Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

Skill AP
Learn Anytime Anywhere
Andhra Pradesh State Skill Development Corporation

Web designing using ReactJs

Hyper Text Markup Language

Web design vs Web development.

- In essence, web design refers to both the aesthetic portion of the website and it's usability. Web designers use various design programs such as Adobe Photoshop to create the layout and other visual elements of the website.
- Web Developers on the other hand, take a website design and actually make a functioning website from it. Web developers use HTML, CSS, JavaScript, PHP and other programming languages to bring to life the design files.

  Web developers are of 3 kinds:
- Front-end developer
- Back-end developer &
- Full stack developer

1. Front-End Developer (FED):

Front End typically refers to what you actually see on the website in the browser (and is often called "client-side"). This means that front end developers are responsible for everything that you see when you're navigating around the Internet, from fonts and colours to dropdown menus and sliders.

Skills needed: JavaScript, HTML, CSS, jQuery, HTML5, CSS3, Ajax, UI (User interface), UX (user experience), JavaScript Frameworks such as Angular-s, Angular.js, ReactJS, vue.js, Ember.js and Meteor.js

2. Back-End Developer

While the front end is everything the user interacts with directly, "Back End" typically refers to the guts of the application which live on the server (and is often called "server-side"). The back end of a website consists of a server, an application, and a database. Back end developers generally work with front end developers to make their code work within the site's design (or to tweak that design when necessary) and user interface.

Skills needed: JavaScript, HTML, CSS, jQuery, SQL, Java, Linux, Python, Ruby, PHP

3. Full-stack developer

Full Stack web development is a combination of both the front-end and back-end. "Stack," means layer in this case. A full-stack web developer has expertise in all layers of a website's development. This includes, but not limited to: the server, client and hosting, a form of data structuring or modelling, user interface and experience, as well as the needs of the actual business.

Skills needed: JavaScript, HTML, CSS, jQuery, PHP, SQL, Java, HTML5, Ajax, CSS3, Linux, Python, Ruby, UI (User interface)

HTML

1.1 Introduction to HTML5:

1.11 What is HTML?

- **HTML:** stands for Hypertext Mark-up Language
- HTML is a mark-up language used to create web pages (either static or dynamic).
- 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)

Before going to creation let us know about normal web pages.

In general, the HTML structure consists of 3 parts.

- HTML version information
- Head &
- Body

HTML version information:

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

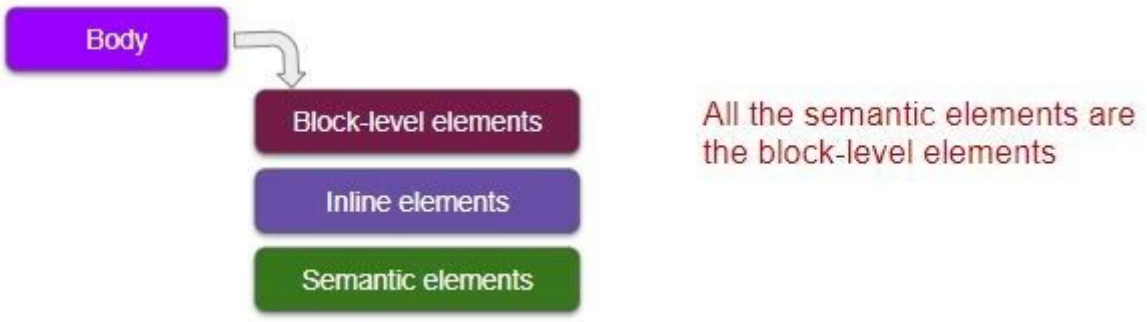
Head:

- In the head of web page contains a title, icon, Meta tags, Styles & Some Scripts related to styles
- The head part content is not visible on the screen. It's works on background of the webpage/website

Body:

A. The body section can be devices into 3 major parts/sessions

1. Block-level elements
2. Inline elements
3. Semantic elements



B. In the web browser entire visible text is written the body session

- Before Going to learn web development we need one of the editors like Visual Studio Code, Atom, Sublime Text, etc..,
- Without an editor we can't able to implement our website
- Html language uses a series of elements to display the content in a web page.
- Any text enclosed between less than (<) and greater than signs (>) is an HTML tag **<html>**
- Most of the HTML tags require a start Tag and an end tag, end tag differs with the opening tag by a slash "/"

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>
```

Open Text Editor and create a folder inside our folder create 3 more folders

Example: Web design (Root folder)

Inside the Web design folder create 3 folders with names CSS, images, JS (In the CSS folder we will store all .CSS files, in the image folder will store all static images, in the JS folder will store all JavaScript files).

After creation of these folders it's time to create the MainSource file i.e. index.html file.
(index is the filename and .html is the file extension; all html files will save with .html extension)

Why is the index.html file the main source file of our project?

Index is the most common name used for the default pages shown in the website. If we mention the name index.html in our website. The browser is considered as a default page. We can say that the index page is as the root file.

Basic Tags in Html:

```
<!DOCTYPE html>
<html lang=" en-US ">
```

These 2 tags are basic and primary tags

By using these tags even a disabled person can understand the document type is html, language we are using is English and the version is 5.

Head part:

Actions we are able to perform in head part:

- Able to give title to our webpage
- Able to keep an icon as favicon
- Allows us to use external CSS and manifest files.
- Moreover, we can use scripting languages by using script tags.
- Provides the way to use meta elements.

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

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

Let us give a title and favicon to our webpage.

To give title the syntax is:

<title> any title </title> (title tag is used to keep a title to our webpage)

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.

Raster Images	Vector images
They are composed of pixels	They are composed of paths
Raster images occupies more space which depends on the quality	Vector images occupies less space
Cost of Raster images is low	Vector graphics cost more as compared to Raster images
The quality of image decreases when we zoom in it.	The quality remains same when we zoom in it.
Load time is very low than Vector images	Load time is fast as they occupies less storage range.
Ex: GIF, BMP, JPG,TIF, etc..,	Ex: SVG,EPS,PDF,AI,etc..,

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

Syntax to keep an image as favicon to our webpage:

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

Link attribute is used to navigate or link

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

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

Let us learn about different types of elements in Html5:

1. Block-level Elements:
   The element which occupies the complete width of the screen is known as Block-level Elements.

1. All heading tags (h1 to h6), paragraph tag (p)
2. All semantic elements
3. Form tag

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:

```
<p> content </p>
```

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, contact 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:

```
<span> content </span>
```

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>
```

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> (anchor tag)

- Inbound navigation
- Outbound navigation
- Mailto
- Tel

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:

```
<a href="tel: +91 9123456789"> phone </a>
```

Form controls

The <form> element formally defines a form and attributes that determine the form's behavior. Each time you want to create an HTML form, you must start it by using this element, nesting all the contents inside. Many assistive technologies and browser plugins can discover <form> elements and implement special hooks to make them easier to use.

It's always possible to use a form control outside of a <form> element. If you do so, by default that control has nothing to do with any form unless you associate it with a form using the form attribute. This was introduced to let you explicitly bind a control with a form even if it is not nested inside it.

Web forms also called Fill-out Forms, let a user return information to a web server for some action. The processing of incoming data is handled by a script or program written in language that manipulates text files and information. Form element is used to delimit a data input form. There can be several forms in a document, but form elements can not be nested. The forms are not hard to code. These follow the same constructs as other HTML elements.

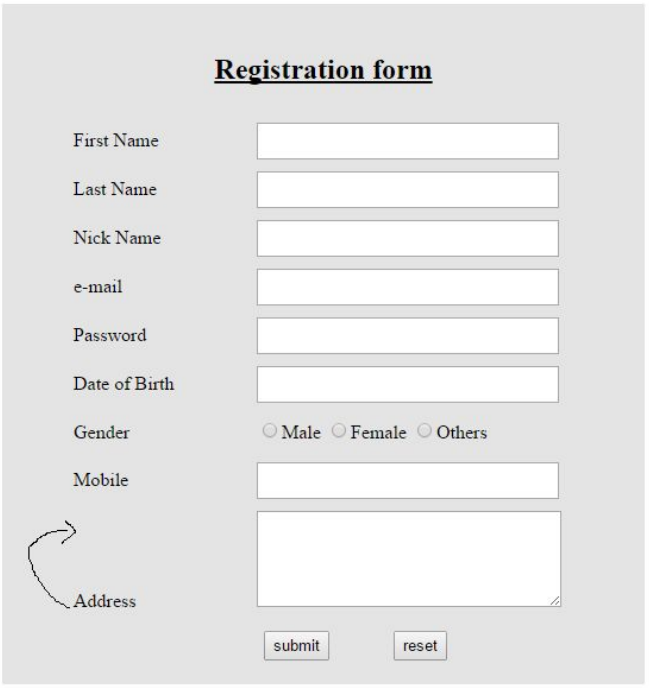
Element has the following attributes: METHOD- The method can be set to either POST or GET. The difference between the two has to do with the way the information is sent to the asp program. Always use the POST method. ACTION- The action attribute is URL specifying the location where the contents of the form are to be submitted to elicit a response. If the ACTION attribute is missing, the URL of the document itself is assumed. This tells the form, what program should be executed by the server when the form's data is submitted. The path to your asp file will vary from system to system. NAME attribute is used to refer to the form when working with controls in it from JavaScript. Despite their names, Forms are not visible objects on the screen, it is purely a logical concept to hold the controls together. Target specifies the name of the frame to put the results in. ENCTYPE This attribute specifies how the data is to be encoded. It sets the MIME type used to encode the name/value pair when sent to the action URL. This attribute is applied only if POST method is used. The default value is “application/X-www-form-urlencoded”. For the File control it has the value “multipart/form-data” These tags start and end a form (all input fields of the form are placed between these two tags). METHOD specifies which technical protocol the web server will use to pass the form data to the program which processes it (always set it to POST), and ACTION tells the server exactly which program that is. Note: POST must be capitalized, otherwise the method defaults to "GET".

GET sends the information entered in the form to the server at the end of the URL. Get is the default method, a question mark (?) and the form data is appended to the target URL of the code on the server and this complete variable is called query-string, which is the environment variable of the server. If there are two text fields named name and id on a form handling data HANUMAN KUMAR and 100 respectively, then the data from these text fields would be appended to the URL like this Filename.asp?name=HANUMAN+KUMAR id=100 But GET method has disadvantage from security point of view as it shows all the values entered by the user POST This method is preferred over the GET method as it sends the contents of the form as a data block through the standard input stream using the http header.

By using form controls we can get the required data from the user. For defining a form, we have to use <form></form> element. We have different form-controls in HTML. Those are

- input
- output
- Textarea
- select
- Label

- Button
- Fieldset
- Optgroup
- **`input`**



We can use the input element in several ways. The type of input element is depending on the type attributes. The corresponding values for the type attribute is

- Text
  - *Getting text format of data*

    `<input type="text" name=? size=?>`
- Email
  - *Sets a single-line textbox for email addresses*

    `<input type="email" name=?>`
- Password
  - *Getting data in the format of password*

    `<input type="password" name=? >`
- Search
  - *For search inputs*

    `<input type="search" name=?>`
- Checkbox
  - *For multiple selection*

    `<input type="checkbox" name=? value=?>`

- Radio
  - *For only one selection*

    `<input type="radio" name=? value=? checked>`

- Color
  - *Provide a way to collect colors*

    `<input type="color" name=?>`

- Date
  - *Getting date information*
- Time
  - *For getting time information*

    `<input type="date/month/week/time" name=?>`

- Url
  - *Sets a single-line textbox for URLs*

    `<input type="url" name=?>`

- Number
  - *Sets a single-line textbox for a number*

    `<input type="number" name=?>`

- Range
  - *Sets a single-line text box for a range of numbers*

    `<input type="range" name=?>`

- Reset
  - *For resetting the form*

    `<input type="reset">`

- **`<textarea>`**

  Creates a text box area. Columns set the width; rows set the height.

  `<textarea name=? cols="x" rows="y"></textarea>`

- **`<select>`**

  Creates a scrolling menu. Size sets the number of menu items visible before the user needs to scroll.

  `<select multiple name=? size=?> </select>`

  `<select name=?> </select>` (Creates a pulldown menu)

- **`<option>`**

Sets off each menu item for select

```
<select>
    <option selected disabled> Select city </option>
    <option> Srikakulam </option>
    <option>  Vizianagaram </option>
    <option>  Visakhapatnam </option>
</select>
```

By using selected we can make the option selected by default. By using disabled, we can disable the selection from dropdown.

- **`<label>`**

Provide a label for the text

```
<label> Sample Label </label>
```

We can apply label for the radio button and checkbox also

```
<label>  <input type="radio" /> Sample Label </label>
```

- **`<button>`**

Creates a submit button. Value sets the text in the submit button.

```
<button type="button"> Click Me </button>
```

- **`<fieldset>`**

The HTML <fieldset> element is used to group several controls as well as labels (<label>) within a web form.

```
<form>
  <fieldset>
    <legend>Choose your favorite monster</legend>

    <input type="radio" id="kraken" name="monster">
    <label for="kraken">Kraken</label><br/>

    <input type="radio" id="sasquatch" name="monster">
    <label for="sasquatch">Sasquatch</label><br/>

    <input type="radio" id="mothman" name="monster">
    <label for="mothman">Mothman</label>
  </fieldset>
</form>
```

The <fieldset> element provides a grouping for a part of an HTML form, with a nested<legend> element providing a caption for the <fieldset>. It takes few attributes, the most notable of which are form, which can contain the id of a <form> on the same page, allowing you to make the <fieldset> part of that <form> even if it is not nested inside it, and disabled, which allows you to disable the <fieldset> and all its contents in one go.

- **<datalist>**

The HTML <datalist> element contains a set of <option> elements that represent the permissible or recommended options available to choose from within other controls.

```
<label for="ice-cream-choice">Choose a flavor:</label>
<input list="ice-cream-flavors" id="ice-cream-choice" name="ice-cream-choice" />

<datalist id="ice-cream-flavors">

  <option value="Chocolate">
  <option value="Coconut">
  <option value="Mint">
  <option value="Strawberry">
  <option value="Vanilla">

</datalist>
```

- **<output>**

The HTML Output element (<output>) is a container element into which a site or app can inject the results of a calculation or the outcome of a user action.

```
<form oninput="result.value=parseInt(a.value)+parseInt(b.value)">

 <input type="range" id="b" name="b" value="50" /> +
 <input type="number" id="a" name="a" value="10" /> =
 <output name="result" for="a b">60</output>

</form>
```

- **<reset>**

<input> elements of type reset are rendered as buttons, with a default click event handler that resets all of the inputs in the form to their initial values.

```
<form>
  <div class="controls">
    <label for="id">User ID:</label>

    <input type="text" id="id" name="id" />
    <input type="reset" value="Reset">
    <input type="submit" value="Submit">
```

```
    </div>
</form>
```

- **<optgroup>**

  The HTML <optgroup> element creates a grouping of options within a <select> element.

```
<optgroup label="Sauropods">

      <option>Diplodocus</option>
      <option>Saltasaurus</option>
      <option>Apatosaurus</option>

   </optgroup>
```

Styling fonts in HTML

If you want to apply styles to a specific element, we have to select it first. We can apply styles by using HTML elements also. We have few elements for applying styles in HTML. Those are

- **Strong** (   ) : For getting bold text
- **Italic** (   ) : For italic text
- **Underline** (<u> </u>): For underlined text

Multimedia elements

There were nearly negligible media elements in previous versions of HTML, yet those were still very helpful and handy. With the rise of HTML5, several media elements have also come into the light.

These media elements will change the entire scenario of designing and development using HTML. This article will provide an introduction and a little description of these elements.

We have two multimedia elements in HTML

- Audio

  The <audio> tag is used to embed sound content in a document, such as music or other audio streams.

- Video

  The <video> tag is used to embed video content in a document, such as a movie clip or other video streams.

For audio/video we have to use child element (<source> </source>) for representing source.

IFrames

The HTML Inline Frame element (<iframe>) represents a nested browser content embedding another HTML page into the current one. By using iFrames we can embed youtube videos also.

```
<iframe id="inlineFrameExample"
    title="Inline Frame Example"
    width="300"
    height="200"
    src="https://www.google.com">
</iframe>
```

Attributes

We can use the following attributes for better results.

- **Height**

  The height of the frame in CSS pixels. Default is 150.

- **Allow**

  Specifies a feature policy for the <iframe>. The policy defines what features are available to the <iframe> based on the origin of the request (e.g. access to the microphone, camera, battery, web-share API, etc.).

  For more information and examples see: Using Feature Policy > The iframe allow attribute.

- **Width**

  The height of the frame in CSS pixels. Default is 150.

- **Src**

  The URL of the page to embed. Use a value of about:blank to embed an empty page that conforms to the same-origin policy. Also note that programmatically removing an <iframe>'s src attribute (e.g. via Element.removeAttribute()) causes about:blank to be loaded in the frame in Firefox (from version 65), Chromium-based browsers, and Safari/iOS.

- **Allowfullscreen**

  Set to true if the <iframe> can activate fullscreen mode by calling the requestFullscreen() method.