Full-Stack Web Development

Those who learn this course, there is a chance to become a Full-stack Web Developer…

**What does a Full-Stack Web Developer Does?**

A Full Stack Web Developer is a person who can develop both client and server software.

Client software—referred as Front End

Everything on a web page from logo, buttons, search bar and user interaction etc..

Server Software—referred as Back End – a part of application which a user cannot see. It focuses on logic of the site, creating the servers, working with databases, and API’s –Application Program Interface.

A Web API is an application programming interface for the Web.

A Browser API can extend the functionality of a web browser.

A Server API can extend the functionality of a web server.

**Skills required for Full-Stack Web Developer?**

**HTML**—Hyper Text Markup Language

**CSS**—Cascading Style Sheets – responsible for style of web site by color, layout and animations.

CSS frameworks, libraries and pre-processors such as **BOOTSTRAP**

**JavaScript**—Use JavaScript with HTML and CSS to create dynamic and interactive web pages and mobile applications

JavaScript libraries and frameworks such as **REACT** and **ANGULAR**

**Database:** A Database in web application is a place to store and organize the data.

Types of Databases to be learned are **SQL, MySQL**, **POSTGERSQL, MongoDB**

**Back End Languages**

Languages used to develop back end is JAVA, Python, Node and PHP.

There are few tech stacks used for both frontend and back end

MEAN STACK(MongoDB, Express, Angular and Node)

MERN STACK(MongoDB, Express, React and Node)

NODEJS

LAMP STACK(Linux, Apache, MySQL, and PHP)

**Testing and Debugging**

As an application is being developed there will be errors in the code and need fixing.

Debugging is the act to identify those errors(bugs) and fixing them.

**Version Control**

Version control is a way to track and manage changes to the code.

Git—is a popular software used to track the code.

If you mess up a lot of things in your code, you can use Git to go back to a previous version of your code instead of manually rewriting everything.

Learning Git also allows you to collaborate with others on a team and make changes to the same code base from different locations.

Unit-1: Introduction

Getting Started with HTML

* HTML is the primary language for developing web pages.
* It is not necessarily a programming language for building applications, but instead a language for describing documents.
* The fundamental approach to express structured documents in software engineering is MarkUp. Here text with tags, denoted by angle brackets that gives meaning.
* Hypertext defines the link between the web pages.
* Markup language is used to define the text document within tag which defines the structure of web pages.
* HTML5 is a new version of HTML with new functionalities with markup language with internet technologies.
* It has improved the markup available for documents and has introduced application programming interfaces(API) and Document Object Model(DOM).
* HTML does not have support for video and audio
* HTML5 supports both video and audio.

[**<DOCTYPE! html>**](https://www.geeksforgeeks.org/html-doctypes/)**:** This is the document type declaration (not technically a tag). It declares a document as being an HTML document. The doctype declaration is not case-sensitive.

[**<html>**](https://www.geeksforgeeks.org/html-html-tag/)**:** This is called the HTML root element. All other elements are contained within it.

[**<head>**](https://www.geeksforgeeks.org/html-head-tag/#:~:text=The%20tag%20in%20HTML,head%3E%20element%20can%20be%20omitted.)**:** The head tag contains the “behind the scenes” elements for a webpage. Elements within the head aren’t visible on the front-end of a webpage. HTML elements used inside the <head> element include:

* [<style>](https://www.geeksforgeeks.org/html-style-tag/)
* [<title>](https://www.geeksforgeeks.org/html-title-tag/)
* [<base>](https://www.geeksforgeeks.org/html-base-tag/)
* [<noscript>](https://www.geeksforgeeks.org/html-noscript-tag/)
* [<script>](https://www.geeksforgeeks.org/html-script-tag/)
* [<meta>](https://www.geeksforgeeks.org/html-meta-tag/#:~:text=The%20tag%20in%20HTML,keywords%2C%20document%20author%2C%20etc.)
* [<link>](https://www.geeksforgeeks.org/html-link-tag/)

[**<body>**](https://www.geeksforgeeks.org/html-body-tag/#:~:text=The%20tag%20in%20HTML,well%20as%20an%20ending%20tag.)**:** The body tag is used to enclose all the visible content of a webpage. In other words, the body content is what the browser will show on the front-end.

An HTML document can be created using any text editor. Save the text file using **.html** or **.htm**. Once saved as an HTML document, the file can be opened as a webpage in the browser.

**Video & Audio**

The HTML <video> element is used to show a video on a web page.

<video width=”100” height=”100” controls>

<source src=”movie.mp4” type=”video/mp4”>

<source src=”movie.ogg” type=”video/ogg”>

</video>

The controls attribute adds video controls, like play, pause and volume.

The source element allows to specify alternative video files which the browser may choose from. The browser first recognize the format of the file.

Autoplay attribute—to start a video automatically

Adding **muted** after autoplay –lets video start playing automatically but muted.

HTML Video file format

MP4 –video/mp4 Ogg—video/ogg

WebM—video/webm

The HTML <audio> element is used to add an audio on a web page.

Ex:

<audio controls>

<source src=”horse.mp3” type=”audio/mp3”>

</audio>

To start an audio file automatically, use the autoplay attribute.

Ex:

<audio controls autoplay>

<source src=”horse.mp3” type=”audio/mp3”>

</audio>

Types of audio file formats in HTML

MP3--audio/mp3

Ogg—audio/ogg

Wav—audio/wav

**CANVAS**

Canvas is a HTML Graphics. <canvas> element is used to draw graphics on a web page.

The <canvas> element is only a container for graphics. One must use Javascript to actually draw the graphics. Canvas has several methods for drawing paths, boxes, circles, text and adding images.

Canvas can draw colourful text, with or without animation.

It has a great features for graphical data presentation with an imagery of graphs and charts.

Canvas objects can move. Everything is possible from simple bouncing balls to complex animations.

Ex:

<canvas id=”myRect” width=”200” height=”100”></canvas>

Must have an id attribute—referred to by javascript

Width and height is necessary – to define the size of the canvas.

**Note: can have multiple <canvas> elements on one HTML page. By default, the<canvas> element has no border and no content.**

To add border, use the style attribute.

Ex:

<canvas id=”myCan” width=”100” height=”200” style=”border:2px solid #000000;”>

</canvas>

To draw in the rectangle area, must add a javascript to do drawing. Canvas can respond to javascript events. Canvas can respond to any user action like key clicks, mouse clicks, button clicks, finger movements.

Ex:

<script>

#find the <canvas> element using HTML DOM method getElementById();

Var a=document.getElementById(“myRect”);

#Need to draw object for the canvas using a built-in HTML Object getContext() with properties and methods for drawing…

Var atx=a.getContext(“2d”);

#moves the path to the specific point in the canvas, without creating a line.

atx.moveTo(0,0);

#lineTo() method adds a new point and create a line to that point from the last specified point in the canvas.

atx.lineTo(200,100)

#this stroke() method is used actually to draw the path on the canvas.

atx.stroke();

</script>

**Draw a Circle in the canvas**

Ex:

<script>

#find the <canvas> element using HTML DOM method getElementById();

Var c=document.getElementById(“myRect”);

#Need to draw object for the canvas using a built-in HTML Object getContext() with properties and methods for drawing…

Var ctx=a.getContext(“2d”);

#begin the path

Ctx.beginPath();

#arc(x,y,r,startangle, endangle)---it creates a curve/arc. Circle can be created with arc(), startangle is set to 0, endangle is 2\*Math.PI, r is the radius of the circle.

Ctx.arc(95, 50, 40, 0, 2\*Math.PI);

#this stroke() method is used actually to draw the path on the canvas.

Ctx.stroke();

</script>

**Draw a Text in the canvas**

the javascript code is followed as

Ex:

<script>

Var c=document.getElementById(“myCanvas”);

Var ctx=get.Context(“2d”);

#to define the font properties for the text.

Ctx.font=”40px Calibri”;

#fillText(text,x,y)—draw “filled” text on the canvas

Ctx.fillText(“Hello”,10,50);

</script>

**Adding Stroke Text in the Canvas**

<script>

        var t=document.getElementById("myText");

        var dtx=t.getContext("2d");

        dtx.font="30px Arial";

        dtx.strokeText("welcome",30,40);

</script>

**Drawing Linear Gradient**

Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.

There are two different types of gradients:

* createLinearGradient(*x,y,x1,y1*) - creates a linear gradient
* createRadialGradient(*x,y,r,x1,y1,r1*) - creates a radial/circular gradient

Once we have a gradient object, we must add two or more color stops.

The addColorStop() method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

To use the gradient, set the fillStyle or strokeStyle property to the gradient, then draw the shape (rectangle, text, or a line).

Ex:

<body>

    <p>Example of linear Gradient in Canvas</p>

    <canvas id="myGrad" width="500" height="300" style="border:1px solid#000000;"></canvas>

    <p>Example to create radial Gradient</p>

    <canvas id="myRaGrad" width="500" height="300" style="border:1px solid#000000;"></canvas>

    <script>

        var c=document.getElementById("myGrad");

        var ctx=c.getContext("2d");

        var gar=ctx.createLinearGradient(0,0,400,0);

        gar.addColorStop(0, "red");

        gar.addColorStop(1,"white");

        ctx.fillStyle = gar;

        ctx.fillRect(0,0,150,80);

        var r=document.getElementById("myRaGrad");

        var rctx=r.getContext("2d");

        var rgad=rctx.createRadialGradient(75, 50, 5, 90, 60, 100);

        rgad.addColorStop(0,"blue");

        rgad.addColorStop(1,"pink");

        rctx.fillStyle = rgad;

        rctx.fillRect(20,20,250,80);

    </script>

</body>

**Adding image in the canvas**

**</script>**

window.onload = function(){

        var i=document.getElementById("myImg");

        var ictx=i.getContext("2d");

        var img=document.getElementById("myImgSam");

        ictx.drawImg(img,10,10);

        };

</script>

**SVG Graphics**

SVG defines vector-based graphics in XML format.

* SVG stands for **Scalable Vector Graphics**
* SVG is used to define graphics for the Web

**<svg>** element is a container for SVG graphics. SVG has several method for drawing paths, boxes, circles, text, image

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

<!DOCTYPE HTML>  
<html>  
<head>  
<script>

##By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.

This is done by calling the event.preventDefault() method for the ondragover event:

function allowDrop(ev) {  
  ev.preventDefault();  
}  
  
function drag(ev) {

##The dataTransfer.setData() method sets the data type and the value of the dragged data  
  ev.dataTransfer.setData("image", ev.target.id);  
}  
##Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)

##Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method

##The dragged data is the id of the dragged element ("drag1")

##Append the dragged element into the drop element

function drop(ev) {  
  ev.preventDefault();  
  var data = ev.dataTransfer.getData("image");  
  ev.target.appendChild(document.getElementById(data));  
}  
</script>  
</head>  
<body>  
  
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">

</div>  
#to make image draggable set draggable to true

#the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.

<img id="drag1" src="imagesam.jpeg" draggable="true" ondragstart="drag(event)" width="336" height="69">  
  
</body>  
</html>

**GeoLocation**

GeoLocaion is used to locate a user’s position.

HTML geolocation is used to get the geographical position of the user.

Example:

<!DOCTYPE>

<html>

<head>

    Geo Location

</head>

<body>

    <p> Click the button to know your geolocation</p>

    <button onclick="getLocation()">Click Me</button>

<p id="sample"></p>

<script>

var x = document.getElementById("sample");

function getLocation() {

  if (navigator.geolocation) {

    navigator.geolocation.getCurrentPosition(showPosition);

  } else {

    x.innerHTML = "Geolocation is not supported by this browser.";

  }

}

function showPosition(position) {

  x.innerHTML = "Latitude: " + position.coords.latitude +

  "<br>Longitude: " + position.coords.longitude;

}

</script>

</body>

</html>

**Basic Styles using CSS – Cascading Style Sheets**

CSS is the language we use to style an HTML document. CSS describes how HTML elements should be displayed.CSS is used to format the layout of a webpage.

CSS can be added 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

An **inline CSS** is used to apply a unique style to a single HTML element. An inline CSS uses the style attribute of an HTML element.

Example:

<body style="background-color: chartreuse;">

<h1 style="color:blue;font-family: Cambria;"> First trail on CSS </h1>

<p style="color:red;font-size:20px;font-family: Gill Sans;">CSS stands for Cascading Style Sheets</p>

An **internal CSS** is used to define a style for a single HTML page. An internal CSS is defined in the <head> section of an HTML page, within <style> element. The following example sets the text color, font family, text-alignment of ALL the <h1> elements (on that page, and the text color and size of <p> elements to red.

<head><title>basic use of CSS</title>

    <style>

        body{

            background-color: darkcyan;

        }

        h1{

            font-size: large;

            font-family: 'Gill Sans';

            text-align: center;

        }

        p{

            color: blueviolet;

            font-size: medium;

        }

    </style>

<body>

<h1> First trail on CSS </h1>

<p>CSS stands for Cascading Style Sheets</p>

</body>

An **external style** sheet is used to define the style for many HTML pages. To use an external style sheet, add a link to it in the <head> section of each HTML page:

<head>  
  <link rel="stylesheet" href="styles.css">  
</head>

The **external style** sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a **.css** extension. the "styles.css" file looks like

body {

    background-color: aqua;

  }

  h1 {

    color: blue;

  }

  p {

    color: purple;

  }

**CSS Position**

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

There are 5 different position values:

* Static
* Relative
* Fixed
* Absolute
* sticky

Elements are then positioned using the top, bottom, left, and right properties.

## position: static

HTML elements are positioned static by default.

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

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

div.static {  
  position: static;  
  border: 3px solid #73AD21;  
}

## position: relative

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

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

div.relative {  
  position: relative;  
  left: 30px;  
  border: 3px solid #73AD21;  
}

## position: fixed

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

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

div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}

## position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

div.absolute {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}

## position: sticky

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

div.sticky {  
  position: -webkit-sticky; /\* Safari \*/  
  position: sticky;  
  top: 0;  
  background-color: green;  
  border: 2px solid #4CAF50;  
}

**CSS Background Images:**

CSS allows you to add multiple background images for an element, through the background-image property. The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.

background-image: url(imgagesam.jpg), url(img\_w3slogo.gif);

the individual background properties

background: url(img\_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;

The requirements are as follows:

* Fill the entire page with the image (no white space)
* Scale image as needed
* Center image on page
* Do not cause scrollbars

background-size: cover;

**Bootstrap – Setup**

A front-end framework that helps you build mobile responsive websites more quickly and easily. The framework is also completely free, versatile, and intuitive. With Bootstrap, you can conjure complex web pages from standard HTML and customize them to your needs. It also comes with additional functionality such as carousels, buttons, popups, and more.

Saves your time and energy.

What is a Bootstrap?

* Bootstrap is a free front-end framework for faster and easier web development
* Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
* Bootstrap also gives you the ability to easily create responsive designs

What is a Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

Advantages:

* Easy to use
* Responsive features
* Mobile-first approach
* Browser compatibility

Set Up:

First—integrate into your development environment aka webpage.

There are two different possibilities

Include bootstrap from CDN or

Download and use bootstrap locally.

1. Load Bootstrap via CDN(Content Delivery Network)—Bootstrap consists mainly of style sheets and scripts. They can be loaded in the header and footer of your web page like other assests such as custom fonts. The framework offers a CDN access path for that.

To get Bootstrap into webpage, simple inset the code into <head> section.

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

Script

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

Example

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

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

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<title>Hello, world!</title>

</head>

<body>

<h1>Hello, world!</h1>

<!-- Optional JavaScript; choose one of the two! -->

<!-- Option 1: Bootstrap Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

<!-- Option 2: Separate Popper and Bootstrap JS -->

<!--

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF" crossorigin="anonymous"></script>

-->

</body>

</html>