Web Development Using ReactJS

- *Introduction*
    - If you wanna be a Front-end developer, This course is absolutely suitable for you.

      I'm Hanuman Kumar, working as a multi-skill trainer at APSSDC. Currently, I'm dealing with the sessions related to the frontend. We will discuss front-end development exclusively in this series of video lectures.

      But before going in, We've to know what a web application is? And why we're using it nowadays

      A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet.

      Millions of businesses use the Internet as a cost-effective communication channel. It lets them exchange information with their target market and makes fast, secure transactions.

      However, effective engagement is only possible when the business is able to capture and store all the necessary data, and have a means of processing this information and presenting the results to the user.

      Web applications use a combination of server-side scripts (PHP and ASP) to handle the storage and retrieval of the information, and client-side scripts (JavaScript and HTML) to present information to users.

In this tutorial, we will discuss the concepts like

- HTML5
- CSS
- Bootstrap
- JavaScript and
- React Js

I hope you understand what are the concepts we are gonna cover in the upcoming sessions.
Why are you waiting? Let's get started with an introduction to HTML

Chapter - 1

HTML

Now, let us start with the concept of HTML:

- **HTML:** stands for HyperText Markup Language.we have to use HTML tags for formatting the data (<element> </element>)
- HTML is used to create web pages(either static or dynamic).

  A **static website** contains **Web** pages with fixed content. Each **page** is coded in HTML and displays the same information to every visitor.

  A dynamic website contains dynamic pages or elements and can be changed according to the visitor's actions.

- By default, a web browser will display exactly what you type
- HTML is the heart of web pages and HTML5 is the latest version to be approved by the World Wide Web Consortium (W3C)

Now it's time to discuss the structure of HTML.

The HTML structure consists of 3 parts.

- HTML version information
- Head &
- Body

We can declare the HTML version by using **doctype.**

Head:

- The head part content is not visible on the screen. It works on the background of the webpage/website

- In the head of the web page contains a title, Meta tags, favicon, Styles & Some Scripts related to the document.

Body:

A. The body session can be devices into 3 major parts/sessions
    1. Block-level elements
    2. Inline
    3. Semantic

Now start with the practical exposure. I am going to create a folder on my desktop environment.
I chose a desktop as a workspace here because it is very easy to access.

I am using the name as a project here. You can use your customized name instead.

It's better to use a text editor for organizing the code.

Here I selected Visual studio code as text editor. It is available online and this is open source.

I am opening my project folder by clicking on the file menu and selecting the created project by using the open folder option. We can use ctrl +k or ctrl + o as shortcuts for opening the project.

Here by clicking on a new file icon, we can create a new file. I gave my name as index.html here. We have to use the html extension. If we are using a server, we have to select the root file so that the execution will be starting from this file. Index name defines itself as a root file.

Initially we have to use doctype for declaring the version information.

We have to start with html tags for building.

The html tags are started with markup tags and we have to close them with slash like this.

As we discussed earlier, we have to use two more sections within the html. Those are head and body.

Syntax:

`<starting tag> content </ending tag>`

Example:

`<html>  </html>`

1.1.12 Structure of Html:

```
<html>
   <head>
      Head part here
   </head>
   <body>
      body part here
   </body>
</html>
```

Now we came to understand how to execute the HTML code and the basic building blocks of HTML.

We will discuss the head section in the next video.

Head part:

Actions we are able to perform in head part:

- Able to give a title to our webpage
- Able to keep an icon as favicon
- Allows us to use external CSS, internal css and manifest files.
- Moreover, The **HTML** `<script>` **element** is used to embed executable code or data, this is typically used to embed or refer to JavaScript code.
- Provides the way to use meta elements.

The HTML <head> element is a container for the following elements: <title>, <style>, <link>, <meta>,<script>, and <base>

By using title (<title> </title>) tags we can denote the title of the page that displays on the page tab.

Style element allows us to use internal styles

By using link element we can add favicon, manifest and external styles

The information represented in meta is not visible anywhere but will reflect the entire document.

Script tag allows us to us Javascript functionalities

We already took a look into the concept of the title. Now let's start with the favicon

To keep an icon as a favicon to our webpage first of all let me introduce you to different types of images.

1. Raster Images
2. SVG Images

Raster Images: Raster Images are images which can be taken by camera or uploaded with mobile.
If we Zoom Raster Images at a certain point Raster Images miss their clarity

SVG Images: Svg Images are the images which can be downloaded through google.

If we zoom Svg Images won't miss their clarity.

Download an image in SVG format and store that image in our project at the image folder.

<u>Syntax to keep an image as favicon to our webpage :</u>

<link rel="icon" type="favicon" href="image/myimage.txt" />

Link attribute is used to navigate or link

Here **rel** stands for the relation
The type used to define the type
Href stands for head reference path where we have to mention the path.

Meta:

- The information specified in meta tags is not visible anywhere on the web page. But it reflects the entire HTML document.
- The type of metadata provided by the `meta` element can be one of the following:
    - If the `name` attribute is set, the `meta` element provides *document-level metadata*, applying to the whole page.
    - If the `http-equiv` attribute is set, the `meta` element is a *pragma directive*, providing information equivalent to what can be given by a similarly-named HTTP header.
    - If the `charset` attribute is set, the `meta` element is a *charset declaration*, giving the character encoding in which the document is encoded.
    - If the `itemprop` attribute is set, the `meta` element provides *user-defined metadata*.

Let's look into the example

```
<meta charset="utf-8">
```

This element simply specifies the document's character encoding — the character set that the document is permitted to use. `utf-8` is a universal character set that includes pretty much any character from any human language. This means that your web page will be able to handle displaying any language; it's, therefore, a good idea to set this on every web page you create! For example, your page could handle English and Japanese just fine:

If you set your character encoding to `ISO-8859-1`, for example (the character set for the Latin alphabet), your page rendering may appear all messed up:

Many `<meta>` elements include the `name` and `content` attributes:

- `name` specifies the type of meta element it is; what type of information it contains.

- `content` specifies the actual meta content.

Two such meta elements that are useful to include on your page define the author of the page, and provide a concise description of the page. Let's look at an example:

Bodypart :
The main content will be displayed in this body tag.

Let us learn about different types of elements in Html5:

1. Block-level Elements :
    - A Block-level element occupies the entire horizontal space of its parent element (container), and vertical space equal to the height of its contents
    - A block-level element is an html element that being start with a new line in the web application

    1. All heading tags (h1 to h6)
    2. paragraph tag (p)
    3. Form
    4. Division (div)
    5. Lists (ol, ul, dl,li and dt)
    6. Table
    7. Horizontal line (hr)
    8. All semantic elements

➔ Next video
    - ◆ We covered all the concepts of block level elements except semantic elements, Let's start with this concept now.
    - ◆ Semantic elements are replacements of division tags.

- A **semantic element** clearly describes its meaning to both the browser and the developer.
- This is for increasing the accessibility of a website.

- Semantic elements are
  - Header
  - Footer
  - Section
  - Article
  - Aside
  - Main
  - Nav
  - summery

1.All heading tags:
To define heading we have to use h1 to h6 tags.
The font size decreases gradually from h1 to h6.
The syntax goes here:
<h1> content </h1>
<h2> content </h2>
<h3> content </h3>
<h4> content </h4>
<h5> content </h5>
<h6> content </h6>

All the heading tags occupy the complete width of the screen.

Paragraph tag :
To display the content in the paragraph we will use paragraph tag

Syntax:

 content

This paragraph tag occupies the complete width of the screen.

Before going to semantic tags/elements in the html5 version all tags are divided into 2 types i.e.,

1. Semantic(Header, main, footer, section, article etc.,)
2. Non-Semantic(div, span)

Semantic Elements:

Semantic Elements are similar to div tags used to divide the content into sections.
But div tags don't contain any Description .
Semantic Tags clearly explains its meaning to both the user and browser.

Semantic Elements are :

1. Section
2. Article
3. Aside
4. Nav
5. Header
6. Footer etc.,

Section :

The <section> element defines a section in a document.
Syntax goes here :

```
<section>
    content
</section>
```

Article :

The <article> element is used to define a independent section in a webpage
Example : Blog in a Newspaper

Syntax :

```
<article>
      Content
</article>
```

Header :

The <header> element represents a container for introductory content or a set of navigational links.

A <header> element typically contains:

- one or more heading elements (<h1> - <h6>)
- logo or icon
- authorship information

Syntax :

```
<article>

   <header>

     <h1> content </h1>

     <p> content   </p>

  </header>

</article>
```

Footer :

The <footer> element defines a footer for a document or section.

A <footer> element typically contains:

- authorship information
- copyright information
- contact information
- sitemap
- back to top links
- related documents

You can have several <footer> elements in one document.

Syntax:

```
<footer>
    <p> content </p>
    <p> content 1 </p>
</footer>
```

Nav element:

The `<nav>` element defines a set of navigation links like home, contactUs etc.,

Syntax :

```
<nav>
  <a href = “path” >
</nav>
```

Aside :

The `<aside>` element defines some content aside from the content it is placed in (like a sidebar).

The `<aside>` content should be indirectly related to the surrounding content.

Syntax :

```
<aside>
  <h1> content </h1>
  <p> content </p>
</aside>
```

Inline Elements :

The Elements which don't occupy the complete width of the screen (we are able to see the output side by side).

1. Span tag
2. Image tag
3. Anchor Tag
4. Button
5. Input etc..,

Span tag :

The tag is an inline container used to mark up a part of a text, or a part of a document.

The tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.

The tag is much like the element, but is a block-level element and is an inline element.

Syntax :

 content

Image Tag :

Image Tag is used to display an image in the body part.

Syntax :

<img src=”path of the image”>

Anchor Tag:

The <a> tag defines a hyperlink, which is used to link from one page to another page

Syntax:

<a href="< resource page link> Name for displaying purpose "

A linked page is normally displayed in the current browser window, unless you specify another target

Button:

It defines a clickable button. It tell the browser what type of button is

Syntax:

<button > Click Here..! </button>

Input: <input>

It's the most important element in the form data. By using this we can display it in several ways depending on the type attribute.

Navigation Elements :

All Navigation Elements are inline Elements where we are able to see the output side by side.

Navigation Elements are used to navigate

All Navigation Elements uses <a,href> (anchor tag)

1. Inbound navigation
2. Outbound navigation
3. Tel
4. Mailto

Inbound navigation :

Inbound navigation is used to navigate to the content present in the same file .

(we need an identifier selector to select that particular tag).

Syntax :

```
<a href="#one">h3 content </a>

<h1> Hai all </h1>

<h2>welcome </h2>

<h3 id="one"> to APSSDC </h3>
```

Outbound navigation :

Outbound navigation is used to navigate to the content present in another using <a,href>

(create another file "resume.html" and write some content using h1 tag)

Syntax :

```
<a href="resume.html>Content in resume.html </a>
```

Mailto :

Mailto navigation element is used to send a mail to a particular person (after doing some configuration settings we are able to send the mail within the web page).

Syntax :

```
<a href="mailto: xyz@gmail.com"> mail  </a>
```

Tel :

Tel navigation element is used to call a particular person.

Syntax :

 phone

Chapter - 2

CSS

Introduction

- CSS stands for Cascading Style Sheets
- Which is used to apply the beautification to the HTML document

What are we gonna discuss in the CSS concept?

- CSS Types
    - Inline
    - Internal
    - External
- CSS Selectors
    - Basic or simple selectors
        - Universal Selector
        - Type Selector
        - Class Selector
        - ID Selector
        - Attribute Selectors

    - Combinators or Relational selectors

        - Descendant combinator
        - Child combinator
        - General sibling combinator
        - Adjacent sibling combinator
        - Columns combinator
    - Pseudo Selector
        - Pseudo class selector
        - Pseudo element selector
- CSS Flexbox
- CSS Responsive Web Design

Syntax of CSS:

Selector {property: value;}

Kinds of CSS

CSS can be include to HTML documents in 3 ways:

- Inline - by using the style attribute inside HTML elements
- Internal - by using a <style> element in the <head> section
- External - by using a <link> element to link to an external CSS file

Inline CSS:

<h1 style="color:blue;"> A Blue Heading </h1>

Internal CSS:

We have to include css styles in style tag in head tag

```
<!DOCTYPE html>
<html>
<head>
    <title> :) Css Document (:</title>
    <style type="text/css">
        h1 {
            color: blue;
        }
    </style>
</head>
<body>
 <h1>Internal CSS</h1>
</body>
</html>
```

External CSS:

For external css we have to take the css file as separate and include that in the respective html file. We've to do this by using link tag

CSS Selectors

- **Universal selector**
  Selects all elements. Optionally, it may be restricted to a specific namespace or to all namespaces.

  Syntax: * ns|* *|*

  Example: * will match all the elements of the document.

- **Type selector**

  Selects all elements that have the given node name.
  Syntax: elementname { property : value }

  Example: input will match any <input> element.

  body{ background: #ddd}

  Here body is the type selector, background is the property and #ddd (grey color) is the value

- **Class selector:**

Selects all elements that have the given class attribute. We've denote this with dot(.) symbol

Syntax: .className

Example: .index will match any element that has a class of "index".

- **ID selector (*Identifier*):**
  Selects an element based on the value of its id attribute. There should be only one element with a given ID in a document.

  Syntax: #idName

  Example: #demo will match the element that has the ID "demo".

- **Attribute selector:**

  Selects all elements that have the given attribute.

  Syntax: [attr] [attr=value] [attr~=value] [attr|=value] [attr^=value] [attr$=value] [attr*=value]

  Example: [autoplay] will match all elements that have the autoplay attribute set (to any value).

  a[target] { background: #ddd; }

  a[target="_blank"] { background: yellow; }

- **Grouping Selector**

  The , is a grouping method, it selects all the matching nodes.
  Syntax: A, B
  Example: div, span will match both and elements.
  h1,
  h2,

.heading { background: #000; color: #fff; }

- **Descendant combinator**
  The (space) combinator selects nodes that are descendants of the first element. The descendant selector matches all elements that are descendants of a specified element.

  Syntax: A B

  Example: div p { background-color: yellow; }

- **Child combinator**
  The > combinator selects nodes that are direct children of the first element.

  Syntax: A > B

  Example: ul > li will match all <li> elements that are nested directly inside a <ul> element.

- **General sibling combinator**
  The ~ combinator selects siblings. This means that the second element follows the first (though not necessarily immediately), and both share the same parent.

  Syntax: A ~ B

  Example: p ~ span will match all elements that follow a <p>, immediately or not.

- **Adjacent sibling combinator**
  The + combinator selects adjacent siblings. This means that the second element directly follows the first, and both share the same parent.

  Syntax: A + B

  Example: h2 + p will match all <p> elements that directly follow an <h2>.

- **Column combinator**
  The || combinator selects nodes which belong to a column.

  Syntax: A || B

  Example: col || td will match all <td> elements that belong to the scope of the <col>.

- **Pseudo classes**
  The : pseudo allows the selection of elements based on state information that is not contained in the document tree.

  Example: a:visited will match all <a> elements that have been visited by the user.

- **Pseudo elements**
  The :: pseudo represents entities that are not included in HTML.

  Example: p::first-line will match the first line of all **<p>** elements.

CSS FlexBox:

CSS Flexible Box Layout is a module of **CSS** that defines a CSS box model optimized for user interface design, and the layout of items in one dimension.While we working flexbox concept first you have to create parent element with some child elements.Then we can apply flexbox layout with display:flex property. We can see child elements can align side by side.

Syntax: .parentClass { display:flex; }

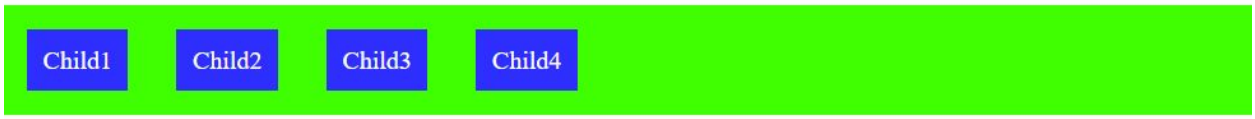
Example:

For this concept we will take the below html file as common for this file and we can apply flex properties one by one.

Flex.html	*Style.css*

<!DOCTYPE html>
<html>
<head>
<title> :) Css Document (:
</title>
 	
<link rel="stylesheet" type="text/css" href="css/style.css">
 	
</head>
 	
<body>
 	
Child1
 	
Child2
 	
Child3
 	
Child4
 	
</body>
</html>
	.parent {
	display: flex;
	background-color: #40FF00;
	}
	
	.child {
	width: auto;
	padding: 10px;
	margin:1%;
	color: #fff;
	background-color: #2E2EFE;
	/*#2E2EFE is hexadecimal code for colors*/
	}

Output:



Flex-direction:

The flex-direction property defines in which direction the container wants to stack the flex items.It has property value row,row-reverse,column,column-reverse.

Style.css:

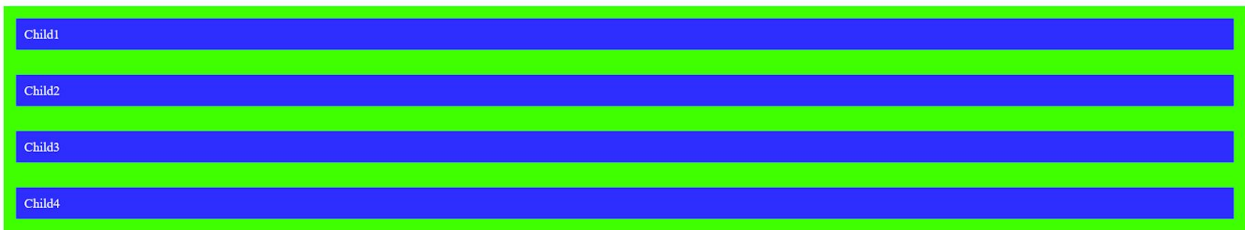
```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-direction: row-reverse;
}
```

Flex-direction:column

We can get childelements in column format

Example:

```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-direction: column;
}
```



Flex-direction:column-reverse

We can get data in column reverse format

Example:

```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-direction: column-reverse;
}
```

Flex-wrap:wrap:

The flex-wrap property specifies whether the flex items should wrap or not.Default is property value is nowrap.If child elements exceed it's total width we can't see those data for that we will use flex-wrap:wrap

Example:

Style.css:

```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-wrap: wrap;
}
```

```
.child {
        width: auto;
        padding: 100px;
        font-size:30px;
        margin:1%;
        color: #fff;
        background-color: #2E2EFE;
}
```

Note: The child4 element we can see in the next line.

Output:

Justify-content:

This property used to align flex items with property values of center,space-around,space-between,flex-start and flex-end .Try to apply those properties to the parent element.

Align-items :

This property used to align flex items with different values as shown in below

- Align-items:center
- Align-items:flex-start
- Align-items:flex-end
- align-items:stretch

Align-content:

This property used to align the flex lines. Let's some property values for the align-content

- Align-content:center
- Align-content:flex-start
- Align-content:flex-end
- Align-content:stretch
- Align-content:space-around
- Align-content:space-between

CSS Responsive Web Design

- Responsive Web Design make our web page look good on different devices
- Html and Css we will make webpage as a responsive

Viewport :

Viewport is the user's visible area of a webpage and it varies with respective device.To control viewport for different devices we will use viewport through <meta> tag

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

The above viewport we have to include in our webpage to get responsive for that webpage.

The meta tag gives the browser instructions on how to control the page's dimensions and scaling.The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

CSS Grid view:

To place elements easily on a web page grid-view will be helpful.A responsive grid-view often has 12 columns with a total width of 100% .The property box-sizing: border-box makes sure that the padding and border are included in the total width and height of the elements and we have to include in our css file.

```
* {  box-sizing: border-box; }
```

Example:

Style.css:

```
* {
        box-sizing: border-box;
   }

.child1{
        width:20%;
        float: left;
        background-color: red;
        padding: 10px;
}

.child2{
        width:70%;
        float: left;
        background-color: green;
```

```
        padding: 10px;
}
```

Output:

Media queries:

- Media queries are introduced in css3
- It uses the **@media** rule to include a block of CSS properties only if a certain condition is true
- We will use different breakpoints for different devices
- We can look same content of webpage with different response on different devices by using media queries

Example:

Here breakpoint is 768px .The below css styles applicable up to device width is 768px if device width is exceed 768px styles in media query is not worked.

```
@media only screen and (max-width: 768px) {
        body {
                background-color: lightblue;
        }
}
```

Breakpoints for different devices:

extra-small devices	max-width:600px
small devices	min-width:600px
medium devices	max-width:768px
large devices	min-width:992px;
extra-large devices	min-width:1200px

Chapter - 3

BootStrap

Contents of Bootstrap:

Introduction:

What is bootstrap:

- Bootstrap is open source css Framework
- And it is one of the most popular frameworks for developing responsive web applications
- It is very easy to use and implement the web web application in a simple manner

- By using this we can build Responsiveness web applications with the help of predefined classes
- Bootstrap 4 is compatible with all modern browsers
    - Firefox, Chrome, Safari, Opera and Internet Explorer 8 and 9
    - There are two ways to use bootstrap in our websites
        a. Include Bootstrap 4 from a CDN
        b. Download Bootstrap 4 from bootstrap.com
            - https://getbootstrap.com/

How to use bootstrap in the application:

- We can use bootstrap to websites in 2 ways

1.Online(using CDN)
2.Offline(download directory)

1.Online CDN for Bootstrap:
CSS:

- Have to follow some steps to install this CDN in websites

step 1: Open your html file **ex: "index.html" page**
step 2: In that html page head section past the CDN link
step 3: After that check the CDN link is working or not

- **Include Bootstrap 4 from a CDN:**

  When we are using CDN links, the internet is mandatory.

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>



<!-- Popper JS -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js">
```

```
</script>

<!-- Latest compiled JavaScript -->
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">
</script>
```

Note: Bootstrap 4 uses jQuery and Popper.js for JavaScript components (like modals, tooltips, popovers etc). Based on our requirement we can include these js libraries.

2. Download Bootstrap for offline:

- First Download CDN's Directory and use by following steps

A. Go to Website getbootstrap.com
B. Click ON Download button
C. Goto Your download directory and extract it
    a. Copy the extracting files and paste it into your project direct
    b. Obser the file structure. it has many files in that folder
    c. from those you have to open the dist folder
D. Bootstrap File Structure:

E. In the dist folder you will found two folders

1.CSS

CSS files	Layout	Content	Components	Utilities
bootstrap.css bootstrap.min.css	Included	Included	Included	Included
bootstrap-grid.css bootstrap-grid.min.css	Only grid system	Not included	Not included	Only flex utilities
bootstrap-reboot.css bootstrap-reboot.min.css	Not included	Only Reboot	Not included	Not included

2.JS

JS files	Popper	jQuery
bootstrap.bundle.js bootstrap.bundle.min.js	Included	Not included
bootstrap.js bootstrap.min.js	Not included	Not included

step 7:

In CSS Folder contains many files that you have to select and copy bootstrap.min.css and paste that file in the project directory.

Step 8:

Same like as CSS files there also js files in JS folder from there we will copy the bootstrap.min.js file and place this in project directory

Step 9:

Open the “Index.html” file.

Step 10:

<link rel=”stylesheet” href=”Here your file directory path”> copy this line and paste in the top of </head > tag. This is for the “bootstrap.min.css” file.

Step 11:

We have pasted the CSS file now Past Javascript in the top of </body> tag. for example <script src=”Here your file path”>. This is for the “bootstrap.min.js” file.

Example:

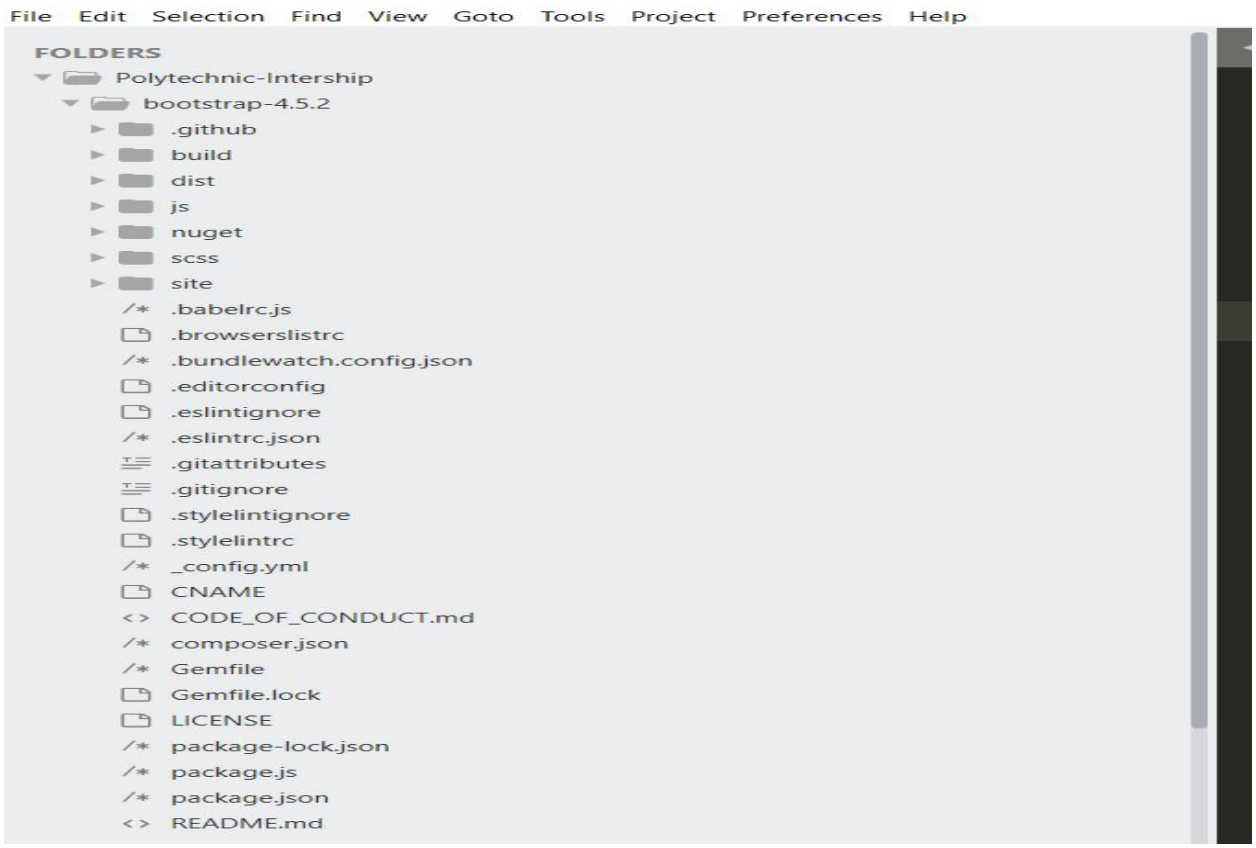
```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>bootstrap</title>
    <link rel="stylesheet" href="/css/bootstrap.min.css">
  </head>
  <body>
```

```
    <h1>Welcome to bootstrap</h1>
    <script type="text/javascript" src="/js/bootstrap.min.js">
  </body>
</html>
```

Note:

- If you are using the Bootstrap directory online then this will actually access data from the bootstrap own directory. this will take time from retrieving data from there.
- And if you are using the Bootstrap 4.0 directory from the download. then this will access data from your own directory. So this will easy to access data for your website

.CSS File



Then you have to include css and js file path in your required html pages.If you want js functionality you have to download jquery.min.js separately and include in html file.

Color codes:

Bootstrap 4 has some contextual classes to convey meaning through colors. Lets we will see classes with colors.

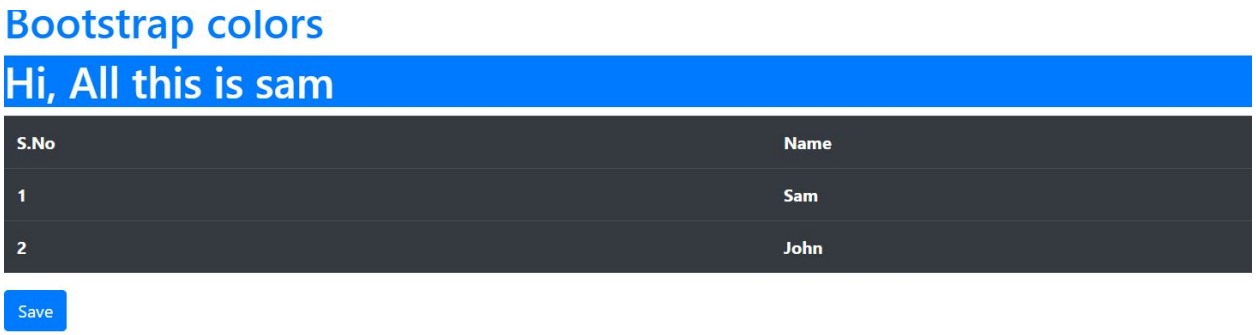
Color code	Color
Primary	blue
Secondary	gray
Info	sky blue
Warning	yellow
Danger	red
Success	green
Light	white (background color is light gray)
White	white(background color is white)
Dark	black

These color classes can apply for text, background ,buttons and tables as contextual classes

Let see some examples:

```
<h1 class="text-primary"> Bootstrap colors</h1>
<h1 class="text-white bg-primary">Hi, All this is sam</h1>
<table class="table table-dark">
    <tr>  <th>S.No</th>   <th>Name</th>  </tr>
    <tr> <th>1</th>  <th>Sam</th>     </tr>
    <tr> <th>2</th>  <th>John</th>     </tr>
```

```
</table>
<button class="btn btn-primary">Save</button>
```



container:

- container is used to wrap the site contents
- There are two container classes.
  .container
  .container-fluid
- The .container class provides a responsive fixed width container

Syntax:

```
<div class="container">
        <!-- Content here -->
   </div>
```

Example for Container:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="/css/bootstrap.min.css">
  </head>
  <body>
    <div class="container">
      <h1>Surya</h1>
      <h2>Narla</h2>
```

```
    </div>

  </body>
</html>
```
O/P:
Surya
Narla

.Container fluid:

- Use .container-fluid for a full width container, spanning the entire width of the viewport.

Syntax:

```data
** **
---content---
** **
```

**Example of container and container fluid:**

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
    <link rel="stylesheet" href="/css/bootstrap.min.css">
  </head>
  <body>

        <h1>Bootstrap container</h1>
        Normal Size

          <button type="button" class="btn
btn-primary">container one</button>
```
```

```
          <button type="button" class="btn btn-danger">container
two</button>
          <button type="button" class="btn
btn-warning">container three</button>
        </p>
       </div>
    </div>
    <div class="container-fluid">
       <div class="col-md-6">
        <h1>Bootstrap fluid container</h1>
        <p>Normal Size</p>
        <p>
          <button type="button" class="btn
btn-primary">container one</button>
          <button type="button" class="btn btn-danger">container
demo</button>
          <button type="button" class="btn
btn-warning">container demo</button>
       </p>
     </div>
  </div>
  </body>
</html>
```

Output:
<img src="/images/container.PNG" alt="container & container-fluid image">

Bootstrap Jumbotron:

- A Bootstrap jumbotron specifies a big box for getting extra attention to some special content or information. It is displayed as a grey box with rounded corners. It can also enlarge the font sizes of the text inside it.
- You can put any valid HTML or other Bootstrap elements/ classes inside a jumbotron.
- The class .jumbotron within the element is used to create a jumbotron.

Jumbotron Inside Container:

- The Inside container is used in a jumbotron, if you want the jumbotron to not extend to the edge of the screen.
- Put the jumbotron inside the .

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootst
rap.min.css">
</head>
<body>

<div class="container">
  <div class="jumbotron">
    <h1>This is Jumbotron inside container!</h1>
    <p>Jumbotron specifies a big box for getting extra attention
to some special content or information.</p>
  </div>
  <p>This is some text.</p>
  <p>This is another text.</p>
</div>

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.
min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstra
p.min.js"></script>
</body>
</html>
```

Output:
<img src="/images/jamboinside.PNG" alt="Jumbotron Inside Container">

Jumbotron Outside Container:

- It is used when you want the jumbotron to extend to the screen edges.
- Put the jumbotron outside the .

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootst
rap.min.css">
</head>
<body>

<div class="jumbotron">
  <h1>This is Jumbotron outside container!</h1>
  <p>Jumbotron specifies a big box for getting extra attention
to some special content or information.</p>
</div>

<div class="container">
  <p>This is some text.</p>
  <p>This is another text.</p>
</div>

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.
min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstra
p.min.js"></script>
</body>
</html>
```

Output:

<img src="/images/jambooutside.PNG" alt="Jumbotron Outside Container">

Full-width Jumbotron:

- To get a jumbotron without rounded borders, you have to add the .jumbotron-fluid class and a .container or .container-fluid inside it.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css
/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.m
in.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.6/umd
/popper.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/b
ootstrap.min.js"></script>
</head>
<body>

<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1>Full-width Jumbotron</h1>
  <p>Jumbotron specifies a big box for getting extra attention
to some special content or information.</p>
</div>

<div class="container">
  <p>This is some text.</p>
  <p>This is another text.</p>
```

```
</div>

</body>
</html>
```

Output:
<img src="/images/jambofullwidth.PNG" alt="Full-width Jumbotron">

Grid system:

Bootstrap Grid System is mobile-first fluid grid system which is made up of a series rows and columns to provide a structure to a website and to place it's content in the intersected areas easily.It's built with flexbox and is fully responsive.

Bootstrap's grid system allows up to 12 columns across the page.Based up on our design we can use 12 columns.

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

Grid Classes:

The Bootstrap 4 grid system has five classes:

- .col- (extra small devices - screen width less than 576px)
- .col-sm- (small devices - screen width equal to or greater than 576px)
- .col-md- (medium devices - screen width equal to or greater than 768px)
- .col-lg- (large devices - screen width equal to or greater than 992px)
- .col-xl- (xlarge devices - screen width equal to or greater than 1200px)

Grid System Rules:

Some Bootstrap 4 grid system rules:

- Rows must be placed within a `.container` (fixed-width) or `.container-fluid` (full-width) for proper alignment and padding
- Use rows to create horizontal groups of columns
- Content should be placed within columns, and only columns may be immediate children of rows
- Predefined classes like `.row` and `.col-sm-4` are available for quickly making grid layouts
- Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and last column via negative margin on `.rows`
- Grid columns are created by specifying the number of 12 available columns you wish to span. For example, three equal columns would use three `.col-sm-4`

Bootstrap Tables:

To add styles to a table use **.table** class within the table tag.If you want you can apply color class for the table.

Example:

```
<table class="table table-dark">
  <thead>
    <tr>
      <th scope="col">S.No</th>
      <th scope="col">FirstName</th>
      <th scope="col">Last_NAme</th>
      <th scope="col">Age</th>
    </tr>
  </thead>
  <tbody>
```

```
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Jukar</td>
      <td>20</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jeff</td>
      <td>bezos</td>
     <td>50</td>
    </tr>

        </tbody>
</table>
```

Output:

S.No	FirstName	Last_NAme	Agee
1	Mark	Jukar	20
2	Jeff	bezos	50

- Use the modifier classes .thead-light or .thead-dark to make <thead>s appear light or dark gray
- Use .table-striped to add zebra-striping to any table row within the <tbody>
- Add .table-bordered for borders on all sides of the table and cells
- Add .table-borderless for a table without borders
- Add .table-hover to enable a hover state on table rows within a <tbody>

- A <caption> functions like a heading for a table. It helps users with screen readers to find a table and understand what it's about and decide if they want to read it

Responsive table:

Responsive tables allow tables to be scrolled horizontally with ease. Make any table responsive across all viewports by wrapping a .table with .table-responsive. Or, pick a maximum breakpoint with which to have a responsive table up to by using .table-responsive{-sm|-md|-lg|-xl}.To get responsive we have to include a table in division with class of .table-responsive.

Example:

```
        <div class="table-responsive">
  <table class="table">

  </table>
</div>
```

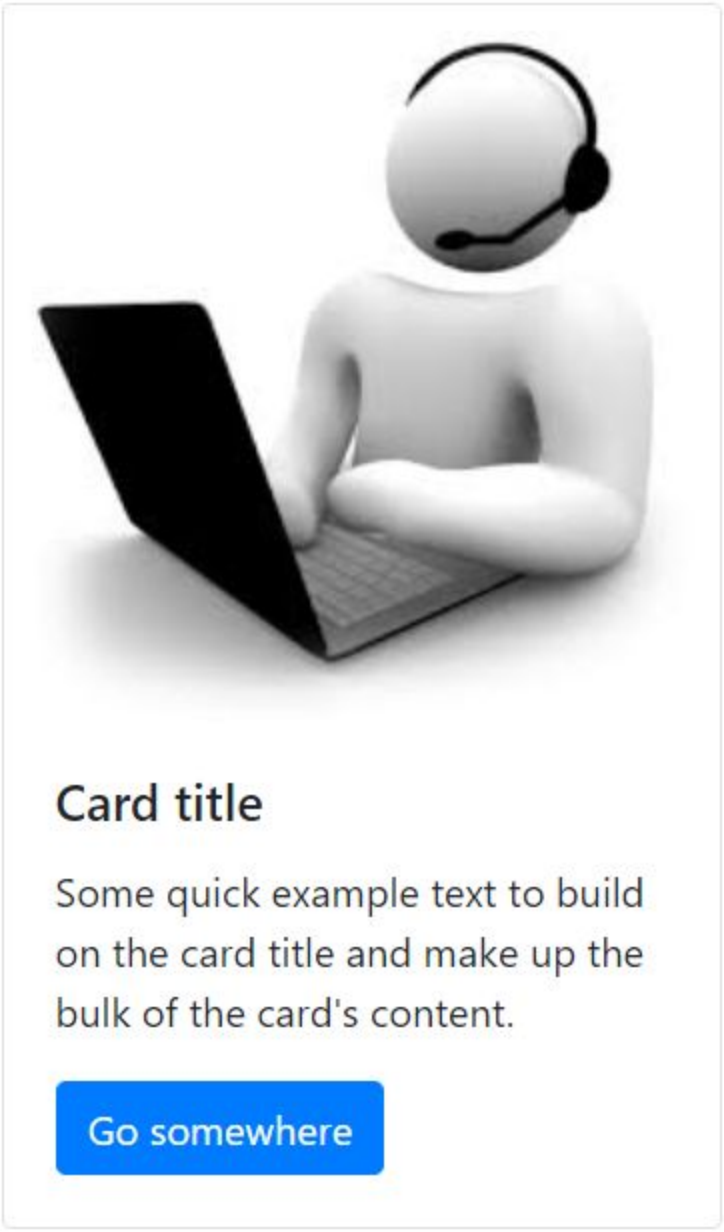
Cards:

A card is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options.

Example:

```
   <div class="card" style="width: 18rem;">
            <img src="images/pic.jpg" class="card-img-top" alt="...">
            <div class="card-body">
               <h5 class="card-title">Card title</h5>
               <p class="card-text">Some quick example text to build on the card title and
        make     up the bulk of the card's content.</p>
              <a href="#" class="btn btn-primary">Go somewhere</a>
            </div>
  </div>
```

Output:



- To align cards in a single row we have to use .card-group class within the tag.Then we have to include multiple cards in that division. Use .card-deck to align cards with some space between each one

Chapter - 4

JavaScript

Introduction:

- JavaScript was Invented by Brendan Eich during has time at Netscape Communication
- JavaScript was originally developed as LiveScript by Netscape in the mid 1990s
- It's first name is Netscape's Mocha. It's not much more popular. At that time Java is most popular programming language
- He decided to change the language name Mocha to JavaScript in 1995, and became an ECMA(European Computer Manufacturers Association) standard in 1997
- Now JavaScript is the standard client-side scripting language for web-based applications and Now it is supported by all web browsers available today, such as Google Chrome, Mozilla Firefox, Apple Safari, etc..,
- JavaScript is the most popular and widely used for client-side scripting language
- Client-side scripting refers to scripts that run within your web browser
- JavaScript is designed to add interactivity and dynamic effects to the web pages by manipulating the content
- It is an object-oriented language, and it also has some similarities in syntax to Java programming language. But, JavaScript is not related to Java in any way

What You Can Do with JavaScript?

- You can modify the content of a web page by adding or removing elements
- You can change the style and position of the elements on a web page
- You can monitor events like mouse click, hover, etc.., and react to it
- You can perform and control transitions and animations
- You can create alert pop-ups to display info or warning messages to the user
- You can validate user inputs before submitting it to the server

Variable:

- Variables are fundamentals to all programming languages
- Variables are used to store the data like strings, numbers etc...

Syntax: var varName = value;

```
var x=10
var y="APSSDC" or 'APSSDC'
var z=true;
```

General Rules for constructing variable names are:

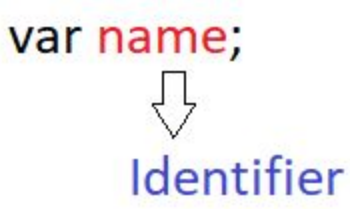
- A variable name must start with a letter, underscore (_), or dollar sign ($)
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters (A-z, 0-9) and underscores (_).
- A variable name cannot contain spaces
- Variable names are case sensitive
- In JavaScript, variables can also be declared without having any initial values assigned to them. This is useful for variables which are supposed to hold values like user inputs

Example:

```
// Declaring Variable
var name;
```

JavaScript Identifiers :

- All JavaScript variables must be identified with Unique names(Identifiers)



Example:

```
// Assigning value
userName = "APSSDC";
```

- **Note:** In JavaScript, if a variable has been declared, but has not been assigned a value explicitly, is automatically assigned the value undefined.

- JavaScript no print predefined functions available. If we print anything we can use console.log(<variable name>)

Declaring Multiple Variables At Once:

- In addition, you can also declare multiple variables and set their initial values in a single statement. Each variable are separated by commas, as demonstrated in the following example:

Example:

```
// Declaring multiple Variables
var name = "Peter Parker", age = 21, isMarried = false;

/* Longer declarations can be written to span
multiple lines to improve the readability */
var name = "APSSDC",
age = 21,
isMarried = false;
```

- let & const for declaring variables & it's ES6 properties
- Unlike var, which declare function-scoped variables, both let and const keywords declare variables, scoped at block-level ({}), Block scoping means that a new scope is created between a pair of curly brackets {}.

Example:

```
// let
let a=10
 //Declaring constant
const PI = 3.14;
console.log(PI); // 3.14

// Trying to re-assign
PI = 10; // error
```

Generating Output in JavaScript:

- In JavaScript there are several different ways of generating output including writing output to the browser window or browser console, displaying output in dialog boxes, writing output into an HTML element, etc.,

Example:

```
console.log("hello");
// Displaying o/p alert Dialog Box
alert("Hello World!"); // Outputs: Hello World!
// Displaying o/p to  the browser window
document.write("Hello World!"); // Prints: Hello World!
```

Data Types:

- datatypes are majorly two types of languages
  - Statically(c,c++, Java)

    ```
    int x=1
    str y="rajesh"
    ```
  - Dynamically(JS, Python, Ruby etc.,)

    ```
    var x=1
    ```

- `var y="rajesh"`
  - Datatypes are basically type of data that can be used and manipulated in program
  - The latest ES6 has 7 data types. In those 7 data types 6 data types are primitive(predefined). i.e.,
    - **Numbers:** The number data type is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. 1.5e-4

      ```
      var a = 25;         // integer
      var b = 80.5;       // floating-point number
      var c = 4.25e+6;    // exponential notation, same as 4.25e6 or 4250000
      var d = 4.25e-6;    // exponential notation, same as 0.00000425
      ```

    - The Number data type also includes some special values which are: Infinity, -Infinity and NaN(Not a Number).

      ```
      alert(16 / 0);  // Output: Infinity
      alert(-16 / 0); // Output: -Infinity
      ```

```
alert("Some text" / 2);       // Output: NaN
```

- **String:** The string data type is used to represent textual data. it's created using single/double quotes

```
var a = "Let's have a cup of coffee."; // single quote inside double quotes
var b = 'He said "Hello" and left.';  // double quotes inside single quotes
var c = 'We\'ll never give up.';     // escaping single quote with backslash
```

- **Boolean:** The Boolean data type can hold only two values: true or false. It is typically used to store values like yes (true) or no (false), on (true) or off (false), etc.

Example:

```
var a = 2, b = 5, c = 10;
alert(b > a) // Output: true
alert(b > c) // Output: false
```

- **Undefined:** If a variable has been declared, but has not been assigned a value, has the value undefined

Example:

```
var a;
var b = "Hello World!"

alert(a) // Output: undefined
alert(b) // Output: Hello World!
```

- **Null:** A null value means that there is no value. It is not equivalent to an empty string ("") or 0, it is simply nothing

Example:

```
var a = null;
alert(a); // Output: null
```

- **Object:** The object is a complex data type that allows you to store collections of data(it's key-value pair).

        - It has the most important data type and forms the building blocks for modern JS. we'll learn about this in future concepts

```
var car = {
  modal: "AUDI",
 color: "white",
 doors: 5 }
```

    - **Array:** An array is a type of object used for storing multiple values in a single variable. Each value (also called an element) in an array has a numeric position, known as its index

Ex:

```
var colors = ["Red", "Yellow", "Green", "Orange"];
```

    - **Function**: The function is a callable object that executes a block of code. Since functions are objects, so it is possible to assign them to variables

Example:

```
var greeting = function(){
return "Hello World!";
}
alert(greeting());
function ShowMessage() {
        alert("Hello World!");
        }

   ShowMessage()=> {
                alert("Hello World!");
        }
```

- **Typeof Operator:** The typeof operator can be used to find out what type of data a variable or operand contains. It can be used with or without parentheses (typeof(x) or typeof x)

Example:

```
a=5
typeof(a);  // Returns: "number"
```

How to find the length, sorting of array in JS?

```
console.log(<variable name>.length)
```

Example:

```
Let a =['apssdc','rajesh','ravi']
    console.log(a.length)    //3
    length is used to return the no of elements available in array
    console.log(a.sort()) // ['apssdc','rajesh','ravi']
    sort() is used to sorting the array values
    console.log(a[0])    //apssdc
    console.log(typeof(a))  //Object
```

- If we can access all array values at a time use *for loop*

Example:

```
for (i in a){
console.log(a[i])
}
```

Conditional Statements:

- The conditional statements included in the JS code assist with decision. Based on certain conditions. The conditional Statements are several types they are:
  - if
  - else
  - else if
  - switch
- **if:** It executes a specified code segment if the given condition is True

Syntax:

```
If(condition){
        //code execution
}
```

Example:

```
var age=25;
if(age=>25){
console.log("valid");
```

```
}
```

- **Else:** When the condition inside the if statement is false, the code associated with the else statement gets executed

  Syntax:

```
if (condition){
//code to get executed when the condition is true
}
else{
//code to get executed when the 'if' condition fails
}
```

  Example:

```
var age=25;
if(age<25){
console.log("valid");
}
else{
console.log("not valid");
}
```

- **Else if**: When the first condition fails, the else-if allows the program to specify some new condition(s).

  Syntax:

```
if (condition_one){
//code to execute if condition_one is true
}
else if (condition_two){
//code to execute if the condition_one fails but the
condition_two is true
}
else{
//code to execute if the condition_one is false
}
```

Example:

```
if (marks > 90 && marks <= 100){
grade = 'A+';
}
else if (marks > 80 && marks <= 90){
grade = 'A';
}
else if (marks > 70 && marks <= 80){
grade= 'B';
}
else{
grade='fail';
}
```

- **Switch**: The switch case statement has an expression which is compared with values of each case statement and if a match is found, the associated code is executed.
    - *Syntax*:

```
switch(expression){
case a:
//block of code
break;
case b:
//block of code
break;
```

Example:

```
var d = new Date();
switch(d.getDay()) {
    case 0:
        alert("Today is Sunday.");
        break;
    case 1:
        alert("Today is Monday.");
        break;
    case 2:
        alert("Today is Tuesday.");
```

```
            break;
        case 3:
            alert("Today is Wednesday.");
            break;
        case 4:
            alert("Today is Thursday.");
            break;
        case 5:
            alert("Today is Friday.");
            break;
        case 6:
            alert("Today is Saturday.");
            break;
        default:
            alert("No information available for that day.");
            break;
}
```

Looping Statements:

- Loops are used to execute the same block of code again and again, as long as a certain condition is met
- Loop is used to automate the repetitive tasks within a program to save the time and effort
- JavaScript now supports five different types of loops:
    - while
    - do while
    - for
    - for ..in
    - for ..of
- **while:** The while loop, loops through a block of code as long as the specified condition evaluates to true. As soon as the condition fails, the loop is stopped
    - *Syntax*:

```
while(condition) {
// Code to be executed
}
```

Example:

```
var a=1
while(a<=10){
console.log(a);
a++;
}
```

- **do while:** The do-while loop is a variant of the while loop, which evaluates the condition at the end of each loop iteration. With a do-while loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

syntax:

```
do {
// Code to be executed
}
while(condition);
```

Example:

```
var a=1
do{
console.log(a);
a++;
}
while(a<=10);
```

What is the difference b/w while & do-while?

- The while loop differs from the do-while loop in one important way — with a while loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed
- With a do-while loop, on the other hand, the loop will always be executed once even if the conditional expression evaluates to false, because unlike

the while loop, the condition is evaluated at the end of the loop iteration rather than the beginning

- **For:** The for loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for a certain number of times. Its syntax is:

  syntax:

  ```
  for(initialization; condition; increment) {
  // Code to be executed
  }
  ```

  Example:

  ```
  for(var i=1; i<=5; i++) {
     console.log("Iteration "+i);
  }
  ```

- **for ..in:** Loops through the properties of an object or the elements of an array.
  - *Syntax*:

    ```
    for(variable in object) {
    // Code to be executed
    }
    ```

    Example:

    ```
    var data=['apssdc','apssdc1','apssdc2','apssdc3']
    for (i in data){
    console.log(data[i]);
    }
    //for( var a in data){
        here getting index values
    // }
    ```

- **for ..of**: ES6 introduces a new for ..of loops over iterable objects such as arrays, strings, etc., Also, the code inside the loop is executed for each element of the iterable object

Syntax:

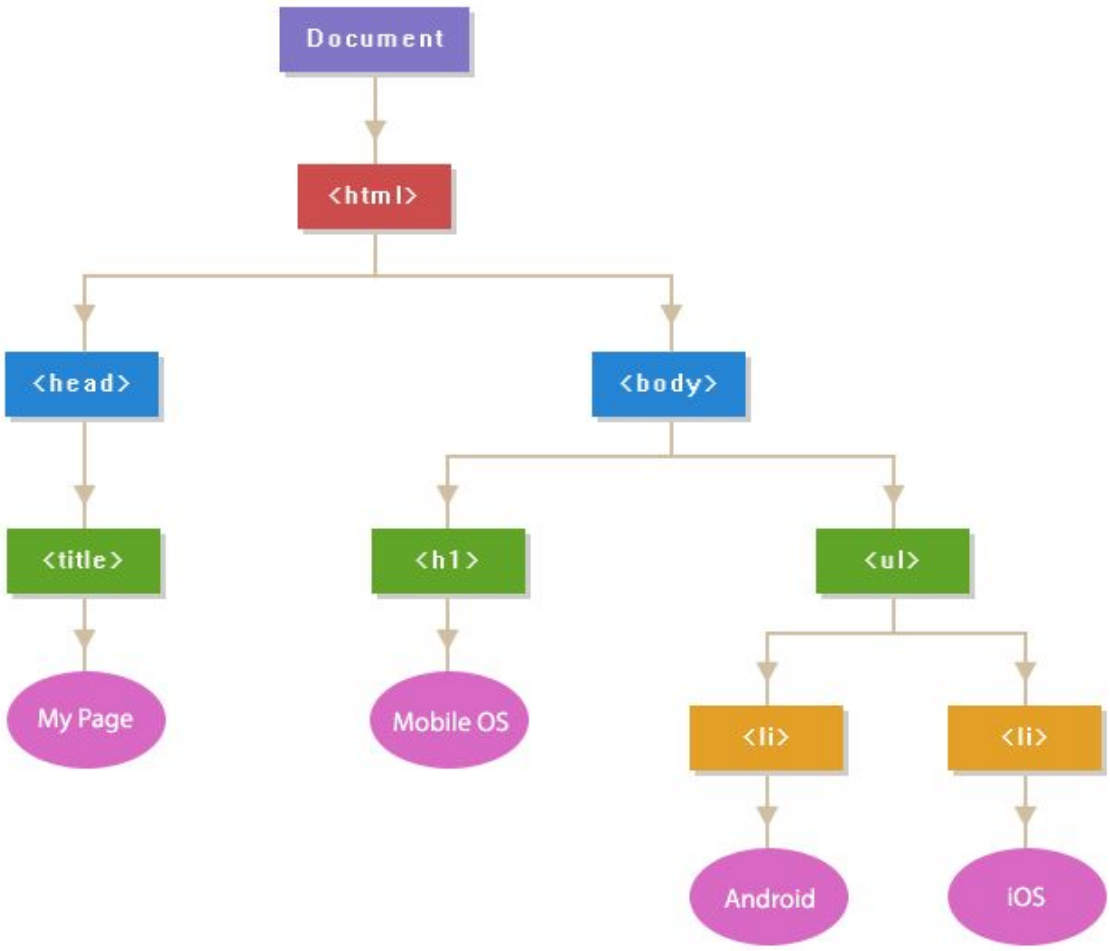
```
for(variable of object) {
// Code to be executed
}
```

Example:

```
var a="APSSDC Welcome's you"
for (i of a){
console.log(i);  //A,P,S,S,D,C, ,W,e,l,c,o,m,e,',s, ,y,o,u
}
```

DOM (Document Object Model):

- The Document Object Model, or DOM for short, is a platform and language independent model to represent the HTML or XML documents
- It defines the logical structure of the documents and the way in which they can be accessed and manipulated by an application program
- In the DOM, all parts of the document, such as elements, attributes, text, etc. are organized in a hierarchical tree-like structure; similar to a family tree in real life that consists of parents and children
- DOM Tree depends on HTML
- Example:



- The above diagram demonstrates the parent and child relationship between the nodes
- <head> and <body> elements are the child nodes of <html>. It's a parent node
- The <head> and <body> elements are also siblings since they are at the same level
- The text content inside an element is a child node of the parent element. For example, *Mobile OS* is considered as a child node of the <h1>
- HTML attributes such as id, class, title, style, etc. are also considered as nodes in DOM hierarchy but they don't participate in parent/child relationships like the other nodes do
- Each element in an HTML document such as image, hyperlink, form, button, heading, paragraph, etc. is represented using a JavaScript object in the DOM hierarchy

- Each object contains properties and methods to describe and manipulate these objects

How to call JavaScript into html page?

- use <script> tag in html
- If you are using internal script in the same html file use like this <script><write your code here></script>
- If you are using external script to get into html use like this

  <script src="<external file name> ex: main.js"></script>

DOM Manipulations:

```
<body id='root'><body>

<script>
var root=document.getElementById('root');
   var h2=document.createElement("h2"); // creating h2 element
   h2.textContent="Rajesh APSSDC"; //add some text
   root.appendChild(h2);
   </script>
```

- The above example we are creating heading 2 tag with some text on that it's happened to the root *ID*
- i want to add a class/id to the h2 tag

```
h2.classList.add("<classname>"); //add class name
h2.setAttribute("id/class","<id/class name>"); // add either class name or Id name
```

- If you want to apply some styles (like text color etc.,) to the h2 element you can use like this

  ```
  h2.style.color='red';
  ```

- i want creating one paragraph tag in html page and write some text in paragraph tag using javascript as shown below

```
<p id='p'></p>
```

document.querySelector(".p").innerHTML="Hello APSSDC";

How many ways to get html tag/class/id names in JS?

document.getElementById(<idname>)
document.getElementByClassName(<classname>)
document.getElementByTagName(<tag name>)
document.querySelector("<.idname>/<.classname>/<tagname>")
document.querySelectorAll("<.idname>/<.classname>/<tagname>") //Its used for getting multiple id/class/tag with same name

BOM(Browser Object Model):

- The Browser Object Model (BOM) in JavaScript includes the properties and methods for JavaScript to interact with the web browser
- BOM provides you with window objects, for example, to show the width and height of the window. It also includes the window.screen object to show the width and height of the screen
- JS BOM has types Window, Screen, Location, History, Dialog Box, Timer

document.write("Screen width: " + screen.width); //display the screen width
document.write("
Screen width: " + screen.height); //display the screen height
document.write("The URL of this page is: " + window.location.href); // it's used to find the exact page reference link

- alert is used for display some alert message like popup box alert("Hello apssdc")
- The prompt dialog box is used to prompt the user to enter information. A prompt dialog box includes a text input field, an OK and a Cancel button.

prompt("What's your name?");

- A timer is a function that enables us to execute a function at a particular time
    - Using timers you can delay the execution of code so that it does not get done at the exact moment an event is triggered or the page is loaded. For example, you can use timers to change the advertisement banners on your website at regular intervals, or

display a real-time clock, etc. There are two timer functions in JavaScript: setTimeout(function, milliseconds) and setInterval(function, milliseconds)

- if you are playing x game. When the game is over the page would be restart/refresh

```
function gameover() {
   window.location.reload(true);
}
```

```
window.history.back(); // its used for get back to previous page
window.history.go(0); // its used to reload the current page
window.history.forward(); // its used to navigate for next page
```

- If you attempt to access the page that does not exist in the window's history then the methods back(), forward() and go() will simply do nothing

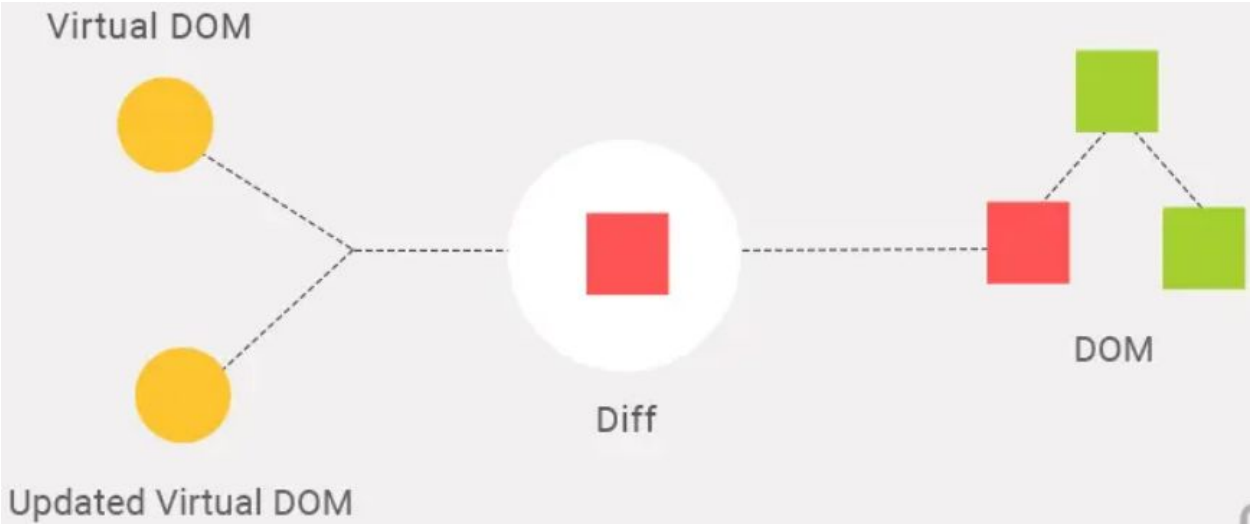
Chapter - 5

ReactJS

Hello there in this video we will discuss the React introduction concept. First of all we've to know what ReactJS is.

- **What is React ?**
  - Very often **React** is misinterpreted as a tool, framework,language.
  - But React is a JavaScript Library to build user interfaces.
  - React is created by facebook as an open source project.
  - ReactJs is extremely fast, credit goes to React's virtual DOM.
  - ReactJs is Componentbased, In a website everything is a component.
  - Skills learned with React can be applied to mobile App development with React Native.

- **What is VIRTUAL DOM ?**
  - Lemme revise the concept of DOM once to understand the VDOM better.
  - The DOM is a tree-like structure of various elements. This structure may start from the basic building blocks such as **HTML,HEAD,BODY.** It's an API for HTML and XML documents and defines the way a document is accessed and modified.
  - DOM manipulation is the primary goal for Interactive web. But the limitation is very slow.
  - Because most js frameworks update the DOM much more than they have to.
  - To address this problem, React popularized something called **Virtual DOM.**
  - The VDOM is the virtual representation of the Real DOM. i,e for every DOM object, there is a corresponding virtual DOM object.
  - But it directly changes whats on screen. Manipulating DOM is slow but manipulating virtual DOM is fast.
  - When the virtual DOM gets updated. React compares the new VDOM with a pre-update version, React figures out exactly which VDOM objects have changed this process is called **diffing.**
  - Once React knows which VDOM objects have changed. It updates only those objects on the real DOM and other unnecessary elements won't get updated as opposed to what the DOM normally does.



- **How to install react ?**
  For installing React in our systems, we need to have node js installed in our machines.
  For downloading node js we've to visit www.nodejs.org.
  After the installation procedure for node completed we have to run the following command to install reactjs
  `npm install create-react-app -g`

  After the installation process completed, we've to run another command to install react project
  `create-react-app <project_name>`

- **Components in React**
  Components are the basic building blocks of react. Conceptually components are like javascript functions.
  - Components give us a way to split the UI into independent reusable pieces and think about each piece in isolation.
  - They accept arbitrary inputs (Props) and return React elements describing what need to appear on the screen.
  - Components are of 2 kinds:
    - Functional components and
    - Class components

    - Functional components (Example)

```
import React from 'react';
let Sample=()=>{
    return(
        <div>
            <h2> This is a functional component. </h2>
        </div>
    )
}
export default Sample;
```

- HTML attributes. The component receives the argument as a prop object.
- React props are read-only! You will get an error if you try to change their value.Class Components (Example)

```
import React from 'react';
export default class Sample extends React.Component{
    render(){
        return(
            <div>
                <h2> This is a class component. </h2>
            </div>
        )
    }
}
```

- **JSX**
  - This is one of the features of ReactJs. It has a funny tag like syntax. It's neither a string nor HTML. This is a syntax extension to JavaScript.
  - React keeps both HTML and JavaScript together in the same file.
  - We need not call the HTML elements for interactive manipulations.
  - JSX may remind you of a template language, but it comes with full power of JavaScript.

```
import React from 'react';
export default class Sample extends React.Component{
    render(){
        var name="Hanuman";
        return(
            <div>
                <h2> My name is {name} </h2>
            </div>
        )
    }
}
```

- **Props in React**
    - Props are the properties for a component. The concept of props allows us to pass information between the components.
    - The props are like function parameters in javascript and also the attributes in HTML. i,e to send props into a component, use the same syntax as

    - Accessing props in functional components
    - Accessing props in class components

States:

- It's one of the most important concept in reactJS
- Each and every component has maintain it's own private states
- States are mutable

What is the difference between Props & States?

Props	State

• it get passed to the component • it's a functional parameter • parent usually passes the data the child component but it's an immutable • in functional component we can use → props • class based component we can use → this.prop	• managed within the component • we can declared variables in the function body • it has managed within the component and the component has full control to change it • state has a another in the functional component we use → useState Hook ○ This.state → class component

step1: create a state object and initialize it. It's usually done in **constructor()**

within the constructor we called **super()** it is required because we are extended the component to the base class constructor

```
constructor(){
    super()
    this.state={
    message: "Rajesh"
    }
    }
```

step-2: After creating states you can bind the render function { }

Step-3: add a button. When we click on a button I need to change the text.

- <button>Click Me </Button>

We add onClick={()=>this.change()}

after constructor

Add the change() after the constructor()

change(){

this.setState({

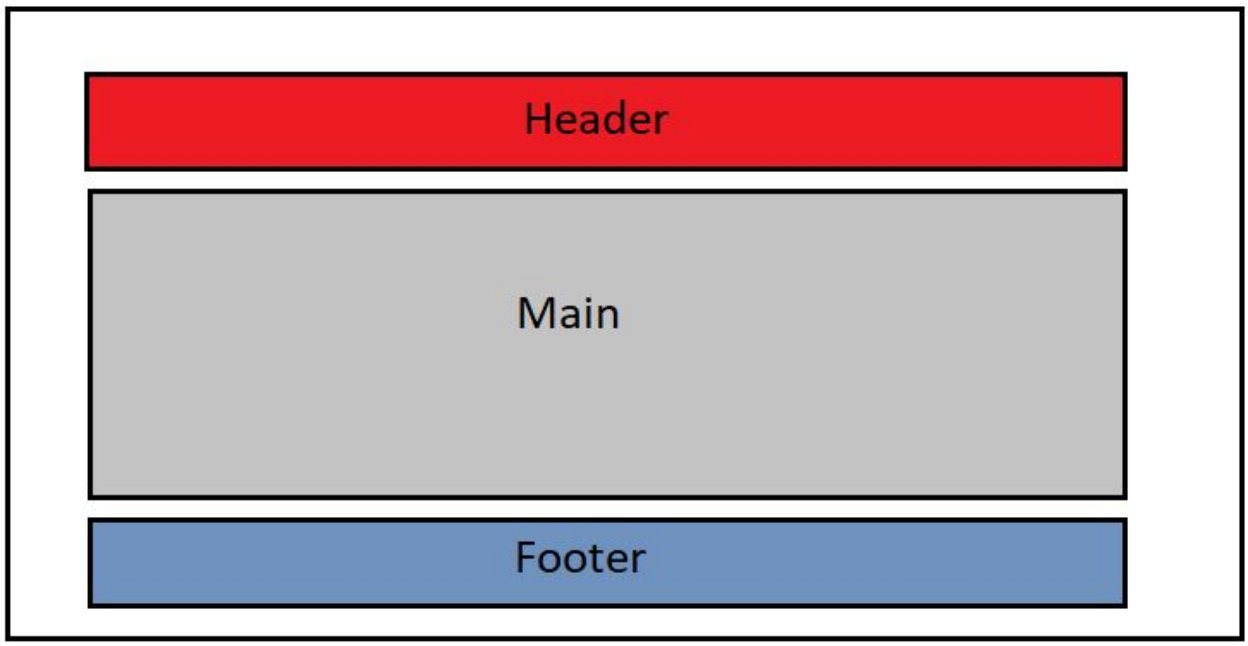
message:”hello”

})

setState:

It is used to change the state in class based components

Example:


Header
Main
Footer