# Web Development

Frontend technologies include -
- HTML
- CSS
- Bootstrap
- Java Script
- Jquery

Back end
- Python
- DJango
- Sequel

## Sylkabus

- HTML
- CSS
- Bootstrap
- Java Script
- DOM
- Jquery
- Nodejs
- Postgres SQL

Sample LOD

Virtual environment - make sure that in whatever version of the software or technologies you are using still the final project will run on other devices also.

## platforms

(TE) - Text editer - atom TE or Sublime or Vs code
web browsy - google chrome
anaconda distibution - for creating virtual environments

* Web

The WWW (World Wide Web) is known as web is an information system where documents & other web resources are identified by uniform resources locators which may be interlinked by hypertext, & are accessible over the internet

* How to develop websites

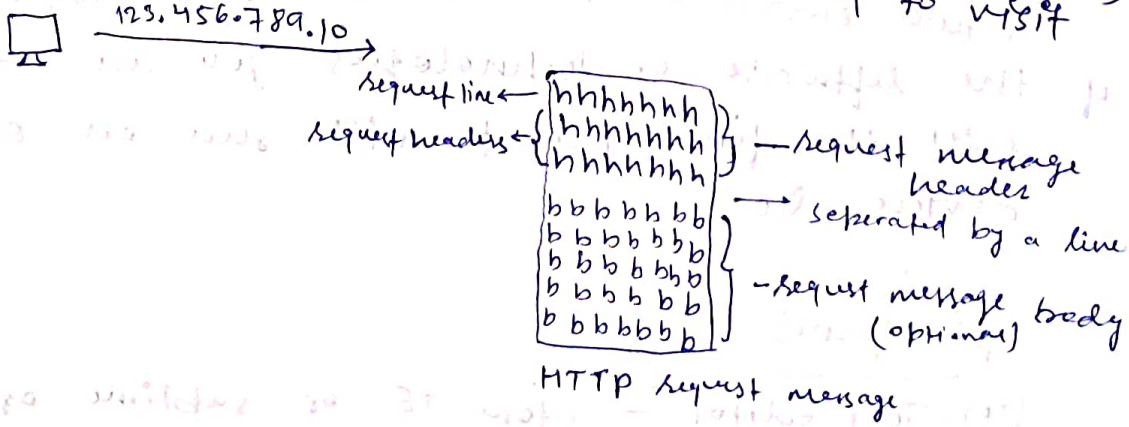
Create a file for each page of the website using html that gives the content of that page. To do this you will typically pay an internet service provider (ISP) for web hosting & move your files to their server using FTP (File transfer protocol)

* websites is invented by Tim Berners-Lee
it ran on NeXT computer
1st page of web address was http://info.cern.ch/hypertext/WWW/TheProject.html

## How websites works

Step 1 – usually one would type the URL of the websites inside a default web browser to visit it

Step-2 – your local machine sends a request in form of a packet which has IP (internet protocol) address of the website you want to visit




Step-3 – Then a computer generally send this request using wires using your ISP (internet service provider)
/
Broadband, wifi etc

Step-4 - your ISP resolves this request to the corresponding server location, using IP address as a guide

* Server
  - it is a device that provides service to other devices
  - in other words, device will request access to an HTML file from our server, and our server will return the HTML file to that device

Step-5 - Once your request reaches the server, it can now send back the website you asked for

Step-6 - but generally a complete website with so much content is very big in size to send as a single packet of data which is a problem

Step-7 - To resolve this issue the server will send back the website which is divided into many small packets of data

Step-8 - These packets come with certain instructions on how to get back to requested end user & finally reassembles to display the websites

Step-9 - These packets dont mind what path they follow to reach the final end user but their objective is just reach the final location of request made

Step-10 - Once these packets reach the final end user, they are reassembled to show the requested web page in correct format

Step-11 - All of this happens at a speed of light (299792458 m/s) so it happens extremely faster

# HTML

Hypertext markup language

↓

link that connect web pages to one another either within a single websites or b/w websites

static

## CSS

* cascading style sheets
* it is actual styling of any websites
* colors, fonts, borders etc

## Java script → dynamic

it is used to build dynamic web applications

* Static websites are those which are fixed and only display the same content for every user usually written exclusively in HTML
* Dynamic is one that can display diffent content & provide user interaction. by making use of advance programming & database in addition to HTML
  * JS makes the websites responsive.
  * It allows you to add interactivity to a websites including the programming logic

Backend of any websites generally has 3 components

① - the language

② - The framework

③ - The database

# Domains of full stack WD

Having domain knowledge is more important than your technical skills while working for a corporate company some domains are:-

* Backend architecture
* Mobile app development
* Backend & framework technologies
* Front end technologies
* Data base technologies
* cloud service
* Version control system(VCS) & debugging
* Devops

## Domain

It is a nothing but core knowledge of any particular sector, specific industry or nature of business.

# HTML

- HTML basics
- Tagging
- Lists
- Divs & Spans
- attributes
- images.

* It is the first ~~step~~ fundamental step to understand how web applications are built

once we open any text editor

# 1st line

<!DOCTYPE html>

# 2nd line

<html lang="en" dir="ltr">

* HTML lang attribute
  - lang attribute of HTML sets the language of an element in HTML document. usage of lang attribute help the browser to display the text in the desired language
  - "en" stands for english language
* HTML dir attribute
  - dir attribute of HTML sets the direction of the text within an element in HTML document
  - The value of the dir attribute is either "ltr" (i.e. left to right) or rtl (i.e. right to left)

# 3rd line

* <head>

```
<meta charset = "Utf-8">
<title></title>
</head>
```

* HTML <meta> charset attribute
  - The charset attribute specifies the character encoding for the HTML document
  - UTF-8 : character encoding for unicode
* HTML <title> Tag
  - <title> defines the title of the document
  - It must be text only shown in the page's tab
  - The <title> tag is required in HTML document

# 4th line

HTML <body> Tag

- <body> tags define the documents body.
- <body> element contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, list, etc.

# HTML Comment tags

- Comments are not displayed by the browsers but they can help document your HTML source code
- Syntax

<!-- Write your comments here -->

## Elements in HTML

* An HTML elements is an individual components of an HTML documents.
* Main parts of elements are :
* The opening tags
  - This states where the elements begins or start to take effect.

    e.g. <P> (paragraph)
* The closing Tag
  - This states where the elements end

    e.g </P> (paragraph)
* The content
  - <P> This is content </P>
* The element
  - The opening, the closing tag & the content together comprise the element

# Basic Tags

* Heading tags are indicators used in HTML to help your structure your webpage & also helping the browsers to read your piece of content
* Heading tags ranges from H1 - H6 & form a hierarchical structure to your page
* H1 is highest & H6 is lowest in size but are heading tags are in bold.

* paragraph tag the HTML <P> element defines a paragraph
* A paragraph always starts on a new line & browsers automatically add some white space before & after a paragraph
* we can put multiple paragraph tags in same line but seperation is created by paragraph tags
* FILTER TAGS Lorem ipsum (used to fill data when required for sample projects)
* BOLD & ITALICS. <b> or  & <i> or 

# HTML List

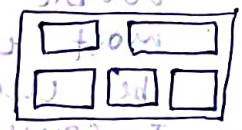
* one way to get all our information organised is by using lists in HTML
* 2-types
  - ordered
  - unordered lists
* ordered
  - an ordered list typically is a numbered list of items
  - HTML gives you the ability to control the sequence number - to continue where the previous list left off, or to start at a particular number
  - The <OL> defines an ordered list
  - an ordered list list can be numerical or alpha
  - The <li> tag is used to define each list item
* unordered
  - an unordered list typically is a bulleted list of items
  - HTML gives you the ability to customize the bullets, to do without bullets & ~~use~~ to wrap list items horizontally or vertically for multicolumn list.
  - To create unordered list use <UL> tag
  - The unordered list starts with <UL> tag

- The list items starts with the <li> tag & will be masked as disc, square, circle etc.
- The default is bullets, which is small black circles

* list inside list is called nesting of lists.

# div Tag

* The  tag defines a division or a section in a HTML document.
* The  tag is used as a container for HTML elements - which is then styled with CSS or manipulated with Java script later
* It doesn't affect the content or layout & is used to group HTML elements to be styled with CSS or manipulated with scripts.



# Span Tag

* The HTML  elements is a generic inline container for phrasing content, which does not inherently represent anything.
* It can be used to group elements for styling purposes (using the class or id attributes), or because they share attribute value such as lang
* The  tag is an inline container used to mark up a part of text, or a part of document.
* The  tag is much like the  element but  is a block-level elements & span is an inline element.


# Attribute in HTML

* In general, attribute is a characteristic
* In HTML it is a characteristic of page element such as font size or color.
* attributes are used to amplify a tag
* When a web browser interprets an HTML tag, it will also look for its attributes so that it can display the web page's elements properly

* attributes provide additional information about elements
* attributes are always specified in the start tag
* attributes usually come in name/value pairs like: name = "value".
* example - The href attribute
* <a href = "https://www.verzeo.in/"> Verzeo </a>
* width = "500" ; so width "500" means 500 pixels wide
* <img src = "img-girl.jpg" alt = "image not displayed">
  if the image is not displayed or crashed
* language attribute can be declared in <HTML> tag
  <html lang = "en-US">
* Double quotes around attribute values are the most common in HTML but single quote can also be used.
  In some situation when the attribute value itself contain double quotes, it is necessary to use single quotes; <P title = 'John "shotgun" Nelson'>

# CSS

* CSS stands for cascading style sheets
* CSS defines how HTML elements are displayed on a web page
* While styling inside an HTML file, it is much more common to create a separate .css file and then link it to the HTML file

## # background in CSS

* The background property in CSS allows you to control the background of any element (what paints underneath the content in that element). it is a shorthand property which means that it allows you to write what would be multiple CSS properties in one
* Sample code body {
  background: lightblue url("img_tree.gif") no-repeat fixed center;
  }

* CSS background properties
  - background-color - specifies the background color to be used
  - background image - specifies one or more background images to be used.
  - background repeat - specifies how to repeat the background images

# border in CSS

* The border property in CSS is used to style the border of an element. This property is a combination of these other properties border width, border style & border color as can be used as a shorthand notation for these 3 properties
* CSS border properties allow you to specify the style, width, & color of the element.
* The border style property specifies what kind of border to display.
  - dotted - defines a dotted border
  - dashed - " " dashed "
  - solid - " " solid "
  - double - " " double "
  - groove - " " 3D grooved ". This effect depends on the border color value
  - ridge - " " 3D ridge " " "
  - inset - " " 3D inset " " "
  - outset - " " 3D outset " " "
  - none - defines no border
  - hidden - defines a hidden border.

# CSS Selector

* a CSS selector is a part of a CSS rule set that actually selects the content you want to style
* CSS selectors are used to "find" (or select) the HTML elements you want to style.
* The element selector selects the HTML based on the element name.
* example
```
p{
    text-align: center;
    color: red;
}
```

# CSS id Selector

* The id selector uses the id attribute of an HTML element to select a specific element
* The id of an element is unique is within the page so that 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
* eg.

```
# Para1 {
    text-align: center;
    color: red;
    }
```

## Selector in CSS

element selector

```
h2 {
    color: #70039;
    }
```

Universal selector

```
* {
    color: #70039;
    }
```

id selector

```
#content {
    color: #E4293;
    font: size: 5px
    }
```

class selector

```
.main {
    margin: 100
    margin bottom: 10px
    }
```

# CSS selector specificity (specificity)

* If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific & therefore wins out think of speciality as a score / rank that determines which style declarations are ultimately applied to an element
* Specificity is a weight that is applied to given CSS declaration determined by the number of each selector type in the matching selector. When multiple declaration have equal speciality, the last declaration found in the CSS is applied to the element. Specificity only applies when same element is targeted by multiple declaration. As per CSS rules, directly targeted

elements will be always takes precedence over rules which an element inherits from its ancestor

* The Universal selector (*) has low specificity while (ID) selectors are high specific!

*

| Style attribute , ID , Class Psuedo-class attribute , Elements |
| --- |
| most specificity value ⟶ least specificity value |

# CSS fonts

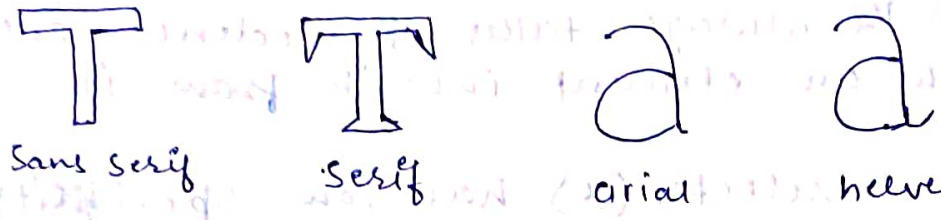
* css fonts is a module of CSS that defines font related properties and how fond resources are loaded. It lets you define the style of a font, such as its family, size & weight, line height, and glyph variants to use when multiple are available for a single character.
* a glyph is an element symbol within an agreed sets of symbols, instead to represent a readable characters for the purposes of writing

## CSS font families

* In HTML & XHTML, a css font family property is used to specify a list of prioritized fonts & generic family names; in conjunction with co-relating font properties, the list determines the particular font face used to render characters. a font family is a grouping of fonts defined by a shared design styles
* In CSS there are 2 types of font family names

1.) Generic family - a group of font families with a similar look (like "serif" or "monospace")

2.) font family - a specific font family (like "Times new roman" or "Arial")




# CSS font size

* The font size property sets the size of the text
* eg.
```
h1 {
    font-size: 40px;
   }
h2 {
    font-size: 30px;
   }
```

# CSS font-weight property

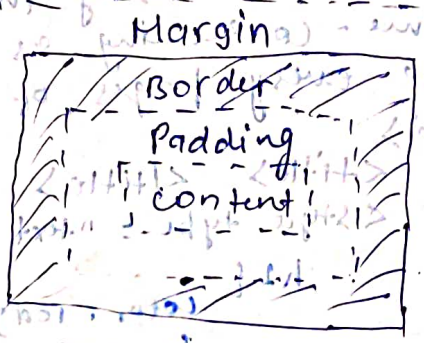
* The font-weight property sets how thick or thin characters in text should be displayed.
* CSS syntax.
* font-weight: normal | bold | bolder | lighter | number | initial | inherit;
  - normal - defines normal characters. This is default
  - Bold - defines thick characters
  - Bolder - Defines thicker characters
  - Lighter - defines lighter characters
  - 100 - defines from ~~thick~~ thin to thick characters
    400 is same as normal
    700 is same as bold.
  - initial - sets this property to its default value
  - inherits - inherits this property from its parent element.

# CSS text-align property

* The text-align property specifies the horizontal alignment of text in an element
* syntax
* text-align: left | right | center | justify | initial | inherit;

- left - Aligns text to the left
- Right - aligns text to the right
- Centes - centess the text
- Justify - stretches the line so that the each line has equal width
- initial - sets this propesty to its default value
- inhesit - inhesits this propesty from its parent element

## # CSS box Model

* in css, the term "box model" is used when we talk about design & layout
* The css box model is essentially a box that wraps around every HTML element
* It consists of - margin, bordes, Padding, & the actual content.
* 

  - Margin - clears an area outside the border The margin is transparent.
  - bordes - A bordes that goes all around the padding & content
  - Padding - clears an area around the content. The padding is transparent
  - content - The content of the box, where text & images appear

* The box model allows us to add a bordes around elements, & to define space between elements.

## # Inline vs Internal vs External css

* Inline css
  - It is used to style a specific HTML element. for this css style, you'll only need to add the style attribute to each HTML Tag, without using

the selectors

- The CSS type is not really recommended, as each HTML tag needs to be styled individually. Managing your website may become too hard if you only use inline CSS.
- It is useful in some cases where we don't have to access to CSS files or need to apply styles for a single element only.
- e.g.

```
<h1 style="color:red; font-size:50px;">
        inline css example </h1>
```

* Internal CSS

- Internal or embedded CSS requires you to add <style> tag in the <head> section of your HTML documents.
- This CSS style is an effective method of styling a single page. However using this style for multiple pages in time-consuming as you need to put CSS rules to every pages of your websites.
- eg.

```
<head>
  <title> </title>
  <style type = "text/css">
    h1 {
      color : red;
      font-size : 50px;
    }
  </style>
</head>
<body>
  <h1> Inline CSS example </h1>
</body>
```

* External CSS

- With external CSS, you'll link your web pages to an external .css file, which can be created by any text editor in your device (e.g notepad++).
- This type is a more efficient method, especially for styling a large websites by editing one .css file, you can change your entire site at once
- eg.

```
<head>
  <link href "myscc-code.css"
        rel = "stylesheet" />
</head>
<body>
  <h1> external CSS example </h1>
</body>
```

# Bootstrap

* Bootstrap is a potent front-end framework used to create modern websites & web pages. It's an open source & free to use, yet features numerous HTML & CSS templates for UI interface elements such ~~that~~ as buttons & forms. It also supports Java Script extensions.
* In other words, It is a free collection of tools for creating a websites & web applications. It contains HTML & CSS-based design templates for typography, forms, buttons, navigation & other interface components as well as optional JS extensions.
* Bootstrap is a very common framework used for front-end development.

## Framework

* a framework, or software framework, is a platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform. e.g a framework may include predefined classes & functions that can be used to process input, manage hardware devices & interact with system software.
* A framework facilitates the following:
  - Inversion control
  - default behaviours
  - extensibility
  - non-modifiable framework code
* Usually bootstrap is not about memorizing the code but understanding how to refer the documentation & use it for your own projects from the official websites.

# Bootstrap buttons

* Bootstrap provide us with different classes that they can be used with different tags, such as <buttons>, <a>, <input>, & <label> to apply custom button styles. Bootstrap also provides classes that can be used for changing the state & size buttons. also, it provides classes for applying toggle, checkbox & radio button like effects.
* Different types of buttons are :-

1. btn.
2. btn-default.
3. btn-primary.
4. btn-success.
5. btn-info.
6. btn-warning.
7. btn-danger.
8. btn-link.

* ex- <button type="button" class="btn btn-primary"> primary </button>
<button type="button" class="btn btn-danger"> Danger </button>

# Jumbotron

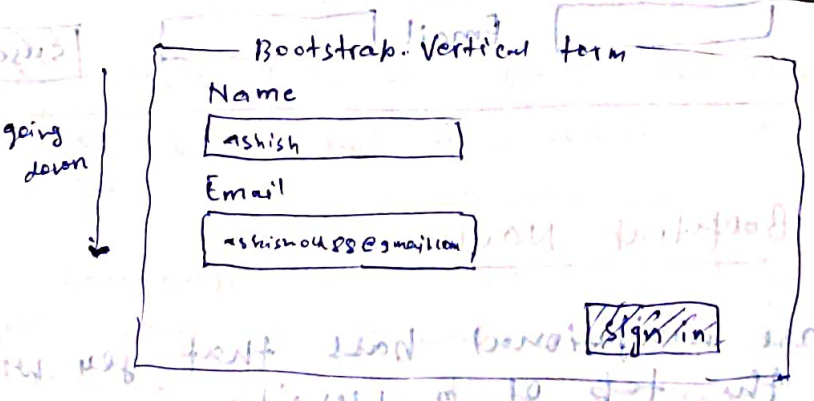
* a jumbotron indicates a big box for calling extra attention to some special content or information. A jumbotron is displayed as a grey box with rounded corners. It also enlarges the font of the font sizes of the text inside it.
* Inside a jumbotron we can put nearly any Valid HTML, including other bootstrap elements / classes.
* In other words, Jumbotron is a lightweight, flexible component that can be optionally extend the entire viewport to showcase key marketing messages on your site.
* To insert a Jumbotron into your HTML file you need to copy & paste as shown below.

```
<div class = " Jumbotron ">
<h1 class = " display-4 > VERZEO ! </h1>
<p class = " lead "> welcome to Verzeo web page </p>
<hr class = " my-4 ">
<p> you can register .... </p>
<a class = " btn btn-primary btn-lg " href = " http://...
                                    ..... | " role = " button "> register </a>
</div>
```

# Bootstrap forms

* Bootstrap forms are input-based components which are designed to collect users' data. Used as a login, subscribe or contact form, all of them can be easily customized
* Bootstrap comes with many default classes for forms.
* Bootstrap provides 3 types form layout :-
  - Vertical form (default)
  - Horizontal form
  - Inline form.

- Vertical form

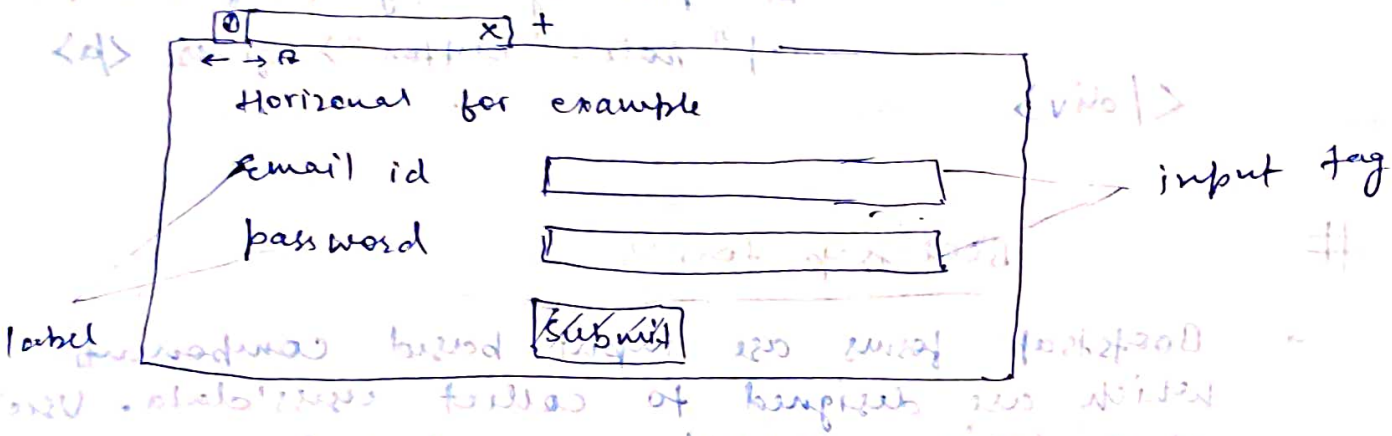



* ex.

```
<form action = " /action-page.php ">
<div class = " form-group ">
<label for = " email "> Email address : </label>
<input type = " email " class = " form-control " id = " email ">
</div>
<div class = " checkbox ">
<label> <input type = " checkbox "> Remember me </label>
</div>
```

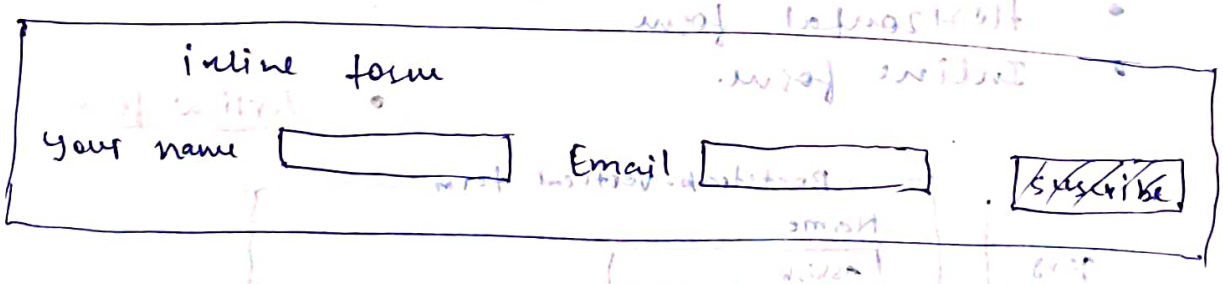
- Horizontal form

a horizontal form means that label are aligned next to the input field (horizontal) on large & medium screens. on small screens (767 px & below), it will transform to a vertical form (labels are placed on top of each input).




- Inline form

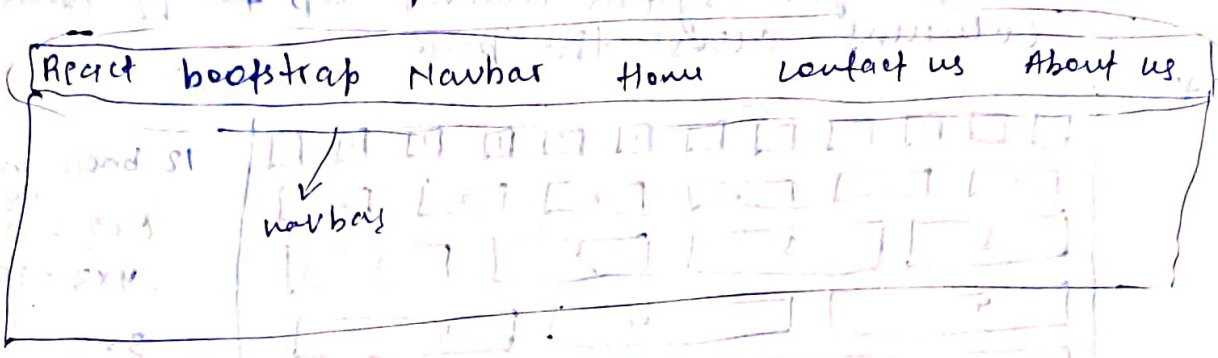
In an inline form, all of the elements are inline, left-aligned, & the labels are alongside. This only applies to forms with viewport that are at least 768px wide.




# Bootstrap Navbar

* Navbars are navigational bars that you will often see on the top of a websites
* A navigational bar a navigation header that is placed at the top of the page.
* With Bootstrap, a navigation bar can extend or collapse, depending on the screen size.
* A standard navigation bar is created with <nav class = "navbar navbar-default">
* followed by a responsive collapsing class : .nav

-bar-expand-xl|lg|md|sm (stack the navbar vertically on extra large, large, medium or small screens).




* ex:

```
<nav class="navbar navbar-inverse">
<div class="container-fluid">
<div class="navbar-header">
<a class="navbar-brand" href="#">website name</a>
</div>
<ul class="nav navbar-nav">
<li class="active"><a href="#">home</a></li>
.
.
.
</ul>
</div>
</nav>
```

# Bootstrap Grids

* Every screen size is different for different devices. so your web application must accomodate for those changes. for this we use grids

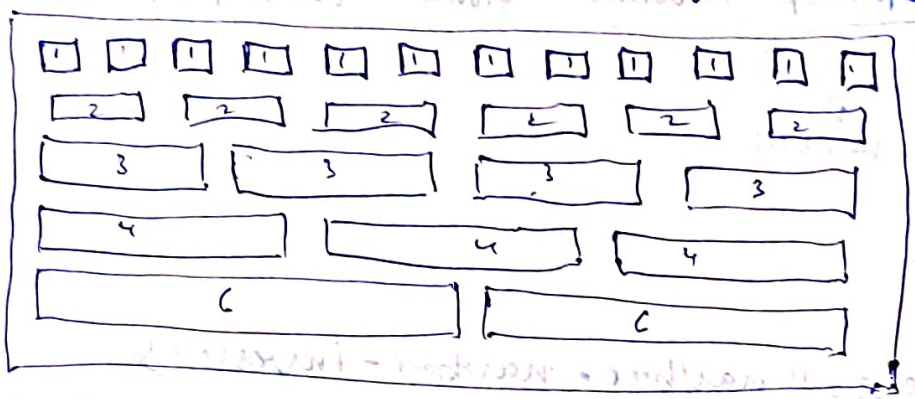
* The bootstrap grid system is used for layout, specifically responsive layouts. Understanding how it works is vital to understanding bootstrap. The grid is made up of groupings of rows & columns inside 1 or more containers. The bootstrap grid can be used alone, without the bootstrap javascript & other css components

* The grid system for bootstrap is one of its most fundamental features.

* The grid system provides the core mechanism by which using bootstrap allows websites to look

good across multiple devices of multiple screen sizes.

* Bootstrap's grid system allows up to 12 columns across the page.



12 basic available columns

6 x 2 = 12

4 x 3 = 12

3 x 4 = 12

2 x 6 = 12

* Bootstap grid system is responsive, & the columns will re-arrange depending on the screen size: on a big screen it might look with the content organised in 3 columns but on a small screen it would be better if the content items were stacked on top of each other.
* The grid system call will make use of the class = "row"
* Steps:-
  - So, to create the layout you want, create the container ( ).
  - Next create a row ( ).
  - Then add the desired number of columns (tags with appropriate .col-*-* classes).
  - Note that numbers in .col*-* should always add up to 12, for each row.
* Inside of a row class, the format :-
  - col-screensize-Numbers of columns

    xs, sm, md, xl

    1, 2, 3, 4, 6, 12
* Inside a row class, we have the following format:
  - col-md-6
* So we can define how the columns should be shown when the screen gets resized.

* The following table summarizes how the bootstrap grid system works across multiple devices.

- extra small
- < 768 px
- small
- >= 768 px
- medium
- >= 992 px
- Large
- >= 1200 px

* Basic structure.

```
<div class="container">
<div class="row">
<div class="col-*-*"></div>
<div class="col-*-*"></div>
</div>
<div class="row">
<div class="col-*-*"></div>
</div>
<div class="row">
...
</div>
</div>
```

screen size

no of columns

# HTML

* Hypertext markup language

/

means angle bracket

* Every browser has a rendering engine installed in it.

It is the only engine which renders the whole HTML page & get the proper interactive good looking UI out of the HTML

That is why the content we write within the tags are visible thing over the webpage

Some HTML attributes

1.) Href attribute
2.) Src attribute
3.) Width & heigh attribute
4.) Alt attribute
5.) lang attribute
6.) Title attribute

HTML formatting

- <b> - bold text
-  - Important text
- <i> - Italic text
-  - emphasized text
- <mark> - marked text

- <small> - small text
- <del> - deleted text
- <ins> - inserted text
- <sub> - suscribe text - $a_b$
- <sup> - superscript text w- $a^b$

## HTML Links

- defined with <a> tag (anchor tag)
- the target attribute
  1.) -blank : opens in new window
  2.) -self : opens in same window
  3.) -parent : opens in parent frame
  4.) -top : opens in full body of the window

<a href=" " target "_blank"> Visit! </a>

## HTML Tables

- defined with <table> tag.
- Table is defined with the <tr> tag
- Table header is defined with the <th> tag
- Table data/cell is defined with the <td> tag

  - Table optimisation
    - bordered table
    - collapsed table
    - cell padding
    - Border spacing
    - cell spanning : several columns
    - adding a caption
- rowspan - is property

| Name | Ashish |
|---|---|
| Ph. no | 1224 |
| | 5678 |

> because of rowspan=2

# HTML forms

- <input type = "text"> Defines one-line text input field
- <input type = "radio"> - for selecting one of many choices
- <input type = "submit"> - for submitting the form

~~<input type = "text" name = "first name" value = "John">~~

~~first name~~

ex - gender = male ☐ female ☐

## HTML colors

- background-color : blue;
- color : red;
- color for borders
  - border : 2px solid green;

* use - table, th, td {
  border : 2px solid green;
  }

# # CSS frames

~~@keyframes anim1 {~~

div {
  width : - ;
  height : - ;
  background-color - red;
  animation-name - anim1;
  animation-duration - 5s;
  }

@keyframes anim1 {
from { background-color : blue ; }
to { background-color : green ; }
}

OR

@ keyframes anim1 {

0% { background-color: ; }
25% { ; }
50 % { ; }
100% { ; }
}

*Animation means repeatedly change of background colors;

* try - 0% { background-color; red; left: 0px; top: 10px; }

## CSS animation

- @ keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation : name 7s infinite;

four-properties

1.) normal
2.) reverse
3.) alternate
4.) alternate-reverse

# CSS - Multiple-columns

- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width

# HTML

- We can use extension .Htm or .Html.
- <a href = " "> </a>

  a is the anchor tag
-  bold 
   Italic 
  <u> underline </u>
  <big> </big>
  <small> </small>
  <hr> horizontal lines </hr>
-  CO < sub > 2 </sub> 
   a < sub > 2 </sub> + bx+c 
- <pre> this is written in order to preserve the text as it is rendered as it is
  </pre>
- <header>
  <main>
  <footer>
- <main> → The main opening tag
  <section> → A page section
  <article> → A self contained content
  <aside> → content aside from the content (eg. Ads etc)
  </main> → The main closing tag
-  is an example of inline element.
- Block elements

  <address> <article> <aside> <blockquote> <canvas>
  <dd>  <dl> <dt> <fieldset> <figcaption>
  <figure> <footer> <form> <H1> ... <H6> <header>

<hr> <li> <main> <nav> <noscript> <ol> <p> <pre> <section> <table> <tfoot> <ul> <video> </video>.

Inline elements

<a> <abbr> <acronym> <b> <bdo> <big> <br> <button> <cite> <code> <dfn>  <i> <img> <input> <kbd> <label> <map> <object> <output> <q> <samp> <script> <select> <small>   <sub> <sup> <textarea> <time> <tt> <var>

- thead tag: used to wrap table head (caption ~~head~~ & tr with th)

  tbody tag: used to wrap the table body

- colspan attribute
  this attribute is used to create cells spanning multiples columns

  <th colspan = "3"> ashish </th>
  ↳ spans 3 columns

- <video src = "vid.mp4" controls autoplay, 100% width = "523 px">
  </video>

* SEO (search engine optimization)

- types
  - on page SEO
  - off page SEO

- HTML SEO
  HTML developers can implement SEO using the following techniques.
  1. set the title very nice & to the point
  2. set the meta description

<meta name = "description" content = "..." >

3. Set a nice URL slug
4. Set the meta keywords tag
5. Set the meta author tag

<meta name = "author" content = "ashish" >

6. Set a favicon
7. compress image & other resources
8. Remove unused HTML/CSS & JS files + compress them
9. add alt text to images

minify.com

# CSS

* DOM

Dom stands for document object model. When a page is loaded, the browser creates a DOM of the page which is constructed as a tree of objects

* Selectors

- h1, h2, h3, div {
  color: blue;
  }
- P.red {
  color: red; → all paragraphs of class=red will get color of red
  }
- * can be used as a universal selector to select all the elements

* {
  margin: 0;
  padding: 0;
}

* HSL → hue, saturation, lightness
  hsl(8, 90%, 63%)

  RGBA → A for alpha

* background-repeat

- repeat-x → repeat in horizontal direction
- repeat-y → " " vertical "
- space

* background-size

- cover - fits & no empty space remains
- contain - fits & image is fully visible
- auto - displays in original size
- {{width}} - set width & height will be set automatically
- {{width}} {{height}} → set width & height

* background-attachment: fixed; → fix the image

* text-transform: capitalize
* line-height: 1.8x

* z-index property

the z-index property specifies the stack order of an element
It defines which larger will be above which in case of overlapping elements

* Flexbox

- float property is simple. It just flows the element towards left & Right.
- clear property is used to clear the float. It specifies what element can float beside a given element
- css flexbox aims at providing a better way to layout, align and distribute space among items in a container.

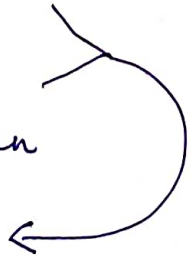
```
container {
        display : flex;
      }
```

  - flex-wrap :
  - justify-content : defines alignment along main axis
  - align-items : defines alignment along cross axis
  - align-content : align a flex containes lines when there is extra space in the cross axis

  - flex items
    - → order :
    - → align-self : allows default alignment
    - → flex-grow :
    - → flex-shrink :

* grid

- grid-row-gap :
- grid-column-gap :
- grid-template-column
- grid-gap : apx bpx



- css media queries

```
@media only screen and (max-width: 800px) {
    body {
        background: red;
    }
}
```

* Transform

- transform-origin;
- css 2D transform methods
  - translate()
  - rotate()
  - scaleX()
  - scaleY()
  - skew()
  - matrix()
  - scale()
- css 3D transform
  - rotateX()
  - rotateY()
  - rotateZ()