of an XML document

XPath uses path expressions to navigate in XML documents




**Xpath**

**XML Schema**

XML Schema is commonly known as XML Schema Definition (XSD).

It is used to describe and validate the structure and the content of XML data.

XML schema defines the elements, attributes and data types.

Schema element supports Namespaces.

It is similar to a database schema that describes the data in a database.

**Syntax**

You need to declare a schema in your XML document as follows:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
</xs:schema>
```

We have to define the "shiporder" element.

This element has an attribute and it contains other elements, therefore we consider it as a complex type.

The child elements of the "shiporder" element is surrounded by a xs:sequence element that defines an ordered sequence of sub elements:

```
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

**Example**

The following example shows how to use schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shipto">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="address" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

The basic idea behind XML Schemas is that they describe the legitimate format that an XML document can take.

**XHTML**

XHTML is a revised version of  HTML with rules from XML.

It is a combination of HTML and XML documentation.

## 15.9 Summary

In this chapter you have learned about:

- What is XML
- XML Verses HTML
- XML Syntax
- XML References
- XML Declaration
- XML Comments
- XML Terminologies
- XML Namespace

BAOU
Education for All

## યુનિવર્સિટી ગીત

સ્વાધ્યાયઃ પરમં તપઃ
સ્વાધ્યાયઃ પરમં તપઃ
સ્વાધ્યાયઃ પરમં તપઃ

શિક્ષણ, સંસ્કૃતિ, સદ્‌ભાવ, દિવ્યબોધનું ધામ
ડૉ. બાબાસાહેબ આંબેડકર ઓપન યુનિવર્સિટી નામ;
સૌને સૌની પાંખ મળે, ને સૌને સૌનું આભ,
દશે દિશામાં સ્મિત વહે હો દશે દિશે શુભ-લાભ.

અભણ રહી અજ્ઞાનના શાને, અંધકારને પીવો ?
કહે બુદ્ધ આંબેડકર કહે, તું થા તારો દીવો;
શારદીય અજવાળા પહોંચ્યાં ગુર્જર ગામે ગામ
ધ્રુવ તારકની જેમ ઝળહળે એકલવ્યની શાન.

સરસ્વતીના મયૂર તમારે ફળિયે આવી ગહેકે
અંધકારને હડસેલીને ઉજાસના ફૂલ મહેંકે;
બંધન નહીં કો સ્થાન સમયના જવું ન ઘરથી દૂર
ઘર આવી મા હરે શારદા દૈન્ય તિમિરના પૂર.

સંસ્કારોની સુગંધ મહેંકે, મન મંદિરને ધામે
સુખની ટપાલ પહોંચે સૌને પોતાને સરનામે;
સમાજ કેરે દરિયે હાંકી શિક્ષણ કેરું વહાણ,
આવો કરીયે આપણ સૌ
ભવ્ય રાષ્ટ્ર નિર્માણ...
દિવ્ય રાષ્ટ્ર નિર્માણ...
ભવ્ય રાષ્ટ્ર નિર્માણ

●

DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY
(Established by Government of Gujarat)
'Jyotirmay' Parisar,
Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481
Website : www.baou.edu.in | Email : office.scs@baou.edu.in