Web Design

Sai Krishna B V

Contents

Part I
HTML 5

1 Introduction

HTML stands for HyperText Mark Up Language, a way of marking up files so that browsers know how to display the content as a webpage. HTML uses tags to differentiate between content and the instructions for displaying them.

1.1 History

HTML (I) was created in 1990 as a way to electronically connect documents via hyperlinks. HTML was created by Sir Tim Berners-Lee in late 1991 but was not released officially, which was published in 1995 as HTML 2.0. HTML 4.01 was published in late 1999 and was a major version of HTML. HTML is a very evolving markup language and has evolved with various versions updating.

- First graphical browser - Mosaic 1993
- Netscape - 1994
- Internet Explorer - 1995

The competition between these led to **The Browser Wars**. And lead to origin of **Best viewed on** messages. As each browser had a few tags that would work only that particular browser.

1.2 Web Standards

- IETF:
  The Internet Engineering Task Force (IETF) is an open standards organization, which develops and promotes voluntary Internet standards, in particular the standards that comprise the Internet protocol suite (TCP/IP).
- W3C:
  WorldWideWeb Consortium deals with HTML and evolution of HTML. They know what kind of tags each browser will support
- WAI:
  The Web Accessibility Initiative they make sure that irrespective of how people are accessing the web, they have the same ability to view the content

1.3 Evolution of Browsers

1990 - 1994	Html was simple, content was primarily text-based
1993	Mosaic enters with images
1995 - 1999	Cross-browser compatibility falls apart
2000 - 2005	Browsers move toward separating content from style
2005 - 2008	Using HTML files in coordination with CSS becomes new Standard

1.4 Evolution of HTML

1993	HTML 1.0	Developed by Tim Berners Lee to link document
1995	HTML 2.0	Developed by IETF RFC to include stylized text and tables
1996	CSS 1	
1997	HTML 3.2	Developed by W3C and included browser specific features
1997	HTML 4.0	A move back to normalizing the pages across platforms
1998	CSS 2	
1999	HTML 4.01	Introduced different document types
2012	HTML 5	Back to HTML plus multimedia and semantic tags

HTML5 is a cooperation between W3C and WHATWG (Web Hypertext Application Technology Working Group). They have established guidelines of how HTML5 should be approached as browsers go to support it

- New features should be based on HTML, CSS, DOM and Javascript
- Reduce the need for external plugins
- More markup to replace scripting
- HTML5 should be device independent

1.5 What happens when you enter the url in your browser?

The user who types the url is the client, by typing the url the client makes a request to the server to get a particular page, the server responds to this with the appropriate files found in the address given by the client.

1.5.1 Uniform Resource Locator (URL)

It mainly has 3 parts

- protocol: how to connect
  The different available protocols are:
  - HTTP: Hypertext Transfer Protocol
  - HTTPS: Secure Hypertext Transfer Protocol
  - FTP: File Transfer Protocol

  HTTP
- domain: the server
  - Identifies the entity you want to connect to
  - The different top level domain names are given by Internet Corporation for Assigned Names and Numbers (ICANN)
- (optional) document: the specific file needed
  Usually we skip the document, for example: www.wikipedia.org, here we are not trying to access a particular page on that server. In such cases a default page is returned, in most cases the default page is "index.html"

1.5.2 IP address

Every website is uniquely represented by its IP address, but since it is difficult to remember a set of numbers corresponding to each website, domain names are used.

Internet Protocol Version 6 (IPv6) is the communication protocol that identifies computers on networks. Every computer will have an unique IP address

Domain Name Server (DNS) will look up the IP address based on the url typed by the client

1.5.3 HTTP Request

Once the DNS obtains the IP address corresponding to the url. The browser creates an HTTP Request, and this is sent to the IP address.
Lots of information like header, cookies, form, data is present in the HTTP Request.

1.5.4 HTTP Response

Based on the HTTP Request received, the server responds to with the content that was requested. The requested files (not webpages) are returned to the client in the HTTP Response.

Usually the response includes the status codes

1. Informational responses (100–199)
2. Successful responses (200–299)
3. Redirects (300–399)
4. Client errors (400–499)
5. Server errors (500–599)

1.6 Browsers

There are different browsers available and each have their own pros and cons

1.6.1 Internet Explorer

For a long time it was the most popular browser because it was preinstalled on Microsoft Windows. And it was platform independent (i.e. Internet Explorer does not work on a Mac)

1.6.2 Microsoft Edge

It is a web browser developed by Microsoft and was released Windows 10 and Xbox One in 2015, then for Android and iOS in 2017, and for macOS in 2019. Edge is meant to replace Internet Explorer

1.6.3 Google Chrome

Google Chrome is a cross-platform, free web browser developed by Google. It was first released in 2008 for Microsoft Windows, and was later ported to Linux, macOS, iOS, and Android where it is the default browser built into the OS. Provides greater security as compared to the other web browser

1.6.4 Mozilla Firefox

Mozilla Firefox, or simply Firefox, is a free and open-source web browser developed by the Mozilla Foundation and its subsidiary, Mozilla Corporation. Available for Windows, Mac, Linux and other BSD (Berkeley Software Distribution) operating systems.

The main factor that is to be considered while choosing a browser is its **Accessibility**

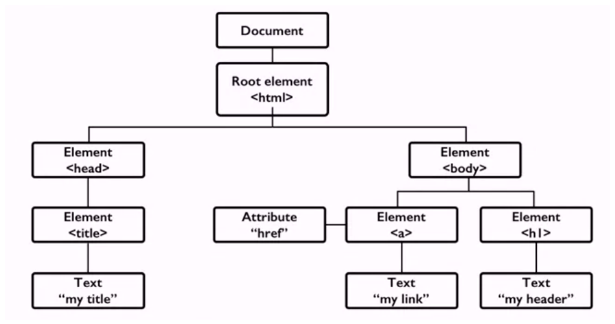
1.6.5 What is accessibility?

Accessibility is basically the ability of the browser to support all the special functionalities that are available with the evolution in the HTML language. The accessibility of different browsers to various HTML5 features can be seen in the following link.

`https://www.html5accessibility.com/`

2 The Document Object Model (DOM)

DOM provides common tree-like structure that all pages should follow



Three part of a well-formed document

1. Doctype
   Version of HTML that you will be using

2. Head
   Metadata

3. Body
   Displayable content

3 HTML Tags

3.0.1 Comments

$$<!--\ ..content...\ -->$$

is the syntax used to write a comment in HTML

3.0.2 Block vs Inline elements

Block-level elements begin on a newline, and occupy any available width. Block-level elements may be nested inside one another and may wrap inline-level elements.

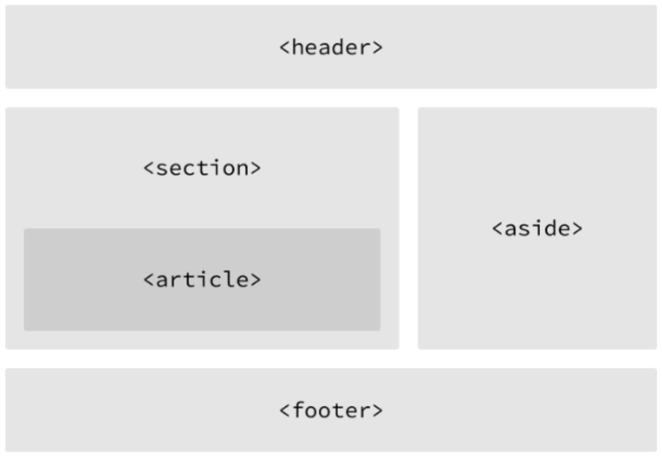
Inline-level elements do not begin on a newline, they fall into the normal flow of a document, and only maintain the width of their content. Inline-level elements may be nested inside one another; however they cannot wrap block-level elements.

3.1 <div> and <span>

The <div> and <span> tags are HTML elements that act as containers solely for styling purposes.

A <div> is a block-level element commonly used to identify large groupings of content. A <span> is an inline-level element commonly used to identify smaller groupings of text within a block-level element.
class or *id* attributes are commonly used with $< div >$ and $< span >$ tags for styling

The problem with $< div >$ tag is that it does not provide any semantic value and hence it gets difficult to determine the intention of these divisions, to overcome this HTML5 introduced new structurally based elements $< header >$, $< nav >$, $< article >$, $< section >$, $< aside >$ and $< footer >$



3.1.1 Header

It is used to identify the top of a page, article, section. This is a structural element that outlines the heading of a segment of a page. IT falls within the $< body >$ element

3.1.2 Navigation

This element identifies a section of major navigational links on a page. Most commonly, links included within the $< nav >$ element will link to other pages within the same website or to parts of the same webpage

3.1.3 Article

This element is used to identify a section of independent, self-contained content that may be independently distributed or reused.

3.1.4 Section

This element is used to identify a thematic grouping of content. It is commonly used to break up and provide hierarchy to a page

3.1.5 Aside

This element holds content such as sidebars, inserts or brief explanations, that is tangentially related to the content surrounding it.

3.1.6 Footer

This element identifies the closing or end of a page, article, section

3.2 Encoding special characters

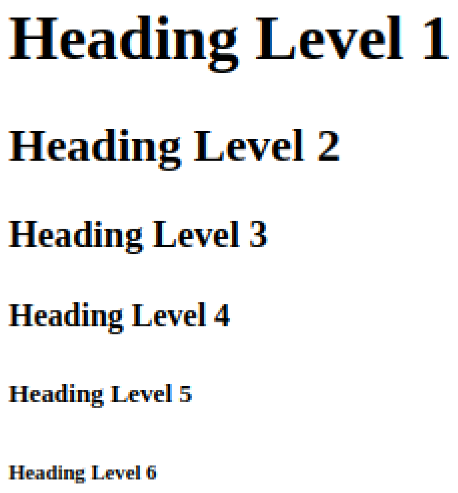
Special characters include punctuation marks, accented letters and symbols. When types directly in HTML they may be misunderstood thus they are encoded. Each encoded character will begin with & and end with a semicolon. A long list of available encodings in HTML can be found in the following link

`http://copypastecharacter.com`

3.3 Headings

Headings are block-level elements and come in 6 different rankings $< h1 >$ through $< h6 >$. Headings should be used in an order that is relevant to the content of a page. The primary heading of a page should be writtent within $< h1 > \; < /h1 >$ and the subsequent headings should use
$< h2 >, < h3 >< h4 >< h5 >< h6 >$ according to the importance of the heading

```
<h1>Heading Level 1</h1>
<h2>Heading Level 2</h2>
<h3>Heading Level 3</h3>
<h4>Heading Level 4</h4>
<h5>Heading Level 5</h5>
<h6>Heading Level 6</h6>

OUTPUT
```



Heading Level 1

Heading Level 2

Heading Level 3

Heading Level 4

Heading Level 5

Heading Level 6

3.4 Paragraphs

Paragraphs are defined using $< p >$ block-level element

3.4.1 Bold text

To make the text bold, the $< strong >$ tag is used. This is an inline-level element. Alternatively, $< b >$ can also be used to bold the text.

$< strong >$ is semantically used to give strong importance to text, where as $< b >$ is semantically used to stylistically offset text.

3.4.2 Italicize text

To italicize text, the $< em >$ tag is used which is an inline-level element. The $< em >$ element is used semantically to place a stressed emphasis on text. The other option $< i >$ is used semantically to convey text in an alternative voice or tone.

3.5 Hyperlinks

Hyperlink provides the ability to link from one webpage to another. Hyperlinks are established using the anchor, $< a >$ inline-level element. The $href$ attribute known as hyperlink reference, is required to link to other webpages

<a href="https://www.google.com">Google</a>

3.5.1 Linking to an Email address

When this hyperlink is clicked, it opens the user's default email client and pre-populates part of an email.

To create this *href* attribute needs to start with *mailto* : followed by the email address to which the email should be sent. Additionally subject, body, text and other information for the email may be populated. To add a subject line include *subject* = parameter. The first parameter following the email address must begin with a question mark (?), to bind it to the hyperlink path. And if the subject has multiple lines %20 can be used for spaces. For including the body of the email use *body* = parameter

```
<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">

<title>Hyperlink Email</title>
</head>
<body>
<a href = "mailto:saikrishnabv1998@gmail.com?subject=Test%20Email
&body=Checking%20how%20hyperlinking%20to%20mail%20works">Email me</a>
</body>
</html>
```

3.5.2 Opening links in new window

Usually link opens in the same window from which they are clicked; however, links may also be opened in new windows. To trigger the action of opening a link in a new window, the *target* attribute is used with a value of *_blank*

```
<a href="https://google.com/" target="_blank">Google</a>
```

3.5.3 Linking to parts of the same page

Common example of these same-page links are "Back to top" links that return a user to the top of a page.

We can create an on-page link by first setting an id attribute on the element we wish to link to, then using the value of that id attribute within an anchor element's href attribute.

```
<body id="top">
  ...
  <a href="#top">Back to top</a>
  ...
</body>
```

3.6 Creating Lists

HTML provides three different types of lists to choose from: unordered, ordered, and description lists.

There are multiple ways to style these lists with CSS. For example, we can choose what type of marker to use on a list. The marker could be square, round, numeric, alphabetical, or perhaps nonexistent. Also, we can decide if a list should be displayed vertically or horizontally.

3.6.1 Unordered Lists

An unordered list is simply a list of related items whose order does not matter. Creating an unordered list in HTML is accomplished using the unordered list block-level element, $< ul >$. Each item within an unordered list is individually marked up using the list item element, $< li >$.

By default, most browsers add a vertical margin and left padding to the $< ul >$ element and precede each $< li >$ element with a solid dot. This solid dot is called the list item marker, and it can be changed using CSS.

```
<ul>
  <li>Orange</li>
  <li>Green</li>
  <li>Blue</li>
</ul>
```

3.6.2 Ordered Lists

The main difference between an ordered list and an unordered list is that with an ordered list, the order in which items are presented is important. Because the order matters, instead of using a dot as the default list item marker, an ordered list uses numbers.

```
<ol>
  <li>Open a text editor</li>
  <li>Type the code in html</li>
  <li>Save the file using .html extension</li>
</ol>
```

Ordered lists also have unique attributes available to them

Start attribute: The start attribute defines the number from which an ordered list should start. The start attribute accepts only integer values, even though ordered lists may use different numbering systems, such as roman numerals.

Reversed Attribute: The reversed attribute, when used on the $< ol >$ element, allows a list to appear in reverse order. An ordered list of five items numbered 1 to 5 may be reversed and ordered from 5 to 1.

Value Attribute: The value attribute may be used on an individual $< li >$ element within an ordered list to change its value within the list. The number of any list item appearing below a list item with a value attribute will be recalculated accordingly.

3.6.3 Description Lists

Description lists are used to outline multiple terms and their descriptions, as in a glossary

Creating a description list in HTML is accomplished using the description list block-level element, $< dl >$. Instead of using a $< li >$ element to mark up list items, the description list requires two block-level elements: the description term element, $< dt >$, and the description element, $< dd >$.

3.7 Adding Media

HTML provides ways to embed rich media in the form of images, audio tracks, and videos, as well as to embed content from another web page in the form of an inline frame.

3.7.1 Adding Images

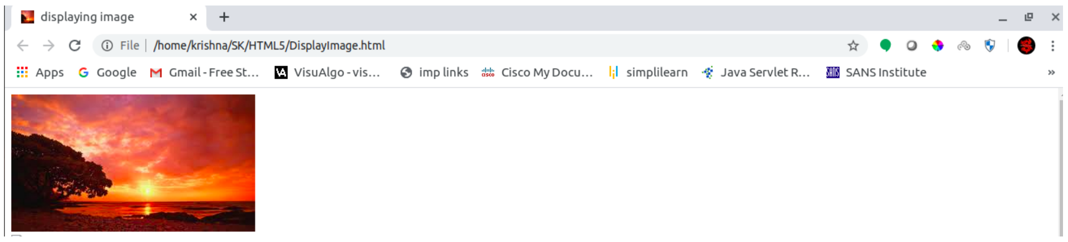
To add images to a page, we use the $< img >$ inline element. The $< img >$ element is a self-containing, or empty, element, which means that it doesn't wrap any other content and it exists as a single tag. For the $< img >$ element to work, a src attribute and value must be included to specify the source of the image. The src attribute value is a URL, typically relative to the server where a website is hosted.

- In conjunction with the src attribute, the alt (alternative text) attribute, which describes the contents of an image, this is displayed incase of any errors in displaying the image
- *title* attribute of the img tag, allows the user to decide what text is to be displayed when the cursor hovers over the image
- By default, the web browser loads the image in its original size as found in the path specified
- To add more details to the image, the $< img >$ tag can be included under the $< figure >$ tag, which allows additional functionalities like adding caption to images

3.7.2 Adding favicons

A favicon is a small, iconic image that represents your website. Favicons are most often found in the address bar of your web browser. These must go in the $< head >$ section

```
<link rel="icon" type="image/png" href="Firefox_wallpaper.png" />
```



It can be observed that a small image appears before the title.
NOTE: Only special icon images or .png images can be used for logos.

3.7.3 Adding videos

The $< video >$ tag is used to add videos to the webpage.
Attributes:

1. height, width : size of the window to display the video
2. autoplay: to play video as soon as you load the page
3. loop: to play the video in a loop
4. controls: to give user control of when to start and stop the video
5. poster: allows us to specify an image, in the form of a URL, to be shown before a video is played.

3.7.4 Adding audios

The $< audio >$ tag is used to add videos to the webpage.

3.8 Organizing data with Tables

Tables are made up of data that is contained within columns and rows, and HTML supplies several different elements for defining and structuring these items. At a minimum a table must consist of $< table >$, $< tr >$ (table row), and $< td >$ (table data) elements. For greater structure and additional semantic value, tables may include the $< th >$ (table header)

Table: We use the $< table >$ element to initialize a table on a page. Using the $< table >$ element signifies that the information within this element will be tabular data displayed in the necessary columns and rows.

Table Header: To designate a heading for a column or row of cells, the table header element, $< th >$, should be used. The $< th >$ element works just like the $< td >$ element in that it creates a cell for data. The value to using the $< th >$ element over the $< td >$ element is that the table header element provides semantic value by signifying that the data within the cell is a heading, while the $< td >$ element only represents a generic piece of data.

Table Caption: The $< caption >$ element provides a caption or title for a table. A caption will help users identify what the table pertains to and what data they can expect to find within it. The $< caption >$ element must come immediately after the opening $< table >$ tag, and it is positioned at the top of a table by default.

Table Head, Body, & Foot: The table head element, $< thead >$, wraps the heading row or rows of a table to denote the head. The table head should be placed at the top of a table, after any ¡caption¿ element and before any $< tbody >$ element.

After the table head may come either the $< tbody >$ or $< tfoot >$ elements. Originally the $< tfoot >$ element had to come immediately after the $< thead >$ element, but HTML5 has provided leeway here. These elements may now occur in any order so long as they are never parent elements of one another. The $< tbody >$ element should contain the primary data within a table, while the $< tfoot >$ element contains data that outlines the contents of a table.

rowspan & colspan: These attributes can be used to customize the height and width of the cells in the table

border: This attribute of $< table >$ can be used to decide the thickness of the table border

3.9 Other tags

There are some other tags which are very frequently used

3.9.1 Horizontal rule

The $< hr >$ tag defines a thematic break in an HTML page by drawing a line.

3.9.2 address

The $< address >$ tag defines the contact information for the author/owner of a document or an article. The contact information can be an email address, URL, physical address, phone number, social media handle, etc.

3.9.3 blockquote

The < *blockquote* > tag specifies a section that is quoted from another source. A < *cite* > tag can be used to mention the person who said that.

3.9.4 summary

The < *summary* > tag defines a visible heading for the < *details* > element. The heading can be clicked to view/hide the details.

Note:

- The < *summary* > element should be the first child element of the < *details* > element.
- Does not work on Firefox, and by defult it will be open

3.9.5 Abbreviation

The < *abbr* > tag will display an abrreviation on the browser. If the title attribute is specified in the tag, the title will appear when the mouse hovers over the abbreviation.

3.9.6 code

This is an inline tag. The < *code* > element tends to be displayed in a different font. It is similar to using *verbatim* on LaTex.

3.9.7 Subscript and Superscript

The < *sub* > and < *sup* > tags can be used to display text in subscript or superscript.

3.10 More attributes

- **class:** applies special properties to groups of elements
- **id:** specifies a unique id to one element on the page
- **style:** specifies a certain visual style
- **accesskey:** a shortcut key to activate an element
- **tabindex:** the order elements will come into focus using the tab key

SPECIAL ENTITIES

If you want..	Then use...
<	&lt;
>	&gt;
©	&copy;
blank space	&nbsp;
&	&amp;

4 Accessibility

Web accessibility is making the web accessible for the widest possible audience, which includes Temporarily Able-Bodied users (TABs). So the best way to accomplish accessibility is to adhere to standards.

These standards are given by **W3C Web Content Accessibility Guidelines (W3C WCAG 2.0)**. These guidelines are principle based and they are

- Perceivable
- Operable
- Understandable
- Robust

To validate the accessibility of a webpage you can use the following link

`https://wave.webaim.org`

5 Validation

To validate (check for errors) your HTML file you can use the following link

`https://validator.w3.org`

There are 3 approaches to validate your html code

1. Validate by URL
2. Validate by Filename

3. Validate by Direct input

Validate by direct input is where you enter your entire code and then check for possible errors in the code.

NOTE: The validator will find most of the errors that break the rules of the DOM structure. So it is possible that it may miss some errors.

6 Hosting your Website

There are two main requirements to host your website

1. Domain name
2. Hosting company

Hosting company/service act as the server which allows people worldwide to access your files

6.1 Domain Name

Domain name is the name given to your website (url). The domain name by itself is useless unless you have a server that can allow other users access files.

6.2 Hosting

A registered IP address is required to connect with your domain name, this is the unique address given to your files which can be used by people worldwide to access it.

6.3 CPanel

This software provides a graphical interface and automation tools designed to simplify the process of hosting a web site to the website owner or the "end user". To use Cpanel all you have to do is add *cpanel* to the url of your website

$$< \textit{your website url} >/\text{cpanel}$$

The hosting service gives provides the required credentials to login to the cpanel account.

6.4 SSH File Transfer Protocol (SFTP)

SFTP is a popular method for securely transferring files over remote systems (server/ someone else's pc). Requirements for SFTP

1. FTP Client
   This is a software used to connect between different machines. Ex: WinSCP (for PC), Fugu/Cyberduck (for MAC)
2. Find the ftp address for your host

Part II

CSS3

7 Introduction

CSS stands for Cascading Style Sheets, a method of styling the HTML documents with various colours, fonts, layouts and spacing. HTML will always represent content, and CSS will always represent the appearance of that content.

NOTE: Browsers may have different default styles.

7.1 Basic Terminology

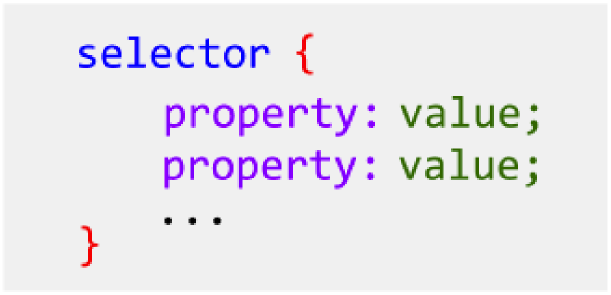
Similar to tags, elements and attributes some of the basic terminologies used with CSS are:

1. **Selectors:**
   A selector specifies to which element/elements within our HTML to target and apply styles. Selectors generally target an attribute value, such as an *id* or *class* value, or target the type of element, such as <h1> or <p>

   Selectors are followed by {}, which contains the styles to be applied to the selected element.

2. **Properties:**
   Once the element is selected, a property determines the styles that will be applied to that element. Property name fall after a selector within the curly braces, and immediately preceding a colon.

3. **Values:**
   The behaviour of a property can be determined with a value. These can be identified as the text between the colon and the semi colon



```
selector {
    property: value;
    property: value;
    ...
}
```

7.2 Types of Selectors

The three most common type of selectors are: *type*, *class* and *ID* selectors

7.2.1 Type Selectors

Type selectors target elements by their element type.
For example, to target all division elements, we can use a type selector of *div*

```
    div { ... }
```

7.2.2 Class Selectors

Class selectors target elements based on their *class* attribute value. Class selectors are a little more specific than type selectors, as they select a particular group of elements rather than all elements of one type. Class selectors allow us to apply the same styles to different elements by using the same class attribute.

In CSS, classes are denoted by a leading period ., followed by the *class* attribute value.

```
CSS
    .highlight { ... }

 HTML
 <div class="highlight"> ... </div>
 <p class="highlight"> ... </p>
```

With the above the syntax, all the elements with the class attribute "highlight" are selcted for styling.

7.2.3 ID Selectors

ID selectors are even more precise than class selectors, as they target only one unique element at a time. ID selectors use an element's *id* attribute value as a selector.

8 Styling with CSS

There are two ways in which we can style the webpage using CSS. They are:

1. Internal Style sheet
2. External Style sheet

Styling using only HTML

```
 <!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title>Styling with HTML</title>
</head>

<body>
<h1 style = "color:blue">Blue color heading</h1>
</body>
</html>

OUTPUT
The heading will be displayed in blue colour
```

8.1 Internal Style sheet

An internal style sheet holds the CSS rules for the page in the head section within the *style* tag. The rules specified are applicable only to that page.

```
<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title>Internal Styling with CSS</title>
<style>
h1{
    color: blue;
}
</style>
</head>

<body>
<h1>Blue color heading</h1>
</body>
</html>

OUTPUT
The heading will be displayed in blue colour
```

8.2 External Style sheet

An external style sheet is a standalone *.css* file that specifies all the styling rules for a webpage. In order to use the same styling for a set of webpages, with Internal Style sheet it results in a lot of repetitive code, so to avoid that we can mention all the styling rules in a separate *.css* file and include this file in all the webpages where we need the same styling.

```
myh1Styling.css

h1{
color: blue;
background-color: yellow;
}

EStylingwithCSS.html

<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title>External Styling with CSS</title>
<link rel="stylesheet" href="myh1Styling.css">
</head>

<body>
<h1>Blue color heading</h1>
</body>
</html>
```

9 Precedence of Styling

Every browser will have certain default styling this gets the lowest priority.

- The browser then checks for **external style sheets**
- Irrespective of finding an external style sheet or not, the browser then checks for **internal style sheets**. And overrides rules from external style sheet with those found in internal style sheet for similar class,id.
- **Inline Styling** has the highest precedence, and overrides any rules mentioned in internal and external style sheet.
- If the styling rules for an element are mentioned in more than one external style sheet. Then rules mentioned in the most recent file have precedence. i.e. The last file listed in the head section with the link tag
- If one selector has more than one rule, then the most recent rule overrides any previous rules that was used for that element.

```
    myh1Styling.css

    h1{
        color: red;
        background-color: yellow;
    }
    h1{
        background-color: orange;
    }

<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title>External Styling with CSS</title>
<link rel="stylesheet" href="myh1Styling.css">
</head>

<body>
<h1>Blue color heading</h1>
</body>
</html>

OUTPUT
The output text will have a font color = red, background-color = orange.
```

To avoid confusion as to which styling to be used when multiple styling
are specified, the !*important* attribute can be used.

```
h1{
    color: red;
    background-color: yellow  !important;
}
h1{
    background-color: orange;
}

OUTPUT
The text will be displayed with a yellow background color as it
is specified as important it wont be overwritten by the following rules.
```

10 Using colors for styling

The three main color conventions that are used for styling are:

1. Color names

2. Hexadecimal
3. RGB value

NOTE: Try to avoid using direct color names, as the colour shades are slightly different on different browsers.

Hexadecimal value is widely used for specifying colors, but lately RGB method is gaining more prominence because of the new feature available with the RGB method **(RGBA)** where A stands for *alphatransparency*

10.1 Color contrast

There are tools that quantify the contrast between text and background

`http://wave.webaim.org`

`http://webaim.org/resources/contrastchecker`

NOTE: Dont use color alone to convey meaning

11 Styling text

There are various options available for styling text

- font (family, style, variant, size)
- color and background
- alignment
- line-height

11.1 FONT

11.1.1 font-family

These are styles of text
Ex:

- Comic Sans MS
- Verdana
- Arial
- Times New Roman
- Courier
- Impact

Syntax:

```
h1{
    font-family: Arial;
}
```

NOTE: Not all font-families are supported by all of the operating systems, so it is always a good practice to provide alternatives.

```
h1{
    font-family: Courier, Impact, Arial;
}
```

The above style of specifying rules instructs the browser to use one of the three mentioned font-families whichever is compatible with the operating system.

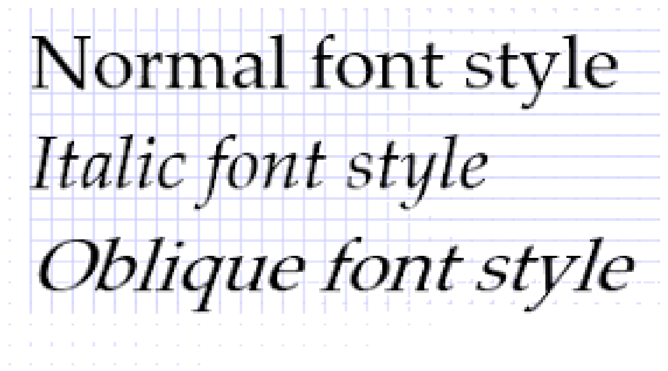
11.1.2 Custom Fonts

In order to use a font-family that is customized according to your needs use the following syntax

```
@font-face{
    font-family: mySpecialFont;
    src: url('sai.ttf');
}

h1{
    font-family: mySpecialFont;
}
```

11.1.3 font-style

- Normal
- Italic
- oblique



11.1.4 font-variant

- normal
- small-caps

11.1.5 font-size

- xx-small, x-small, small, smaller
- medium
- larger, x-large, xx-large, large
- pixel (Widely used)
- percentage

11.2 color and background

```
h1{
    color: #0000FF;                 /* Blue */
    background-color: #B3B3B3;      /* Grey */
}
```

11.3 Alignment

- left
- right
- center
- justify

11.4 Line height

It does not affect the font, it adjusts the space between the lines of text.

```
h1{
    line-height: 150%
}
```

If the value given is less than 100% then the lines will overlap

12 Display

Every item in the webpage is a *box*, using **display** we can decide how multiple boxes should appear (next to each other or one below the other)

- **inline:** Boxes sit next to each other
  - takes up *just enough* width and height
- **block:** Boxes sit one below the other
  - *default* : take up all horizontal width and *just enough* height
  - rules can set height and width
- **inline-block:**
  - Same as inline, but accepts height and width
- **none:** removed from page

12.1 Complimentary properties

- **float:**
  - Reposition elements to the right or left
  - Elements are aware of one another and will not overlap
  - Values: *left*, *right*
- **clear:**
  - Used to keep floating elements away
  - Values: *left*, *right*, *both*

  Usage: clear: left;
  Makes sure that there is nothing floating to the left

12.2 Element overflow

When the height/width used for a particular element is not sufficient to display all of the content it was to supposed to display use *overflow* to determine access

- **visible:** Can cause text to show up "on top" of oher text
- **hidden:** Hides anything that goes beyond the bounding box
- **scroll:** Gives horizontal and vertical scrollbars
- **auto:** Adds scrollbars as needed

12.3 Other display Values

These are the newer features and all the browsers may not support these

- **Table**

  To have a table-like layout without using the table structure,
  use *display* : *table* along with *display* : *table* − *cell* for elements
  Usage:

  ```
  body{
      display: table;
  }
  div{
      width: 30%;
      display: table-cell;
  }
  ```

  NOTE: Do not forget to add display type as *table* for body

- **Grid**
- **Flexbox**

12.4 Visibility

Specifies whether or not element is visible. Options include:

- visible
- hidden
- collapse (only for table elements)

With *display* : *none* the browser does not even indicate the presence of that element, but with *visibility* : *hidden* the browser shows blank space corresponding to the region where the particular element was supposed to appear.

13 Box Model

- Inline elements have fixed height and width
- For block, inline-block elements we can give the desired height and width

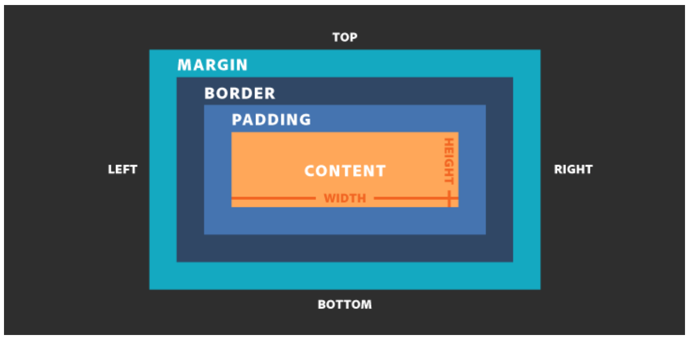
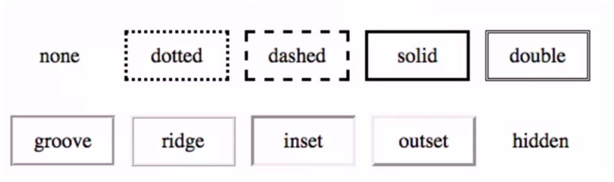


Figure 1: CSS Box model

13.1 Borders

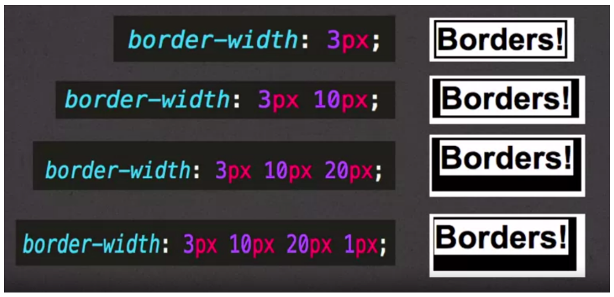
Any elements can have a border around it.

- Border properties specifies style, width and color
- Border style must be specified, width and color are optional parameters. Few of the available styles are
  - none
  - dotted
  - dashed
  - solid
  - double
  - groove
  - ridge
  - inset
  - outset
  - hidden



```
div{
    border: solid 1px #CC00AA;
}
```

- The width can be set by specifying the number of pixels or using keywords like thin, medium or large
- The color can be set by using the color name/RGB value/Hex value/transparent
- Individual borders
  - 2 values specified:
    1 → top,bottom
    2 → left,right
  - 3 values specified:
    1 → top
    2 → left,right
    3 → bottom
  - 4 values specified:
    1 → top
    2 → right
    3 → bottom
    4 → left



```
border-width: 3px;
border-width: 3px 10px;
border-width: 3px 10px 20px;
border-width: 3px 10px 20px 1px;
```

13.2 Margins

This is the additional space outside the border, between the current element and the neighbour

- Positive margin - element moves right/down
- Negative margin - element moves left/up
- Color cannot be given to this, it is by default transparent
- It is possible to have individual margins just like borders

13.2.1 Centering an element

To horizontally center an element use

```
margin: 0 auto;
```

It can be used only when

- The element must display block
- The element must not float
- The element must have a width that is not auto
- The element must not have a fixed or absolute position

13.3 Padding

This is the additional space between the element and its border.

- Positive padding - border moves outward from element
- Color cannot be given to this, it is by default transparent
- It is possible to have individual padding just like borders

  actual width = margin + border + padding + width of element

13.4 box-sizing

Available options:

- content-box
  We have to pass the actual width (Considering border, padding, margin)
- border-box
  It automatically calculates the width considering padding, borders. It wont take into account margin

13.5 Measurements

To set the size of the element

- **Absolute:** to set a fixed size

  px, mm, cm, pt
- **Fluid:** sets size relative to surrounding elements
  - %, vw (viewport width), vh (viewport height)
  - (for font) em: .75 means 75% of current size
  - (for font) rem: .75 means 75% of root element

14 Styling Links and Lists

The links can take on all of the usual styles as well as **text-decoration**. Usually when styling links designers make their links look like buttons.

14.1 States of a link

Normally we would have observed that some links are blue in colour and some other are purple in colour, this indicates the state of the link.

A link which has been previously visited appears purple in colour, and if we have not visited the link it appears blue in colour by default.

- a: link - a normal, unvisited link
- a:visited - visited link
- a:hover - activated by mouse (does not work with touchscreen devices)
- a:focus - activated with the keyboard
- a:active - is being clicked

14.1.1 Precedence rules

- a:hover must come after a:link and a:visited
- a:active must come after a:hover

NOTE: a:link is used to style unvisited links

14.2 Styling lists

There are a whole range of styling options available for lists

- list-style-type
  - lower-roman
  - upper-roman
  - decimal
  - decimal-leading-zero
  - upper-alpha
  - lower-alpha
  - hebrew
  - armenian
- list-style-image

  For unordered lists instead of traditional marker we can use a custom image

```
ul{
    list-style-image: square url('icon.gif');
}
```

```
In the above, square icon is used if the icon.gif file is not found
```

- list-style-position
- list-style

There are several online platforms that help you with CSS coding

`http://css3generator.com`

15 Advanced selectors

With the type selectors that was used till now, it was possible to give only one type of styling for a given tag. But if we want to use multiple different styling for the same tag based on where it occurs, then we need these advanced selectors (ID selectors or Class selectors)

- ID selectors apply the styling only to that particular element which has the same ID, and note that multiple elements cannot have same ID.
  # is used along with ID for styling the element
- Class selectors apply the styling to a range of elements unlike ID selectors that apply styling to a single element.
  . is used along with classname for styling the name
- **Universal:** Applies styling to every element on the page

```
*{
    property: value;
}
```

- **Attribute selectors:** To search the DOM for certain elements that have an attribute you are looking for. Ex:
  - All images that use gif files
  - All images that have empty alt text
  - All links that go to govt sites

  Operators are used to achieve the above

  - ˆ : match the beginning exactly
  - $: match the end exactly
  - : match in any part
  - ˜: Spaced selector

– $\| : Hyphenatedselector$

```
a[href$=‘‘.gov’’]{
                    property: value;
                }
```

- **Child selectors:** Provide a way to select elements that fall within one another, thus making them children of their parent element. These selections can be made two different ways
  - Descendant selector

    ```
    <article>
    <h2> Hi</h2>
    <h2> Hello </h2></article>
    <h2> outside article </h2>

    article h2{
                background: #343434;
                }
    ```

    In the above code styling is applied only to h2 that are within the article tag.
  - direct child selector

    To select only the direct children of a parent element and not every instance of the element nested deeply inside of an ancestor.

    ```
    <p> ... </p>
    <article>
    <p> Only this is selected </p>
    <div>
        <p> ... </p>
    </div>
    </article>

    article > p { ... }
    ```

    In the above example though there are two <p> within article only one of them is the direct child (the one that contains text) and styling is applied to only this element
- **Sibling selectors:** To select all the elements that share the same common parent.

  ```
  <p>..</p>
  <section>
      <p> .. </p>
      <h2>..</h2>
      <p>This will be selected</p>
      <div>
      <p>...</p>
      </div>
  ```

```
<p> This will be selected </p>
</section>

h2 ~ p { ... }
```

h2   p is sibling selector that looks for *p* elements that follow and share the same *h*2 parent

- **Pseudo-Classes**
- **Pseudo-Elements**

16 Website Background

Backgrounds have a significant impact on the design of the website.

16.1 Transparent background

The RGBa color scheme can be used to set the transparency levels. But all browsers do not support this styling so it is better to provide a fallback color which is used in case the RGBa coloring is not supported.

```
body{
        background-color: #b22bb2;  /*fallback color */
        background-color: rgba(0,0,0,.3);
    }
```

16.2 Background Images

Using background images enhances the look of the website but there is a lot of formatting to be done before we get the background image to perfectly fit with our website.
Default Settings

- If the size of the image does not match with the size of the website, then it will repeat horizontally and vertically from the top-left of the given element.
- Positioned at top-left corner of the element

16.2.1 background-repeat

This property can be used to change the direction in which a background image should repeat, if it is supposed to repeat. This property accepts 4 different values

- repeat

- repeat-x
- repeat-y
- no-repeat

16.2.2 background-position

This property can be used to control exactly where the background image is placed relative to that corner. This property requires two values $horizontal\ offset$ and $vertical\ offset$.

If only one value is provided it is considered as horizontal offset, and vertical offset is set to 50% by default. To set the value for this property the following can be used

- keywords like top, right, left, center
- pixels
- percentages
- any length measurement

16.2.3 background-attachment

To have a fixed background image even when scrolling through a webpage this property can be used.

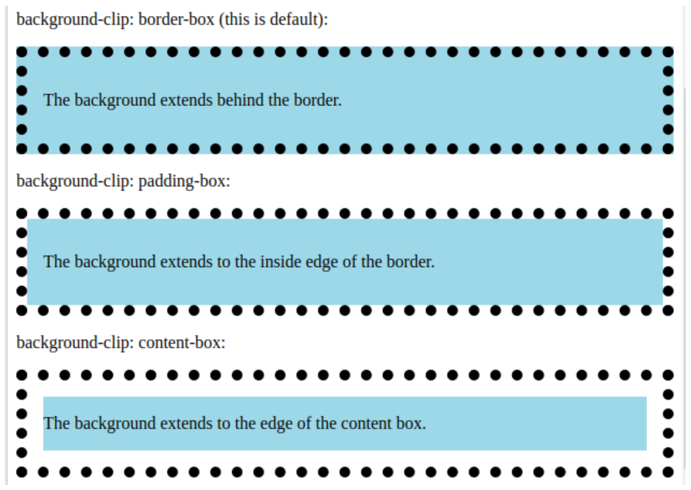
16.2.4 background-blend-mode

This property can be used to define the blending mode for the background image, the possible values for this property are

- normal
- multiply
- screen
- overlay
- darken
- lighten
- color-dodge
- saturation
- color
- luminosity

16.2.5 background-clip

This property can be used to define how far the background (color or image) should extend within an element.


background-clip: border-box (this is default):

The background extends behind the border.

background-clip: padding-box:

The background extends to the inside edge of the border.

background-clip: content-box:

The background extends to the edge of the content box.

16.3 Short hand rules

To pass multiple values to a single property.
Ex:

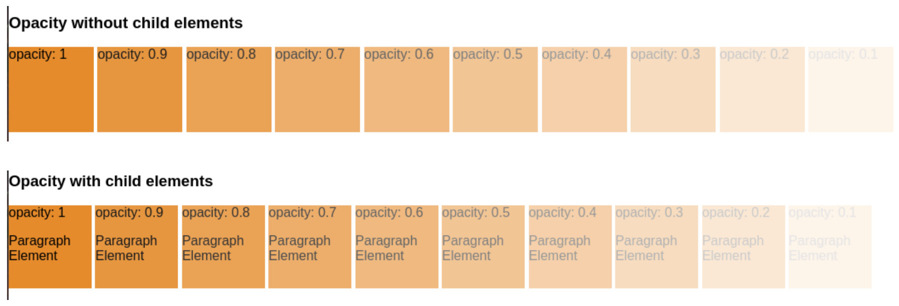
```
body{
        background-color: #00ff00;
        background-image: url("imgs/ocean.jpg");
        background-repeat: no-repeat;
        background-position: fixed center;
    }

body{
    background: #00ff00 url("imgs/ocean.jpg") no-repeat fixed center;
    }
```

Both the above mentioned rules do the same styling.

16.4 Opacity

This property defines how transparent an element is.Values are a number from 0 to 1 representing the opacity of the channel
Syntax:

```
    div{
        opacity: 0.5;
    }
```



17 Handling Unsupported Properties

Not all browsers support all CSS3 properties, a quick fix for this is to use **Browser prefixes**. Using these Browser prefixes we can style the content of the page based on the browser capabilities that the person is using. Few of the standard Browser Prefixes are

- **-webkit-**: Android, Chrome, iOS, Safari
- **-moz-**: Firefox
- **-ms-**: Internet Explorer
- **-o-**: Opera

17.1 Frequently Unsupported properties

- column-count: To add multiple divisions side by side.
- border-radius: Adds a rounded edge to the corners
- gradient: Allows background color to go from one shade of a colour to another colour with a smooth transitioning

`http://caniuse.com`

The above link will tell you when you need to use prefixes.

18 Pseudo-Classes

A pseudo-class is used to define a special state of an element. Pseudo-elements need not necessarily be a part of DOM.
Various types of Pseudo-classes are:

- Link
  - :link
  - :visited
- User Action
  - :hover
  - :active
  - :focus
- Forms (interfaces)
  - :enabled
  - :checked
  - :disabled
- Styling based on structural position of elements
  - :first-child
  - :last-child
  - :nth-child()
  - :only-child
  - first-of-type
  - last-of-type
  - only-of-type
- Textual
  - :first-letter
  - :first-line
- Generate
  - :before
  - :after
- Fragments
  - ::selection

19 Styling Transitions

When elements transition from one state to another, you can later their appearance. The properties that can be altered during a transition are

- transition-property: size, color, position etc that is to be changed during a transition
- transition-duration: How long should the transition take
- transitiion-timing: Should the transition be smooth or different
- transition-delay: How long should user wait before transition begins.

```
div{
    color: #000000;
    background: #2db34a;
    line-height: 200px;
    text-align: center;
    width: 250px;
    height: 200px;
    border-radius: 6px;
    transition-property: color, width, background, border-radius;
    transition-duration: .5s;
    transition-timing-function: linear;
    transition-delay: .5s;
}

div:hover{
        color: #ffffff;
        width: 350px;
        background: #2d3455;
        border-radius: 50%;
}
```

Using shorthands

```
transition: background .2s linear, border-radius 1s ease-in 1s;
```

In the above code, it says the change in background should take 0.2 seconds and the transition should be linear, the border radius changing should take 1 second and the transition should ease-in. The delay before all these transitions should be 1 second

20 Transforms

The transform property applies a 2D or 3D transformation to an element. The available 2D Transform options are

- transform:translate(x,y);
- transform:rotate(deg);
  do not forget to mention **deg** after sprcifying the numerical value
- transform:scale(width,height);
- transform:skew(x-angle, y-angle);
- matrix()

  Combines allof the 2D transform methods into 1

The 3D transform options available are

- transform:rotateY(deg)
- transform:rotateX(deg)
- transform:rotateZ(deg)
- transform:rotate3d(x,y,z)

21 Positioning elements in the webpage

The four position properties are:

- static
- relative
- absolute
- fixed

21.1 Static

This is the default value. It places the element in the next available position

21.2 Relative

- Positioned "relative to itself"
- Take the static position, but add offsets
- The new positioning does not affect any other element
- Relatively positioned elements are often used as container blocks for other elements

21.3 Absolute

Places all the elements at the specified location, even if they overlap.

21.4 Fixed position

The position of the element is fixed relative to the browser window, it will not move even if the window is scrolled.

21.5 Z-index

Multiple elements may be placed in the same position. The z-index property specifies the stack order of an element. An element with greater stack order is always in front of an element with a lower stack order.

22 Navigation menus

We can have horizontal or vertical navigation menus based on the website layout. In case of vertical navigation menus we would want that section to occupy the entire screen height. To do this we can use **viewport height** option to size the navigation section such that it fills the screen height

```
    height: 100vh;
```

Will ensure that the element fills the entire height of the screen. Also note that nav links are block by default to have vertical navigation menu, we have to make nav inline-block. **NOTE:** Line-height same as text-size will center the text.

Part III

JavaScript

Every webpage is represented by the **DOM** structure. Scripting languages like JavaScript use the DOM to interact with the various elements of the documents and modify their attributes. Accessing the DOM is done with an **Application Programming Interface (API)**.

The DOM elements

- **document:** The root of the page
- **element:** A node in the DOM tree
- **nodeList:** An array of elements
- **attribute:** A particular property of node in the DOM

APIs can be used to access elements either by their **id** or **class names**

- document.getElementById(*id*)
- document.getElementsByClassName(*class*)
- element.innerHTML
- element.style
- element.setAttribute(attribute, value)
- element.removeAttribute(attribute)

The main feature that a webpage gets by using JavaScript along with HTML and CSS is **interactivity**. The other things that can be achieved with JavaScript are:

∗ Read and write HTML elements

∗ React to events (keyboard events and mouse events)

∗ Validate data

∗ Detect visitor's browser

∗ Create cookies

JavaScript does not have an built-in print function, so to display data the following can be used

– an alert box using **window.alert()**

– a prompt using **window.prompt()**

– HTML output using **document.write()**

– HTML element using **innerHTML()**

– Browser console using **console.log()**

22.0.1 Creating Alert boxes

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>First javascript</title>
</head>
<body>
    <script>
    alert("This is a pop up alert box");
    </script>

    <noscript>
    Your browser does not support or has disabled JavaScript
    </noscript>
</body>
</html>
```

Note that the tag **script** is used to indicate that the following contents are not just HTML.

22.0.2 Prompts

This is similar to alert but requires input

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Prompt</title>
</head>
<body>
    <script>
    prompt("Enter your name");
    </script>

    <noscript>
    Your browser does not support or has disabled JavaScript
    </noscript>
</body>
</html>
```

22.0.3 Display output

document.write() can be used to display the output. Note that even HTML tags can be used to modify the output

```
document.write("<h1>Hello World</h1>");
```

document.write is not a preferred method to display output, as there are chances that it may overwrite other contents of the page.

Another preferred method to display output is by using **innerHTML**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Inner HTML</title>
</head>
<body>
    <h1 id="test">Tester</h1>
    <script>
    document.getElementById('test').innerHTML = "Hello World";
    </script>
</body>
</html>
```

Though h1 is "Tester". The output displayed is Hello World. This is because the value of the element with id = 'test' has been changed to "Hello World".

NOTE: If there are multiple elements with the same id, then only the first element with that id will be affected while using document.getElementById()

22.0.4 Writing to the browser console

console.log() can be used to write contents to the console. The console is a place to see what is going on during the execution of your program. The console also provides debugging information for JavaScript, HTML and CSS.

22.1 Variables

Data can be stored in Variables. In order to use a variable it should be declared first. The syntax to declare a variable is

```
var name;
```

The above line of code declares a variable called name. **var** is the keyword used to declare variables.

Rules to be followed while giving names to variables

- Consists of letters, digits, underscores and dollar signs
- Cannot start with a digit
- Case sensitive
- Should be meaningful

22.1.1 Data Types

Usually in programming languages the variables should be of a single type (i.e. only one type of values can be assigned to them), but in case of JavaScript the variables can take on many different types. The different types of data are

1. Number
2. String
3. Boolean
4. Object: Any node in the DOM
5. Array

22.1.2 Operators

The various operators supported by JavaScript are

1. Arithmetic operators (+, -, *, /, %)
2. Increment and decrement operators
3. Short hand operators
4. Boolean operators (==, !=, >, <, <=, >=)
5. Equality with type (===, !==)
   x = 12 x == "12" Returns true
   x === "12" Returns false (as the types dont match)
6. Logical operators (&&, ||, !)

22.2 Functions

Functions are bits of codes that can be reused, by repeatedly calling the function in the required place in our code. The syntax for defining a function is

```
function functionName(parameters){
    <code to  be run >
}
```

22.3 Where to place JavaScript code

JavaScript code can be placed in the body, head or in an external file. While using within head/body the script tag must be used.

Putting the JavaScript code in the head section means putting only the javascript functions in the head section. The actual code will still be part of the body.

If the javascript is placed in an external file, script tag need not be used.

```
<script src = "<path to js file>" ></script>
```

23 Events

Functions can be called from the JavaScript code or when particular events occur. The **JavaScript API** lets us add dynamic function calls. The various events that can trigger a function call are

1. Mouse Events:
   ondblclick, *onmousedown*, *onmouseenter*, *onmouseleave*, *onmouseout*
2. Keyboard Events:
   onkeydown, *onkeypress*, *onkeyup*
3. Frame Events:
   onload, *onresize*, *onscroll*, *onerror*

Few very frequently used ones are:

- **onclick**: User clicks on an HTML element
- **onmouseover**: User moves the mouse over an HTML element
- **onresize**: Browser window is resized
- **onload**: Browser finishes loading the page

The syntax to be followed to link JavaScript functions with HTML tags is as follows

```
<HTML element <event> = "jsFunction()"> ..... </HTML element>
```

NOTE: Events change the program flow

"this" keyword: allows an element to reference itself.

24 Arrays

If we want to change a particular attribute of all the elements of a certain type on the webpage, it is not feasible to go to every occurrence of that element and change the property, for tasks such as this we have commands like *getElementsByTagName*, *getElementsByClassName*. These commands can return one or many items, this is where arrays come into picture.
Syntax for array declaration

```
var arrayName = [<array contents>];
```

NOTE:

- The elements in the array dont have to be of the same type.
- JavaScript arrays are Objects and hence they have methods and attributes associated with them. Ex: arrayName.length, arrayName.sort(), arrayName.push(element)

25 Iteration

To access all the elements of an array, we can use looping/iteration.

- for loop is used to repeatedly execute a block of statements fixed number of times.

```
for(j=0;j<array.length;j++)
{
    //Block of code to be repeatedly executed
}
```

26 Flow of control

Used to determine the order in which various blocks of code are executed.

- if

```
if( boolean expression){
    statements;
}
```

- if-else

```
if(boolean expression){
    statements;
}
else
{
    statements;
}
```

- nested if

27 Form Validation

Form validation is a valuable and extremely common way to incorporate JavaScript into the website. The commonly used HTML tags when it comes to creating forms are

- <form>
- <label>
- <input>

```
<form>
<label for='name'>Name</label>
<input type='text' id='name' name='name'><br/>
<input type='submit' value='click here'>
</form>
```

The above field creates a label called name, and a text field where the user can enter his/her name. **name='name'** ensures that the text field is activated even when clicked on the label instead of the text field.

27.1 Attributes of Form elements

27.1.1 name

All all input types have a name attribute. THe name attribute is assigned whatever value is input

27.1.2 id

id attribute is very important for frontend development. Used by labels to connect the associated text entry field.

27.1.3 value

Depends on the input type it is being used with.

- button: text inside button
- textfield: provides a default value (If no value is input for that field, the default value will be considered)

27.1.4 placeholder

Provides a suggestion to the user, on what type of input is expected, in the given textfield. As soon as the user clicks on the textfield the suggestion disappears.

27.2 Input types

- textfield
- email
- password
- radio button
- checkbox
- submit
- number
- range
- color
- date
- url

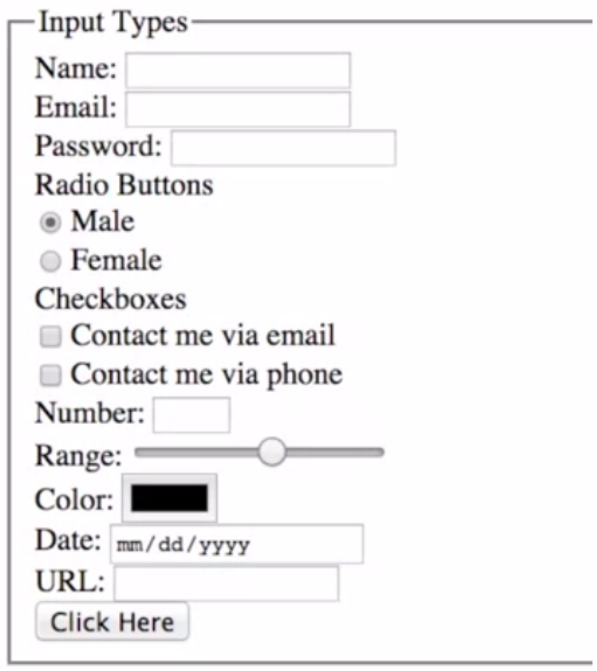


Figure 2: Forms

While using checkbox and radiobutton, the attribute **checked** can be used to get information about which option was selected.

27.3 Validation

The different items that can be validated in a form are:

- The type of input
- The format of the input
  - Is it a valid email address
  - does the phone number have spaces
- The value of the input

To perform validation we can use

- HTML5 input types
- HTML5 attributes
- JavaScript functions

27.3.1 Input types for validation

If using the input types for validation, it requires the browser to validate the format of the input. If the browser does support it, then submission of the form will be halted if there are any non-valid input. IF browser does not support various input types, then the input type is just text.

27.3.2 Input attributes for validation

These attributes give the browser additional information on the type of input that is expected.

- **required**: Submit process is halted if any of the required elements are empty.
- **pattern**: Works with only input type = text, and requires the input to have a specific form.

```
i) [0-9]{5}
     Use 5 numbers between 0-9
ii) [a-zA-Z]+
    Use only lowercase or uppercase characters.
    ``+'' indicates that there should be atleast one character.
iii) [0-9]{13-16}
    Use 13-16 numbers
```

- **limiting number**: min,max and step can place limits on number inputs

Various patterns possible can be found at

`http://html5pattern.com`

NOTE: To ignore all the attributes use **novalidate** inside the form tag. Different browsers have different compatibilities to these features, so it is preferable to write our custom validation using JavaScript.

Part IV

Responsive Design

The idea of designing the website such that we are not limiting the user to only one screen size is called Responsive design. The elements in the website are expected to resize themselves such that they fit the screen size.

Three common approaches to responsive design

- **Responsive Web Design**: Create the website using fluid measurements, flexible grids, varying CSS rules and media queries
  - Same code, but different styling is used for different screen resolution
  - Same URL is used
- **Adaptive Design**: returns one of multiple versions of a page based on the type of the device
  - Different code is used for different screen resolution
  - Same URL is used
- **separate mobile site (.m)**: A separate page URL for the mobile site.
  - Different code is used for different screen resolution
  - Different URLs are used
- Use existing framework (such as **bootstrap**) that does all of the responsive for you.
- Using an hybrid of the above techniques

To test the responsiveness of the website, an online tool is available

`http://ami.responsivedesign.is`

Benefits of Responsive Web Design

1. Easier to share data with a single URL
2. Easier for search engines to index the page
3. Fewer files means less maintenance
4. Lesser redirection means lesser load time

28 Wireframes

Wireframes provide a visual representation of your layout. The desired layout varies based on the device being used to view the website.
NOTE: Mobile view is important. It is a good practice to first design the website for the mobile view.

Using wireframes it is possible to test the interaction of the webpage.

28.1 Breakpoints

Breakpoints are sizes that define a change in the webpage layout or content. These are used to provide best possible experience for users based on deviice information.

Breakpoints should correspond to

1. device being used
2. content

NOTE: It is a standard practice to let the default styling correspond to mobile view (small screens).

29 Fluid Measurements

29.1 Absolute measurements

- px (pixel)
- mm, cm, in
- pt (point = 1/72 of an inch)
- pc (pica = 12 points)

29.2 Relative measurements

- %
- em (font size of upper case M in any typeface)
  - If multiple divisions are nested one inside the other, then using em will result in different font size in each of those divisions, inner we go smaller it gets
- rem (font size of root element)
- vw (viewport width)
  - 1vw = 1% of viewport width

- vh (viewport height)
  - 1vh = 1% of viewport height

$$1\text{em} = 12\text{pt} = 16\text{px} = 100\%$$
$$1\text{in} = 2.54\text{cm} = 72\text{pt} = 6\text{pc}$$

30 Media Queries

Media queries allow the style to depend upon the media properties. CSS allows the user to use different style sheets for different purpose. The media attribute specifies what media/device the CSS style is optimized for.

30.1 Components of a media query

Every media query has 2 components

1. Media type
   Ex: Screen, print, braille, all, etc
2. The actual query of a media feature and "trigger" size.
   Boolean operators can be used to impose more conditions.

30.2 Implementation of media query

There are 3 ways to implement media queries

- @import rule
  Ex:

  ```
  @import url(smallstyle.css) screen and (min-width:600px)
  ```

  Uses the stylesheet smallstyle.css when the screen size is atleast 600px wide.
- Put media query directly in the style sheet.
  Ex:

  ```
  @media screen and (min-width:600px){ ... }
  ```

- Include query in the link
  Ex:

  ```
  <link rel = "stylesheet" media = "screen and
  (min-width:400px)"> and (orientation: portrait)>
  ```

The conditions used with the media queries are called **breakpoints**. These indicate the change of styling for the webpage.

30.3 Using Media queries for Responsive design

The procedure to be followed in order to use media queries for Responsive design are

1. **Grab information**

   ```
   <meta name = 'viewport' content = 'width=device-width, initial-scale=1'>
   ```

   Using the above meta tag informs the browser that styling will change based on the viewport and requests the browser to give information of the viewport and also screen width.
   initial-scale=1 sets the size of the webpage content.

2. **Fluid Layout**
   It is a good practice to avoid absolute measurements even when using breakpoints.
   NOTE: Paddings and margins are affected by width and not height.

3. **Including Media Queries**
   The fluid layout is triggered by certain sizes. Always design for smaller sizes first and then continue to work for the larger ones.
   NOTE: Execution in CSS file takes place from top to bottom, so always place the default rules at the top, and use suitable order for various media queries.

31 Bootstrap

Web frameworks make the job of the developer easy by providing ready made code and structure. Few of the popular frame works are

- Bootstrap
- Foundation by ZURB
- Semantic UI
- Pure by Yahoo!
- Ulkit by YOOtheme

31.1 Introduction

Bootstrap focuses on Responsive Web Design and uses the mobile-first approach. It consists of

- CSS and HTML templates
- JavaScript extensions

The various websites built with bootstrap can be found at

`http://builtwithbootstrap.com`

Benefits of using Bootstrap for designing

- Fast development
- Platform independent
- Responsive by default
- Customizable

31.2 Breakpoints

Bootstrap hardcodes breakpoints for different viewports. With the intention of following mobile-first paradigm.

- The smallest screen size that bootstrap deals with is a width of 0 - 320px. But many browsers work with sizes larger than this. So this breakpoint may never trigger.

The various default sizes used as breakpoints by bootstrap

- **Extra small devices and phones**: Referred to as **xs-**. Devices with width of 480px fall into this category

- **Small devices and Tablets**: Referred to as **sm-**. Devices with width of atleast 768px fall into this category.
- **Medium devices and Desktops**: Referred to as **md-**. Devices with width of atleast 992px fall into this category.
- **Large devices and Wide Screens**: Referred to as **lg-**.Devices with width of atleast 1200px fall into this category.

NOTE: When a user on a smaller screen zooms in, then this changes the size of the viewport and hence may a trigger a styling meant for some other device.

Though bootstrap hard codes breakpoints it is possible to change these default value.
To use bootstrap layouts and code, we can either download a copy of the code or use absolute reference. To save a copy of bootstrap use

`http://getbootstrap.com`

A **content delivery network (CDN)** provides a way to connect to the bootstrap codes using an absolute reference.
Various templates can be found at

`http://getbootstrap.com/getting-started/#examples`

31.3 Bootstrap Grid System

Bootstrap layout is based on a 12 column grid. Every grid consists of

- A container class
    - A row class
        * One or more column classes

Ex:

```
<div class = "container">
    <div class = "row">
        <div class = "col-xx-yy">
```

Where *xx* is the viewport size (xs, sm. md, lg)s and *yy* is the number of columns (0 - 12) that are being used. Many a time all the above options can be combined

```
<img src = "pic.jpg" class = "col-xs-12 col-sm-6 col-md-3 col-lg-2">
```

31.3.1 Positioning classes

By default bootstrap puts elements as far to the left as possible. But it is possible to move them to left/right

col-XX-push-YY

move YY columns to the right

col-XX-pull-YY

move YY columns to the left

31.3.2 Responsive Utility classes

- **hidden-XX**: content will only be hidden on the XX screen size
- **visible-XX**: content will only be visible on the XX screen size.
- **sr-only**: content is hidden on all devices except screen readers.

31.3.3 Navigation bars

There are two types of navigation bar links that bootstrap provides

1. nav-tabs
2. nav-pills

The other layout options available are horizontal, stack, justified, etc.

navbar class serves as a navigation header for the website. The various options available by using navbar are

- navbar-static-top
- navbar-fixed-top
- navbar-fixed-bottom

31.3.4 Drop down menus

To add dropdown menus, you need to include the bootstrap js files and a link to the jQuery.

31.4 Responsive images

Using fluid measurements is one of the easiest way to make images responsive. There are options like **max-width** and **min-width** that can be used to avoid the image becoming extremely large or extremely small. The different classes that bootstrap provides for images are

- img-responsive
- img-rounded
- img-circle
- img-thumbnail

31.5 Responsive tables

Using bootstrap for styling tables requires a styling with many classes on a single element. The *table* class is the foundation.

- .table : adds padding and horizontal dividers
- .table-striped : zebra stripes where "odd" rows have light color
- .table-bordered : add borders to table and cells
- .table-hover : adds hover state styling

Bootstrap allows only one breakpoint for responsive tables. If table size is under 768px it allows horizontal scrolling. If table size is larger then default styling is used.

With bootstrap it is also possible to style smaller elements

- active, success, info
- warning, danger

table-responsive class can be used with the parent element of the table, using this class adds the horizontal scrollbar only for the table and not the entire webpage.

Various free bootstrap templates can be found at

- `http://getbootstrap.com/getting-started/#example`
- `http://www.bootstrapzero.com`
- `http://startbootstrap.com`