**HTML5 controls:**

SVG: Scalable Vector Graphics.

SVG is used to define graphics for web.

Canvas elements is used to draw the graphics on fly via JS.

Canvas is just container for graphics but we must use JS to draw graphics.

Video element specifies a standard way to embed a video in webpage.

Semantic element gives its exact meaning to both browser and developer.

Examples of semantic elements:

<Header> <footer> <aside> <section> <article> <nav> <fig> <figcaption> <main> <details> <mark>

**<!DOCTYPE HTML>** version of html page is written

**HTML FORMS –**

different types of input elements like text fields, checkboxes, radio buttons, submit buttons etc.

Example:

<form action="action\_page.php">

<input type="text" value="Mickey">

<input type="submit" value="Submit">

“radio”

</form>

Inline elements don’t start in a new line. These elements take required width.

**HTML NESTED LIST**

Lists inside lists

<ul>

<li>Coffee</li>

<li>Tea>

<ul>

<li>Black tea</li>

<li>Green tea</li>

</ul>

</li>

<li>Milk</li>

</ul>

**HTML <table> --** <tr> row, <td> data/cell, <th>header

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

border-collapse: collapse;

padding: 5px;

text-align: left;

}

</style>

</head>

<body>

<table style="width:100%">

<tr>

<th>First Name</th>

<th>Last Name</th>

<th>Points</th>

</tr>

</table>

</body>

</html>

<Article> tag contains independent self-contained content.

<section> tag defines sections in a document such as chapters, headers,footers,or any other sections of a document.

**HTML STORAGE API** –

With local storage, web applications can store data locally within the user's browser.

The sessionStorage object stores the data for only one session. The data is deleted when the user closes the specific browser tab.

With local storage, web applications store data locally within users’ browser. Before HTML5 data had to be store in cookies local storage is very secure and larger amount of data can be store locally without effecting the website performance.

Local storage provided two objects for storing data on client.

Window.localstorage: Stores data with no expiration date.

Window.sessionstorage: stores data for only one session.

**<!DOCTYPE HTML>** version of html page is written

**CODE FOR BLOCK ELEMENTS:**

<!DOCTYPE html>

<html>

<body>

<h1>My <span style="color:red">Important</span> Heading</h1>

<div style="background-color:black;color:white;padding:20px;">

<h2>India</h2>

<p>I love India. Its is my home land</p>

<p>India is the cradle of human race, the birth place of human speech, the mother of history, grand mother of tradition </p>

</div>

</body>

</html>

**HTML <iframe>** - a webpage within a web page

<iframe src="http://www.w3schools.com"></iframe>

**HTML <table> --** <tr> row, <td> data/cell, <th>header

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

border-collapse: collapse;

padding: 5px;

text-align: left;

}

</style>

</head>

<body>

<table style="width:100%">

<tr>

<th>First Name</th>

<th>Last Name</th>

<th>Points</th>

</tr>

</table>

</body>

</html>

**Block elements:** Block elements always start in a new line and take up the full width available.

Example: <div>

<h1>……<h6>

<form>

**JS variables:**

For example, price1, price2, and total are variables.

var price1 = 5;  
var price2 = 6;  
var total = price1 + price2;

**JS DATATYPES:**

JavaScript variables can hold many datatypes. They are numbers, strings, objects and more:

var length = 16; This is number.  
var lastName =”Shruthi”; This is string.  
var x = {firstName:”Shruthi” lastName:”Mekarthi”}; This is object

**JS FUNCTIONS:**

function myFunction (p1, p2) {

return p1 \* p2;                
}

**JS properties**

JS properties are values associated with JS object. JS object is collection of unordered properties.

JS properties are values associated with objects.

Person.name = Shruthi;

Here ‘name’ is the property.

**Adding methods/property to existing class using prototype:**

Person. Nationality = “English”;

function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
}  
Person.prototype.nationality = "English";

**Creating an object:**

methodName : function() {

}

Accessing: objectName.methodName()

**Calling request xmlhttp request:**

var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
       // Typical action to be performed when the document is ready:  
       document.getElementById("demo").innerHTML = xmlhttp.responseText;  
    }  
};  
xhttp.open("GET", "filename", true);  
xhttp.send();

**JS ARRAYS:**

JS arrays are used to store multiple values in a single variable.

Var persons = [“Samatha”, “Shruthi”, “Shirisha”];

**JS LOOP:**

While(i<10) {  
    text += “The number is “ + i;   
    i++;  
}

**JS objects:**

Objects are variables. But objects can contain many values.

Var person = { firstname: ”Shruthi”,Lastname:”Mekarthi”,age:25};

**JS SETINTERVEL:**

Setintervel is used to repeat particular task at the given time interval.

**Var** myVar = setInterval(myTimer, 1000);  
  
function myTimer() {  
    var d = new Date();  
    document.getElementById("demo").innerHTML = d.toLocaleTimeString();  
}

**JS FUNCTION:**

Function in JavaScript that does not has arguments.

<script>

function msg(){

alert("hello! this is message");

}

</script>

<input type="button" onclick="msg()" value="call function"/>

Function that has one argument.

<script>

function getcube(number){

alert(number\*number\*number);

}

</script>

<form>

<input type="button" value="click" onclick="getcube(4)"/>

</form>

**JS methods** are actions that can be performed on objects.

**Prototype:**

Every JS object has prototype. All JS objects inherit properties and methods from prototype.

**Inheritance:** If one object acquires properties and methods of parent object is known as inheritance.

**Self-invoking function**: A function that is called by itself.

**XML HTTP request** object is used to request data from server. It is developers dream. We can request the data from a server, update a page without reloading the page.

**JS Hoisting**

It is JavaScript’s default behavior of moving all declarations to the top of current scope.

In JavaScript, a variable can be declared after it has been used.

**Example:**

x =5;  
  
elem = document.getElementById("demo"); //find an element  
elem.innerHTML = x;  //display x in the element        
  
var x; //delare x

**CSS:**

**CSS box model:**

The box model allows us to add a border around elements, and to define space between elements.

div {  
    width: 300px;  
    border: 25px solid green;  
    padding: 25px;  
    margin: 25px;  
}

**CSS Pseudo class:**

A pseudo-class is used to define a special state of an element.

selector: pseudo-class {  
    property: value;  
}

In CSS, selectors are patterns used to select the elements that you want to style.

CSS Z index specifies stack order of an element. The element with greater stack order is always in front with lower stack order.

**> child selector-function**

The :nth-child(*n*) selector matches every element that is the *n*th child, regardless of type, of its parent.

*n* can be a number, a keyword, or a formula.

**Px and % how they work** units expressing length

**combinations of selectors**

1. descendant selector (space)

div p {  
    background-color: yellow;  
}

selects all <p> elements inside <div> elements

1. child selector (>)

div > p {  
    background-color: yellow;  
}

selects all <p> elements that are immediate children of a <div> element

1. adjacent sibling selector (+)

div + p {  
    background-color: yellow;  
}

selects all <p> elements that are placed immediately after <div> elements

1. general sibling selector (~)

div ~ p {  
    background-color: yellow;  
}

selects all <p> elements that are siblings of <div> elements

**css image sprite** collection of images put into single image

("img\_navsprites.gif")

#navlist {  
    position: relative;  
}

**Css reset** styles will be reset to null

Counter-reset: section;

**How to use selectors that we declare in external style sheet**

#id

.class  
**Css3 transition** – changes one value to another

-webkit-transition: width 2s;

**Css3 gradient** – transitions between two or more specified colors

Linear gradients (up down left n right)

Radial gradients (center)

**CSS3 shadows –**

text shadow

h1 {  
    text-shadow: 2px 2px;  
} horizontal, vertical

box shadow

div {  
    box-shadow: 10px 10px grey;  
}

**display-** visible, hide

h1.hidden {  
    visibility: hidden;  
}  
**Color some examples**

<div style="background-color:red"> </div>  
**overflow property** - to clip content or to add scrollbars when the content of an element is too big to fit in a specified area

overflow: visible; not clipped and it renders outside the element's box

hidden; the overflow is clipped, and the rest of the content is hidden

scroll; the overflow is clipped and a scrollbar is added to scroll inside the box

auto; similar to scroll

* x; (specifies with left and right edges) -y; (horizontal and vertical)

**padding –** generate space around element

padding-top

- left

-right

- bottom

p {  
    padding: 50px 30px 50px 80px;  
}  
**offset-** The outline-offset property adds space between an outline and the edge or border of an element.

outline-offset: *length*|initial|inherit;  
**media queries-** type and capabilityof device

@media screen and (min-width: 480px) {  
    body {  
        background-color: lightgreen;  
    }  
}  
**orientation –** landscape when viewport height is larger than the width, portrait  
**view port**  - the browser window size, em and rem   
**link- index** -- <head>  
  <link rel="stylesheet" type="text/css" href="theme.css">  
</head>  
**margin- auto –** horizontally center the element within its container

div {  
    width: 300px;  
    margin: auto;  
    border: 1px solid red;  
}

**css order of precedence**

inline css ( html style attribute ) overrides css rules in style tag and css file.

a more specific selector takes precedence over a less specific one.

rules that appear later in the code override earlier rules if both have the same specificity.

**CSS margin:**

margin: *length*|auto|initial|inherit;

p {  
    margin: 2cm 4cm 3cm 4cm;  
 }

negative values are allowed

**ANGULER JS:**

The MVC pattern is made up of the following three parts:

1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
2. **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
3. **Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

**ANGULER JS DIRECTIVES:**

AngularJS lets you extend HTML with new attributes called **Directives**.

AngularJS has a set of built-in directives which offers functionality to your applications.

AngularJS directives are extended HTML attributes with the prefix ng.

The ng-app directive initializes an AngularJS application.

The ng-init directive initializes application data.

<div ng-app="" ng-init="firstName='John'">  
  
<p>Name: <input type="text" ng-model="firstName"></p>  
<p>You wrote: {{ firstName }}</p>  
  
</div>

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

<div ng-app="" ng-init="quantity=1;price=5">  
  
Quantity: <input type="number" ng-model="quantity">  
Costs:    <input type="number" ng-model="price">  
  
Total in dollar: {{ quantity \* price }}  
  
</div>

**ng-repeat:**

<div ng-app="" ng-init="names=['Jani','Hege','Kai']">  
  <ul>  
    <li ng-repeat="x in names">  
      {{ x }}  
    </li>  
  </ul>  
</div>

<div ng-app="" ng-init="names=[  
{name:'Jani',country:'Norway'},  
{name:'Hege',country:'Sweden'},  
{name:'Kai',country:'Denmark'}]">  
  
<ul>  
  <li ng-repeat="x in names">  
    {{ x.name + ', ' + x.country }}  
  </li>  
</ul>  
  
</div>

**ANGULER JS Http:**

<div ng-app="myApp" ng-controller="myCtrl">   
  
<p>Today's welcome message is:</p>  
<h1>{{myWelcome}}</h1>  
  
</div>  
  
<script>

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("welcome.htm")  
    .then(function(response) {  
        $scope.myWelcome = response.data;  
    });  
});

</script>

**ANGULER JS FORM VALIDATION:**

AngularJS offers client-side form validation.

AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

AngularJS also holds information about whether they have been touched, or modified, or not.

<form name="myForm">  
<input name="myInput" ng-model="myInput" required>  
</form>  
  
<p>The input's valid state is:</p>  
<h1>{{myForm.myInput.$valid}}</h1>

**DIFFERENT EVENTS IN ANGULER.JS:**

We can add AngularJS event listeners to your HTML elements by using one or more of these directives:

ng-click, ng-dblclick, ng-blur, ng-change, ng-copy, ng-cut, ng-keypress, ng-keydown, ng-paste, ng-mouseenter, ng-mouseleave, ng-focus.

**ng-mousemove example:**

<div ng-app="myApp" ng-controller="myCtrl">  
  
<h1 ng-mousemove="count = count + 1">Mouse over me!</h1>  
  
<h2>{{ count }}</h2>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.count = 0;  
});  
</script>

**ng-click example:**

<div ng-app="myApp" ng-controller="myCtrl">  
  
<button ng-click="count = count + 1">Click me!</button>  
  
<p>{{ count }}</p>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.count = 0;  
});  
</script>

**ng-toggle:**

<div ng-app="myApp" ng-controller="myCtrl">  
  
<button ng-click="myFunc()">Click Me!</button>  
  
<div ng-show="showMe">  
    <h1>Menu:</h1>  
    <div>Pizza</div>  
    <div>Pasta</div>  
    <div>Pesce</div>  
</div>  
  
</div>  
<script>  
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.showMe = false;  
    $scope.myFunc = function() {  
        $scope.showMe = !$scope.showMe;  
    }  
});  
</script>

**ANGULER JS SERVICE:**

In AngularJS, a service is a function, or object, that is available for, and limited to your AngularJS application.

**$location service:**

var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $location) {  
    $scope.myUrl = $location.absUrl();  
});

**$http service:**

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $http) {  
    $http.get("welcome.htm").then(function (response) {  
        $scope.myWelcome = response.data;  
    });  
});

**$timeout service:**

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $timeout) {  
    $scope.myHeader = "Hello World!";  
    $timeout(function () {  
        $scope.myHeader = "How are you today?";  
    }, 2000);  
});

**$interval service:**

var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope, $interval) {  
    $scope.theTime = new Date().toLocaleTimeString();  
    $interval(function () {  
        $scope.theTime = new Date().toLocaleTimeString();  
    }, 1000);  
});

**angulerjs vs backbonejs:**

Anguler.js supports the functionalities required to develop the HTML5 applications whereas backbone.js does it independently.

**Anguler.js scopes:**

Scope is binding part between view and controller.

**Example:**

<div ng-app="myApp" ng-controller="myCtrl">  
  
<h1>{{carname}}</h1>  
  
</div>  
  
<script>

var app = angular.module('myApp', []);  
  
app.controller('myCtrl', function($scope) {  
    $scope.carname = "Volvo";  
});

</script>

**Anguler.js transclude:**

This compiles the content of an element and make it available to directives.