# jQuery

## Basics

It is a JavaScript library specialized for changing web page documents on the fly.

jQuery uses selector engine.

|  |  |
| --- | --- |
| **JavaScript** | **jQuery** |
| document.getElementByTagName(“p”)  [0].innerHTML = “Change”; | $(“p”).html(“Change”) |
| function init(){} window.onload = init; | $(document).ready(function(){}); |

The JavaScript interpreter doesn’t change the original HTML and CSS files. It makes changes to the DOM’s representation of the page in the browser’s memory.

The dollar sign “$” with the parentheses is the shorter name of the jQuery function. This shortcut saves us from writing jQuery(). The jQuery function is also often referred to as the **jQuery wrapper**.

jQuery() 🡪 $()

$() can have css selector, JavaScript Object, HTML

If we put a **CSS selector** here, jQuery will return us the **set of elements** that match that selector

If we put a string of **HTML** in here, we can add DOM elements to the browser page.

**JavaScript Object** can also be used as a selector.

### CSS Selector

CSS selectors select elements to add style to those elements; jQuery selectors select elements to add behavior to those elements.

### Element Selector

h1{

text-align: left;

}

### Class Selector

.my\_class{

position: absolute;

}

### ID Selector

#my\_id{

color: #3300FF

}

### jQuery methods

.hide() 🡪 Hides the element

.show() 🡪 Shows the element

.toggle() 🡪 Hide and Show the element

.slideUp() 🡪 It changes the height property of the element until its 0

.slideDown() 🡪 It changes the height property of the element from 0

.slideToggle() 🡪 Slides up and down the element

.fadeIn() 🡪 Fades in the element

.fadeOut() 🡪 Fades out the element

.fadeTo() 🡪 Fades to the element

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> jQuery </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "basics.js"></**script**>

</**head**>

<**body**>

<**p** id = "text"> Hide and Show </**p**>

<**button** id = "hide">Hide</**button**>

<**button** id = "show">Show</**button**>

<**button** id = "toggle">Toggle</**button**><**br**/><**br**/>

<**button** id = "slide-up">Slide Up</**button**>

<**button** id = "slide-down">Slide Down</**button**>

<**button** id = "slide-toggle">Slide Toggle</**button**><**br**/><**br**/>

<**button** id = "fade-in">Fade In</**button**>

<**button** id = "fade-out">Fade Out</**button**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#hide").click(**function**(){

$("#text").hide(2000);

});

$("#show").click(**function**(){

$("#text").show(2000);

});

$("#toggle").click(**function**(){

$("#text").toggle(2000);

});

$("#slide-up").click(**function**(){

$("#text").slideUp(2000);

});

$("#slide-down").click(**function**(){

$("#text").slideDown(2000);

});

$("#slide-toggle").click(**function**(){

$("#text").slideToggle(2000);

});

$("#fade-in").click(**function**(){

$("#text").fadeIn(2000);

});

$("#fade-out").click(**function**(){

$("#text").fadeOut(2000);

});

});

**Code 1: basics.js**

**$(this)**, this selector gives us an easy way to point to the current element. It’s important to think $(this) as **context-dependent.**

### Descendant Selectors

html

↑

body

↑

div

↑

div div p id = “good”

↑ ↑

img p

$(“div p#good”)

$(“div div”)

$(“div p”)

$(“div div img”)

## Binding an Event

* **$(“#myElement”).click(function(){})**
* **$(“#myElement”).bind(“click”,function(){})**

The first method is simply a shortcut for second method, but only **when the DOM elements exist already**

### Events

**Selector + Event + Function = Complex Interaction**

|  |  |
| --- | --- |
| **Events** | **Description** |
| click | Mouse Events 🡪 Click |
| dblclick | Mouse Events 🡪 Double Click |
| focusin | Mouse Events 🡪 Focus In |
| focusout | Mouse Events 🡪 Focus Out |
| hover | Mouse Events 🡪 Hover |
| mousedown | Mouse Events 🡪 Mouse Down |
| mouseenter | Mouse Events 🡪 Mouse Enter |
| mouseleave | Mouse Events 🡪 Mouse Leave |
| mousemove | Mouse Events 🡪 Mouse Move |
| mouseout | Mouse Events 🡪 Mouse Out |
| mouseover | Mouse Events 🡪 Mouse Over |
| mouseup | Mouse Events 🡪 Mouse Up |
| toggle | Mouse Events 🡪 Toggle |
| keydown | Keyboard Events 🡪 Press Down Arrow |
| keypress | Keyboard Events 🡪 Press Enter |
| keypup | Keyboard Events 🡪 Press Up Arrow |
| blur | Form Events 🡪 Focus out of the field box |
| change | Form Events 🡪 Change in the field Text |
| focus | Form Events 🡪 Focusing in the field box |
| select | Form Events 🡪 Selecting the field box |
| submit | Form Events 🡪 Submit Clicked |
| load | Document Events 🡪 Page Fully Loaded |
| ready | Document Events 🡪 Page is ready |
| unload | Document Events 🡪 Page unloaded |
| error | Browser Events 🡪 Error in Browser |
| resize | Browser Events 🡪 Resize the Browser |
| scroll | Browser Events 🡪 Scroll through the Browser |

### Removing Events

$(“#myElement”).unbind(“click”) 🡪 Removes the click event from myElement

$(“#myElement”).unbind() 🡪 Removes all the events from myElement

### Going through Elements

$(“.nav-items”).each(function(){});

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Selectors</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "selectors.js"></**script**>

</**head**>

<**body**>

<**div** id = "image-container">

<**img** src = "images/desktop.ico" class = "picture"/>

<**img** src = "images/hdd.ico" class = "picture"/>

<**img** src = "images/trash.ico" class = "picture"/>

<**img** src = "images/folder.ico" class = "picture"/>

</**div**>

</**body**>

</**html**>

$(document).ready(**function**(){

$(".picture").click(**function**(){

$(**this**).slideUp();

**var** discount = Math.floor((Math.random()\*5 + 5));

**var** message = "<p> Your discount price is: " + discount + "%</p>";

$("#image-container").append(message);

$(".picture").each(**function**(){

$(**this**).unbind("click");

});

$(".picture").hide(5000);

});

});

**Code 2: selectors.js**

### Checking Element

**$.contains(document.getElementById(“parent”), document.getElementById(“child”))**

This method check if the first parameter (parent) contains the second parameter(child)

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Check</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "check.js"></**script**>

</**head**>

<**body**>

<**div** id = "container">

<**button** id = "check"> Check </**button**>

</**div**>

<**button** id = "outsider"> Outsider</**button**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#check").click(**function**(){

console.log($.contains(document.getElementById("container"), **this**));

**var** outsider = document.getElementById("outsider");

console.log($.contains(document.getElementById("container"), outsider));

});

});

**Code 3: check.js**

### CSS Hover

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Check</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "hover.js"></**script**>

</**head**>

<**body**>

<**div** id = "container">

<**button** id = "hover"> Hover </**button**>

</**div**>

</**body**>

<**style**>

#hover{

**border**: **none**;

**padding**: 5px;

transition: 0.5s;

}

**.hover-on**{

**color**: rgba(140, 234, 12, 0.5);

**background-color**: rgba(0, 0, 0, 0.6);

transition: 0.5s;

}

</**style**>

</**html**>

$(document).ready(**function**(){

$("#hover").bind("mouseover",**function**(){

**if**($(**this**).hasClass("hover-on")){}

**else**{$(**this**).addClass("hover-on");}

});

$("#hover").bind("mouseleave", **function**(){

**if**($(**this**).hasClass("hover-on")){

$(**this**).removeClass("hover-on");

}

});

});

**Code 4: hover.js**

## Modify the Document

**$(“img#thumbnail”).remove()**

This removes the method drops the element out of the DOM

**$(“img#thumbnail”).detach()**

This detach method takes the selected elements(s) but holds on to it so that it can be reattached later.

**$(“.fish”).parent()**

It selects all the elements in the fish class and then get the element above those element

**$(“.menu\_list”).children()**

It selects all the elements in the menu\_list get all the elements below it

**$(“.fish”).prev()**

Selects all the element of fish class and get the immediate left sibling

**$(“.fish”).next()**

Selects all the element of fish class and get the immediate right sibling

**$(“h3”).replaceWith(“<h1>My Menu</h1>”);**

Selects all the h3 elements, replace with selected elements with the contents in parenthesis.

It works well when we have a one-to-one replacement.

**$(“.meat”).before(“<li>Tofu</li>”);**

before inserts content before the selected element

**$(“.meat”).after(“<li>Tofu</li>”);**

after inserts content after the selected element

**$(“.menu\_list”).children().first();**

The first method will filter out everything but the first element in a selected set of elements

**$(“.menu\_list”).children().last();**

The last method will filter out everything bu the last element in a selected set of elements.

**$(“.menu\_list”).children().eq(0);**

The **eq** method will filter out everything out the element whose index number is equal to what we put in the parenthesis in a selected set of elements.

**$(“.menu\_list”).children().slice(1,3);**

The slice method will filter out everything but elements with an index between the index number we put it parenthesis.

**$(“.menu\_list”).children().filter(“.organic”);**

The filter method will filter out everything but elements that match the selector we put in its parenthesesis

**$(“.menu\_list”).children().not(“.local”);**

The not method will filter out everything that does not match the selector we place in the parentheses.

### Storing the Variable

**$f = $(“.fish”).parent().parent().detach();**

jQuery variable

**$f[2] = 15;**

jQuery array, **“$”** it is just a coding convention

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Set Menu</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "food.js"></**script**>

</**head**>

<**body**>

<**button** id = "vegetarian">Vegetarian</**button**>

<**button** id = "restore">Restore</**button**>

<**div** id = "menu-warapper">

<**h4**>Dinner Entrees</**h4**>

<**ul** class = "menu-entrees">

<**li**>Thai-Style Halibut

<**ul** class = "menu-list">

<**li**>coconut milk</**li**>

<**li**>pan-fried halibut</**li**>

<**li**>early autumn vegetables</**li**>

<**li**>thai spices</**li**>

</**ul**>

</**li**>

<**li**>House Grilled Panini

<**ul** class = "menu-list">

<**li** class = "meat">prosciutto</**li**>

<**li**>provolone</**li**>

<**li**>avocado</**li**>

<**li**>sourdough</**li**>

</**ul**>

</**li**>

<**li**>Southwest Slider

<**ul** class = "menu-list">

<**li**>whole chiles</**li**>

<**li** class = "meat">hamburger</**li**>

<**li**>pepperjack cheese</**li**>

<**li**>multigrain roll</**li**>

</**ul**>

</**li**>

</**ul**>

</**div**>

</**body**>

</**html**>

**var** $deleted = [];

$(document).ready(**function**(){

$("#vegetarian").click(**function**(){

$deleted.push($(".menu-list > .meat")[0]);

$deleted.push($(".menu-list > .meat")[1]);

$(".menu-list > .meat").each(**function**(){

$(**this**).detach();

})

});

$("#restore").click(**function**(){

$(".menu-list").children().eq(4).before($deleted[0]);

$(".menu-list").children().eq(9).before($deleted[1]);

});

});

**Code 5: food.js**

## Effects and Animation

**Lightning Animation**

**$("#lightning").fadeOut(Math.floor(Math.random()\*2000))**

**.fadeIn(Math.floor(Math.random()\*2000));**

This animation will cause lightning effect

**Animation**

The **.animate()** will only work on CSS properties that use numbers for their settings.

|  |  |
| --- | --- |
| borders, margin, padding | bottom, left, right, and top position |
| element height, min-height, and maz-height | background position |
| element width, min-width, and max-width | letter spacing, word spacing |
| font size | text indent, line height |

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Animation</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "animation.js"></**script**>

</**head**>

<**body**>

<**img** src = "images/lightning.png" alt = "light1" id = "light1"/>

<**img** src = "images/lightning2.png" alt = "light2" id = "light2"/><**br**/><**br**/>

<**img** src = "images/desktop.ico" id = "desktop" class = "icon"/>

<**img** src = "images/folder.ico" id = "folder" class = "icon"/>

<**img** src = "images/hdd.ico" id = "hdd" class = "icon"/>

<**img** src = "images/trash.ico" id = "trash" class = "icon"/>

</**body**>

<**style**>

#light1, #light2{

**width**: 5%;

**height**: 5%;

}

</**style**>

</**html**>

$(document).ready(**function**(){

$(window).focus(blink);

$(".icon").click(**function**(){

$(**this**).animate({

opacity: 0,

width: "200",

height: "200",

},5000);

});

});

**function** blink(){

$("#light1").fadeOut(Math.floor(Math.random()\*2000))

.fadeIn(Math.floor(Math.random()\*2000));

$("#light2").fadeOut(Math.floor(Math.random()\*2000))

.fadeIn(Math.floor(Math.random()\*2000));

setInterval(blink, 5000);

}

**Code 5: animation.js**

### Slide Left or Right

We need to make sure the animated element needs to have a position property to relative.

**.icon{position: relative};**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Animation</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "animation.js"></**script**>

</**head**>

<**body**>

<**img** src = "images/desktop.ico" id = "desktop" class = "icon"/>

<**img** src = "images/folder.ico" id = "folder" class = "icon"/>

<**img** src = "images/hdd.ico" id = "hdd" class = "icon"/>

<**img** src = "images/trash.ico" id = "trash" class = "icon"/>

</**body**>

<**style**>

**.icon**{

**position**: **relative**;

}

</**style**>

</**html**>

$(document).ready(**function**(){

$(".icon").click(**function**(){

$(**this**).animate({

right: "+=100px"

}, 5000);

});

});

**Code 6: slideleft.js**

**jQuery is great for the elements that already exists in the document**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Attribute</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "attr.js"></**script**>

</**head**>

<**body**>

<**button** id = "showImage">Show Image</**button**>

<**div** id = "image-container"></**div**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#showImage").click(**function**(){

**if**(!document.getElementById("desktop")){

**var** image = document.createElement("img");

$(image).attr("src", "images/desktop.ico");

$(image).attr("id", "desktop");

$("#image-container").append(image);

$("#showImage").html("Boom I am back!");

}**else**{

**var** image = document.getElementById("desktop");

document.getElementById("image-container").removeChild(image);

$("#showImage").html("Click me again!");

}

});

});

**Code 7: attr.js**

## Window Events

|  |  |
| --- | --- |
| **window.name** | property, let us access or set the name of the window |
| **window.history** | property, let us access the different URLs that the window has loaded over time |
| **window.document** | property, main content of the loaded document |
| **window.onfocus** | property, detects when the window receives a click, keyboard input or any kind of input |
| **window.setTimeout()** | method, set a period of time to wait before calling a function or other statement |
| **window.clearTimeout()** | method, cancel the period of time of wait |
| **window.setInterval()** | method, set a period of time to wait between repetitions of a function call or other statement |
| **window.clearInterval()** | method, cancel the period of time to wait between repetitions |
| **window.onblur()** | detects when the window loses focus |

## Jquery Ajax

Asynchronous JavaScript and eXtensible Markup Language

<?php

$employees = ["sam" => "engineer", "bob" => "housekeeper"];

**if**(isset($\_REQUEST['postit'])):

**echo** json\_encode($employees);

**elseif**(isset($\_REQUEST['getit'])):

**echo** json\_encode($employees);

**else**:

**endif**;

?>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Ajax</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "ajax.js"></**script**>

</**head**>

<**body**>

<**div** id = "data"></**div**>

</**body**>

</**html**>

$(document).ready(**function**(){

$.ajax({

url: "ajax.php?getit=confirm",

cache: **false**,

dataType: "json",

success: **function**(json\_data){

**var** employees = json\_data;

console.log(employees.sam);

console.log(employees.bob);

}

});

**var** url = "ajax.php?getit=confirm";

$.get(url, **function**(data, status){

console.log(data, status);

});

**var** url = "ajax.php";

**var** data = "postit=confirm"

$.post(url, data, **function**(data, status){

console.log(data, status);

});

**var** url = "ajax.php?getit=confirm";

$.getJSON(url, **function**(employees, status){

console.log(employees.sam);

})

});

**Code 7: ajax.js**

**Javascript Version**

|  |  |
| --- | --- |
| **Post** | **Get** |
| request = new XMLHttpRequest(); | request = new XMLHttpRequest(); |
| var url = **"data.php"**; | var url = "**data.php?call=calling**"; |
| var requestData = **"call=calling"**; | var requestData = **null**; |
| request.open(**"POST"**, url, true); | request.open(**"GET"**, url, true); |
| request.onreadystatechange = func | request.onreadystatechange = func |
| **request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");**  Only Required for Post | |
| request.send(**requestData**); | request.send(**null**)**;** |

**jQuery Version**

|  |  |
| --- | --- |
| **Post** | **Get** |
| $.post(*URL,data,callback*); | $.get(*URL,callback*); |
| $.post(url, {name: “”, age: “”}  function(data, status){}  ); | $.get(url,  function(data,status){}  ); |
| **Works for Both**  $.ajax({  url: “”,  cache: false,  dataType: “json”,  success: function(data){}  }); | |
| **Other Shortcuts**  **$.getJSON()**  **$.getScript()**  **$.load()** | |

## Serialize

<form id = “myform>

<input type = “text” name = “a” value = “1”/>

<input type = “text” name = “b” value = “2”/>

<input type = “submit” name = “submit”/>

**$(“#myform”).serialize()**

The form ID selector serializes the method

a=1&b=2

🡪The value of submit will not show up

**$(“#myform:input”).serializeArray()**

The form id takes only data from the input and then transform into an array

{

{

name: “a”,

value: “1”

}

{

name: “b”,

value: “2”

}

}

🡪The value of submit will not show up

**Advantages of Serialization**

|  |
| --- |
| **Normal Way (Form Data)** |
| //**JavaScript Way**  var a = document.querySelector(“input[name=’a’]);  var value = a.value();  var data = “a=” + value  //**jQuery Way**  var b = $(“input[name=’b’]”).val();  var data = “b=” + b  //**Serialize Way** (Get all the values)  var form\_id\_data = $(“#myform”).serializa() |

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Serial </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/jquery.js"></**script**>

<**script** src = "serial.js"></**script**>

</**head**>

<**body**>

<**form** id = "myForm">

<**label** for = "fullname"> Full Name </**label**><**br**/>

<**input** type = "text" name = "fullname"><**br**/><**br**/>

<**label** for = "age"> Age </**label**><**br**/>

<**input** type = "text" name = "age"><**br**/><**br**/>

<**input** type = "submit" name = "submit" value = "Confirm">

</**form**>

</**body**>

</**html**>

<?php

**if**(isset($\_POST)):

$fullname = $\_POST['fullname'];

$age = $\_POST['age'];

$fullname = "FullName: ".$fullname;

$age = "Age: ".$age;

**try**{

$file = fopen("serial.txt", "w");

$string = $fullname.", ".$age;

fwrite($file, $string);

var\_dump($\_POST);

}**catch**(Exception $e){

**echo** ("Error: ".$e->getMessage());

}**finally**{

fclose($file);

}

**endif**;

?>

$(document).ready(**function**(){

$("#myForm").submit(**function**(e){

e.preventDefault();

*// var data = $("#myForm:input").serializeArray();*

**var** data = $("#myForm").serialize();

**var** url = "serial.php";

$.ajax({

url: url,

type: 'post',

data: data,

success: **function**(data){

alert(data);

}

});

});

});

**Code 8: serialize.js**

## jQueryUI

jQuery offers a plug-in architecture that allows web developers to extend (or add onto) the core jQuery library.

**Effects plug-ins**

It extends jQuery by adding more effects. Make our elements bounce, explode, pulsate, or shake. jQuery UIs also includes easing functions, complex mathematical operations that make animations look more realistic.

**Interaction plug-ins**

Interactions add more complex behavior to web apps. We can enable user to interact with elements by making those elements draggable, droppable or sortable.

**Widget plug-ins**

A web widget is a self-contained component that adds functionality to our web app. Widgets save our tons of coding time and complexity while creating useable and responsive user interface elements.

**Plugins**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Puff | Effect | Makes an element appear to expand and dissipate into transparency like smoke |
| Autocomplete | Widget | Provides a list of possible values when a user types into an input field |
| Droppable | Interaction | Makes a DOM element a target for draggable elements |
| Explode | Effect | Makes an element appear to break into pieces and spread out in several directions. |
| Sortable | Interaction | Makes an element sortable by dragging |
| Progressbar | Widget | Displays the current percentage of completion for some event |
| Resizable | Interaction | Gives an element sortable by dragging |
| Blind | Effect | Makes an element appear to slide up or down like a window treatment |
| Accordion | Widget | Creates stacked and collapsible areas to organize web content |

### Date Picker

$(“#datepicker”)**.datepicker()**

.datepicker({

stepMonths: 3,

changeMonth: true

})

Options:

**stepMonths** 🡪Three months step backward or forward

**changeMonth** 🡪User can choose the month from a drop-down list

**changeYear** 🡪 User can change the year

### Button/Button Set

$("#radio")**.buttonset();**

$("input[type='radio']")**.button();**

### Slider

$("#slider")**.slider();**

$("#slider").slider({

value: 0,

min: 0,

max: 100,

step: 10,

orientation: "horizontal",

slide: function(event, ui){

$("#slider\_value").val(ui.value);

}

});

Options

**value** 🡪 It tells the slider what value to start with

**min** 🡪 It tells the slider the lowest value a user can enter

**max** 🡪 It tells the slider the highest value a user can enter

**step 🡪** It tells the slider what increments we want the values in

**orientation 🡪** It can be horizontal or vertical

**slide 🡪** It is the slide event handler. The user triggers the slide event when the slide is moved. The slide event is attached to a function callback. When the function runs, this sets the input with the jQuery val method.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> jQuery UI</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "required/external/jquery/jquery.js"></**script**>

<**link** rel = "stylesheet" href = "required/jquery-ui.css"/>

<**script** src = "required/jquery-ui.js"></**script**>

<**script** src = "ui.js"></**script**>

</**head**>

<**body**>

<**form** id = "myForm">

<**label** for = "datepicker"> Date </**label**><**br**/>

<**input** type = "text" name = "datepicker" id = "datepicker"/><**br**/><**br**/>

<**div** id = "radio">

<**label** for = "gender"> Gender </**label**><**br**/>

<**input** type = "radio" name = "gender" id = "gender1" value = "male">

<**label** for = "gender1"> Male </**label**>

<**input** type = "radio" name = "gender" id = "gender2" value = "female">

<**label** for = "gender2"> Female </**label**>

</**div**><**br**/><**br**/>

<**input** type = "text" id = "slider\_value" readonly = "readonly"><**br**/>

<**div** id = "slider"></**div**><**br**/><**br**/>

<**canvas** id = "display" height = "100px" width = "100px"></**canvas**><**br**/>

<**input** type = "text" id = "display\_value" readonly = "readonly"><**br**/><**br**/>

<**div** id = "red"></**div**>

<**input** type = "text" id = "red\_value" readonly = "readonly"><**br**/><**br**/>

<**div** id = "green"></**div**>

<**input** type = "text" id = "green\_value" readonly = "readonly"><**br**/><**br**/>

<**div** id = "blue"></**div**>

<**input** type = "text" id = "blue\_value" readonly = "readonly"><**br**/><**br**/>

<**input** type = "submit" name = "submit" value = "Confirm"/>

</**form**>

<**style**>

#myForm{

**width**: 15%;

**text-align**: **left**;

}

</**style**>

</**body**>

</**html**>

$(document).ready(**function**(){

$("#datepicker").datepicker({

changeMonth: **true**,

changeYear: **true**

});

$("#radio").buttonset();

*// $("input[type='radio']").button();*

$("#slider").slider({

value: 0,

min: 0,

max: 100,

step: 10,

orientation: "horizontal",

slide: **function**(event, ui){

$("#slider\_value").val(ui.value);

}

});

$("#red, #green, #blue").slider({

value: 127,

min: 0,

max: 255,

orientation: "horizontal",

slide: refreshDisplay,

change: refreshDisplay

});

**function** refreshDisplay(){

**var** red\_value = $("#red").slider("value");

**var** green\_value = $("#green").slider("value");

**var** blue\_value = $("#blue").slider("value");

$("#red\_value").val(red\_value);

$("#green\_value").val(green\_value);

$("#blue\_value").val(blue\_value);

**var** string = red\_value + ', ' + green\_value + ", " + blue\_value;

$("#display\_value").val(string);

**var** display = document.getElementById("display");

**var** image = display.getContext('2d');

image.fillRect(0, 0, 50, 50);

image.fillStyle = "rgba("+ red\_value + "," +

green\_value + "," +

blue\_value + ")";

}

});

**Code 8: ui.js**

## CDNs

Content delivery networks, or content distribution networks) are large networks of servers, designed to store and deliver information—data, software, API code, media files or videos, etc. – making it easily accessible on the Web. Windows Azure and Amazon CloudFront are examples of traditional CDNs.

## Namespace

**$.noConflict();**

**jQuery(document.ready(function($){});**

We will be only need to use this if we plan to use other JavaScript librarires that use $ as a reference.

## Queues

Queues in jQuery are mostly used for animations. There is an array of functions stored on a per-element basic, using jQuery.data. There are firs-in-first=out(FIFO). We can add a function to the queue by calling **.queue and we remove (by calling) the function using .dequeue**

Every element can have one to many queues of functions attached to it by jQuery. In most applications, only one queue (called **fx**) is used. queues allow a sequence of actions to be called on an element asynchronously., without halting program execution.

# W3CSS

W3.CSS is a modern CSS framework with built-in responsiveness

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> W3CSS Basics </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-container w3-teal">

<**h1**> Header </**h1**>

</**div**>

<**div** class = "w3-container">

<**img** src = "images/desktop.ico"/>

</**div**>

<**div** class = "w3-container w3-teal">

<**p**>Footer</**p**>

</**div**>

</**body**>

</**html**>

**Code 1: basic.html**

## Colors

**w3-color** classes are inspired by modern colors used in marketing, road signs, and sticky notes. It changes the background color

Ex: w3-teal, w3-pale-blue, w3-khaki, w3-lime

**w3-text-color** classes changes the color of the text

Ex: w3-text-teal, w3-text-pale-blue, w3-text-khai, w3-text-lime

**w3-hover-color,** it changes color when hovered

Ex: w3-hover-teal, w3-hover-khaki

**w3-hover-text-color**

Ex: w3-hover-text-teal, w3-hover-text-lime

## Containers

The **w3-container class** **adds a 16px** **left and right padding** to any HTML element.

The w3-container class is the perfect class to use for all HTML container elements like:

**<div>, <article>, <section>, <header>, <footer>, <form>, and more.**

Containers Provides Equality

The w3-container provides equality for all HTML container elements:

* Common margins
* Common paddings
* Common alignments
* Common fonts
* Common colors

Ex: **w3-container**

<form class = "w3-container"></form>

## Panels

The **w3-panel** **class** adds a **16px top** and **bottom margin** and a **16px left** and **right padding** to anyHTML element.

* *Panels are used for notes* **🡪** Use pale color

<div class = “w3-panel **w3-light-blue**”>

<p>Great Display</p>

</div>

* *Panels are used for quotes* **🡪** Use italic, change font to serif, make it larger

<div class = “w3-panel w3-light-blue **w3-xlarge w3-serif**”>

<p>**<i>**Be Simple!**<i>**</p>

</div>

* *Panels for alerts* **🡪** Use Strong Colors

<div class = “w3-panel **w3-red**”>

<p>Warning</p>

</div>

* *Panels as cards* **🡪** Gives shadow effect

<div class = “w3-panel w3-sand **w3-card-4**”>

<p>Shadow</p>

</div>

* *Panels with rounded corners*

<div class = “w3-panel w3-sand **w3-round-xlarge**”>

<p>Rounded Corners</p>

</div>

## Hide and Show

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> W3CSS Basics </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

<**script** src = "basic.js"></**script**>

</**head**>

<**body**>

<**div** class = "w3-container w3-blue w3-center w3-text-black w3-hover-cyan w3-hover-text-white">

<**h1**> Feeling Down? </**h1**>

</**div**>

<**section** class = "w3-container w3-center w3-pale-blue w3-text-cyan">

<**h2**> Please Fill up the form</**h2**>

</**section**>

<**form** class = "w3-container w3-center">

<**label** for = "fullname"> Full Name </**label**><**br**/>

<**input** type = "text" name = "fullname" id = "fullname"/><**br**/>

<**label** for = "age"> Age </**label**><**br**/>

<**input** type = "number" name = "age" id = "age"/><**br**/><**br**/>

<**input** type = "submit" name = "submit" value = "Confirm">

</**form**><**br**/><**br**/>

<**div** id = "show"></**div**>

<**article** class = "w3-panel w3-sand w3-card-4 w3-serif w3-xlarge w3-round-xlarge" id = "article">

<**span** class = "w3-button w3-right w3-text-lime">x</**span**>

<**p** class = "w3-center"><**i**>Make it as simple as possible, but not simpler!<**i**><**p**>

</**article**>

<**footer** class = "w3-container w3-center w3-pale-blue">

<**p**>**&copy;** 2018 All Rights Reserved Photon Enterprise</**p**>

</**footer**>

</**body**>

</**html**>

**function** init(){

**var** article = document.getElementsByTagName("span")[0];

article.addEventListener("click", eventHandler, **false**);

}

**function** eventHandler(event){

**var** target = event.target;

target.parentElement.style.display = "none";

**var** button = document.createElement("button");

button.setAttribute("class", "w3-button w3-round-xxlarge");

button.setAttribute("id", "showButton");

**var** text = document.createTextNode("Show");

button.append(text);

**var** div = document.getElementById("show");

div.setAttribute("class", "w3-center w3-cyan w3-round-xxlarge");

div.append(button);

**var** showButton = document.getElementById("showButton");

**if**(showButton){

showButton.onclick = **function**(event){

**var** article = document.getElementById("article");

article.style.display = "block";

**var** div = document.getElementById("show");

div.removeChild(event.target);

};

}

}

window.onload = init;

**Code 2: hideshow.js**

## Borders

**w3-border** 🡪 Adds borders (top, right, bottom, left) to an element

**w3-border-top** 🡪 Adds a top border to an element

**w3-border-right** 🡪 Adds a right border to an element

**w3-border-bottom** 🡪 Adds a bottom border to an element

**w3-border-left** 🡪 Adds a left border to an element

**w3-border-0** 🡪 Removes all borders

**w3-border-color** 🡪 Displays the border in a specified color (like red, blue, etc)

**w3-hover-border-color** 🡪 Adds a hoverable border color

**w3-bottombar** 🡪 Adds a thick bottom border to an element

**w3-leftbar** 🡪 Adds a thick left border to an element

**w3-rightbar** 🡪 Adds a thick right border to an element

**w3-topbar** 🡪 Adds a thick top border to an element

**Rounded Borders**

w3-round 🡪 round border

w3-round-small 🡪 small rounded border

w3-round-large 🡪 large rounded border

w3-round-xlarge 🡪 extra-large rounded border

w3-round-xxlarge 🡪 double-extra-large rounded border

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> W3CSS Borders </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-center w3-border-pink w3-leftbar

w3-hover-border-black w3-black w3-container">

<**h1** class = "w3-text-pink">

<**b**> Hovering Around</**b**>

</**h1**>

</**div**>

<**div** class = "w3-panel w3-pale-red w3-border

w3-border-pink w3-hover-border-black

w3-round-xxlarge">

<**p**> Panel the ground </**p**>

</**div**>

<**div** class = "w3-panel w3-border-white

w3-leftbar w3-hover-border-green">

<**p**> This ain't the ground </**p**>

</**div**>

</**body**>

</**html**>

**Code 2: border.html**

## Cards

W3.CSS provides the following classes for displaying paper-like cards

**w3-card** Same as w3-card-2

**w3-card-2** Container for any HTML content (2px bordered shadow)

**w3-card-4** Container for any HTML content (4px bordered shadow)

* *Creating a shadow effect* 🡪 **w3-hover-shadow**
* *Create a round picture* 🡪 **w3-circle**
* *Cards as Photo Card*

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> W3CSS Cards </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-card-4 w3-center" style = "width:10%">

<**img** src = "images/desktop.ico" alt = "desktop" class = "w3-circle"/>

<**div** class = "w3-container w3-center">

<**p** class = "w3-hover-shadow"> Desktop <**p**>

</**div**>

</**div**>

</**body**>

</**html**>

**Code 3: cards.html**

## Fonts

**w3-tiny**

**w3-small**

**w3-medium** 🡪 (Default)

**w3-large**

**w3-xlarge**

**w3-xxlarge**

**w3-xxxlarge**

**w3-jumbo**

Text Alignment 🡪 **w3-left-align** and the **w3-right-align** classes are used to align text.

Centering Elements 🡪 **w3-center** class is used to center-align elements:

Wider Text 🡪 The **w3-wide** class specifies a wider text:

Text Opacity 🡪 **w3-opacity**

## Padding

**w3-padding-16** Adds 16px top and bottom padding to an element

**w3-padding-24** Adds 24px top and bottom padding to an element

**w3-padding-32** Adds 32px top and bottom padding to an element

**w3-padding-48** Adds 48px top and bottom padding to an element

**w3-padding-64** Adds 64px top and bottom padding to an element

**w3-padding** Adds 8px top and bottom, and 16px left and right padding

**w3-padding-small** Adds 4px top and bottom, and 8px left and right padding

**w3-padding-large** Adds 12px top and bottom, and 24px left and right padding

## Margin

**w3-margin** Adds a 16px margin to all sides of an element

**w3-margin-top** Adds a 16px top margin to an element

**w3-margin-right** Adds a 16px right margin to an element

**w3-margin-bottom** Adds a 16px bottom margin to an element

**w3-margin-left** Adds a 16px left margin to an element

**w3-section** Adds a 16px top and bottom margin to an element

**Padding** vs **Margin**

|  |  |
| --- | --- |
| Increase the size inside the <div> **padding**  <div class = “w3-padding”></div> | Increase the size outside the <div> **margin**  <div class = “w3-margin”></div> |

## Display

We need to **mention** the **height** of the **display container** to make it work

To include an **image, we** need to **mention** the **width** and **height** of the display container

**w3-display-container** Container for w3-display-classes

**w3-display-topleft** Displays content at the top left corner of the w3-display-container

**w3-display-topright** Displays content at the top right corner of the w3-display-container

**w3-display-bottomleft** Displays content at the bottom left corner of the w3-display-container

**w3-display-bottomright** Displays content at the bottom right corner of the w3-display-container

**w3-display-left** Displays content to the left (middle left) of the w3-display-container

**w3-display-right** Displays content to the right (middle right) of the w3-display-container

**w3-display-middle** Displays content in the middle (center) of the w3-display-container

**w3-display-topmiddle** Displays content at the top middle of the w3-display-container

**w3-display-bottommiddle** Displays content at the bottom middle of the w3-display-container

**w3-display-position** Displays content at a specified position in the w3-display-container

**w3-display-hover** Displays content on hover inside the w3-display-container

**w3-left** Floats an element to the left (float: left)

**w3-right** Floats an element to the right (float: right)

**w3-show** Shows an element (display: block)

**w3-hide** Hides an element (display: none)

**w3-mobile** Adds mobile-first responsiveness to any element.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Display </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-display-container w3-green" style="height:300px">

<**img** src = "images/lightning.png" width = "100%" height = "100%"/>

<**div** class="w3-padding w3-display-topleft">Top Left</**div**>

<**div** class="w3-padding w3-display-topmiddle">Top Mid</**div**>

<**div** class="w3-padding w3-display-topright">Top Right</**div**>

<**div** class="w3-padding w3-display-left w3-display-hover">Left</**div**>

<**div** class="w3-padding w3-display-middle">Middle</**div**>

<**div** class="w3-padding w3-display-right w3-display-hover">Right</**div**>

<**div** class="w3-padding w3-display-bottomleft">Bottom Left</**div**>

<**div** class="w3-padding w3-display-bottommiddle">Bottom Mid</**div**>

<**div** class="w3-padding w3-display-bottomright">Bottom Right</**div**>

</**div**>

<**div** class = "w3-display-container w3-hover-opacity" style="height:200px;width:200px">

<**img** src = "images/desktop.ico" width = "100%" height = "100%">

<**div** class = "w3-display-middle w3-display-hover">

<**button** class = "w3-button w3-black"> Hello </**button**>

</**div**>

</**div**>

<**div** class="w3-bar w3-light-grey">

<**div** class="w3-left w3-red">w3-left</**div**>

<**div** class="w3-right w3-blue">w3-right</**div**>

</**div**>

</**body**>

</**html**>

**Code 3: display.html**

## Buttons

**w3-btn** A rectangular button with a shadow hover effect.

**w3-button** A rectangular button with a gray hover effect.

**w3-bar** A horizontal bar that can be used to group buttons together. Horizontal navigation menu

**w3-block** Class that can be used to define a full width (100%) button.

**w3-circle** Can be used to define a circular button.

**w3-ripple** Can be used to create a ripple effect.

<**div** class = "w3-bar">

<**button** class = "w3-button w3-circle w3-teal w3-xlarge">+</**button**>

<**button** class = "w3-button w3-circle w3-red w3-xlarge">+</**button**>

</**div**>

## Table

**w3-table** Container for an HTML table

**w3-striped** Striped table

**w3-border**  Bordered table

**w3-bordered** Bordered lines

**w3-centered** Centered table content

**w3-hoverable** Hoverable table

**w3-table-all** All properties set

**<table class = "w3-table w3-striped w3-bordered**

**w3-border" style="width:10%;">**

This can be converted into a single word w3-table-all

**<table class = "w3-table-all" style="width:10%;">**

**w3-responsive** class creates a responsive table. The table will then scroll horizontally on small screens. When viewing on large screens, there is no difference.

## List

The **w3-ul** class is used to display a basic list

<**ul** class = "w3-ul w3-border w3-margin w3-card" style="width:20%">

<**li** class = "w3-bar">

<**image** src = "images/desktop.ico" class = "w3-bar-item w3-circle" style="width:20%;"/>

<**button** class = "w3-bar-item w3-button w3-right">x</**button**>

<**span** class = "w3-bar-item">Jill</**span**>

</**li**>

</u>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> W3CSS Utilities </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-bar w3-margin">

<**button** class = "w3-button w3-circle w3-teal w3-xlarge">+</**button**>

<**button** class = "w3-button w3-circle w3-red w3-xlarge">+</**button**>

</**div**>

*<!-- <table class = "w3-table w3-striped w3-bordered*

*w3-border" style="width:10%;"> -->*

<**table** class = "w3-table-all w3-hoverable w3-card w3-margin" style="width:10%;">

<**tr** class = "w3-pale-blue w3-hover-blue">

<**th**>Students</**th**>

<**th**>Score</**th**>

</**tr**>

<**tr**>

<**td**>Jill</**td**>

<**td**>89</**td**>

</**tr**>

<**tr**>

<**td**>Eve</**td**>

<**td**>12</**td**>

</**tr**>

<**tr**>

<**td**>Adam</**td**>

<**td**>76</**td**>

</**tr**>

</**table**>

<**ul** class = "w3-ul w3-border w3-margin w3-card" style="width:20%">

<**li** class = "w3-bar">

<**image** src = "images/desktop.ico" class = "w3-bar-item w3-circle" style="width:20%;"/>

<**button** class = "w3-bar-item w3-button w3-right">x</**button**>

<**span** class = "w3-bar-item">Jill</**span**>

</**li**>

<**li** class = "w3-bar">

<**image** src = "images/trash.ico" class = "w3-bar-item w3-circle" style="width:20%;"/>

<**button** class = "w3-bar-item w3-button w3-right">x</**button**>

<**span** class = "w3-bar-item">Eve</**span**>

</**li**>

<**li** class = "w3-bar">

<**image** src = "images/hdd.ico" class = "w3-bar-item w3-circle" style="width:20%;"/>

<**button** class = "w3-bar-item w3-button w3-right">x</**button**>

<**span** class = "w3-bar-item">Adam</**span**>

</**li**>

</**ul**>

</**body**>

</**html**>

**Code 4: utilities.html**

## Images

**w3-circle** 🡪 Shapes an image to a circle:

**w3-border** 🡪 Adds borders around the image:

**w3-card-\*** 🡪 <div> around the <img> element to display it as a card (add shadows):

**w3-opacity** 🡪 Make images transparent:

**w3-grayscale** 🡪 Add a grayscale effect to an image:

**w3-sepia** 🡪 Add a sepia effect to an image:

We can also add special effects on hover/mouse-over.

w3-hover-opacity class adds transparency to the image on mouse-over, and the w3-hover-opacity-off class removes transparency on mouse-over.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Images </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**img** src = "images/lightning.png" class = "w3-circle w3-border w3-border-black" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-round-xxlarge w3-border w3-border-black" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-border-black w3-padding" style = "width:20%"/>

<**div** class = "w3-card w3-margin" style = "width:20%">

<**img** src = "images/lightning.png"

class = "w3-border w3-padding"

style = "width:100%"/>

</**div**>

<**img** src = "images/lightning.png" class = "w3-border w3-opacity w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-opacity-min w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-opacity-max w3-margin" style = "width:20%"/><**br**/>

<**img** src = "images/lightning.png" class = "w3-border w3-grayscale w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-grayscale-min w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-grayscale-max w3-margin" style = "width:20%"/><**br**/>

<**img** src = "images/lightning.png" class = "w3-border w3-sepia w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-sepia-min w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-sepia-max w3-margin" style = "width:20%"/><**br**/>

<**img** src = "images/lightning.png" class = "w3-border w3-hover-opacity w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-hover-grayscale w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-hover-sepia w3-margin" style = "width:20%"/>

<**img** src = "images/lightning.png" class = "w3-border w3-opacity w3-hover-opacity-off w3-margin" style = "width:20%"/>

</**body**>

</**html**>

**Code 5: images.html**

## Form

**w3-input 🡪** Input of the form

**w3-animate-input 🡪** Animation of the input form

**w3-check 🡪** Checkbox of the form

**w3-radio 🡪** Radio of the form

**w3-select 🡪** Select of the form

## Badges

**w3-badge 🡪** Creates a circular badge

**w3-size** 🡪 w3-tiny, w3-small, w3-large, w3-xlarge, w3-xxlarge, w3-xxxlarge, w3-jumbo

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Form </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**form** class = "w3-container w3-padding w3-card" style = "width:20%;">

<**h1** class = "w3-green w3-padding"> Registration Form </**h1**>

<**label** for = "username" class = "w3-text-green"> Username </**label**>

<**input** type = "text" name = "username"

style="width:50%"

class = "w3-input w3-border w3-round w3-animate-input"/>

<**label** for = "age" class = "w3-text-green"> Age </**label**>

<**input** type = "number" name = "age" class = "w3-input w3-border w3-round"/><**br**/>

<**label** for = "grocery" class = "w3-text-green"> Groceries </**label**><**br**/>

<**div** class = "w3-container w3-card">

<**input** type = "checkbox" name = "grocery[]" value = "milk" class = "w3-check"/>

<**label**> Milk</**label**><**br**/>

<**input** type = "checkbox" name = "grocery[]" value = "sugar" class = "w3-check"/>

<**label**> Sugar</**label**><**br**/>

<**input** type = "checkbox" name = "grocery[]" value = "lemon" class = "w3-check" disabled/>

<**label**>Lemon</**label**><**br**/>

</**div**>

<**label** for = "gender" class = "w3-text-green"> Gender </**label**><**br**/>

<**div** class = "w3-container w3-card">

<**input** type = "radio" name = "gender" value = "male" class = "w3-radio"/>

<**label**> Male</**label**><**br**/>

<**input** type = "radio" name = "gender" value = "female" class = "w3-radio"/>

<**label**> Female</**label**><**br**/>

</**div**>

<**label** for = "car" class = "w3-text-green">Car</**label**>

<**select** class = "w3-select w3-border" name = "car">

<**option** value = "" disabled selected> Choose your Option </**option**>

<**option** value = "toyota">Toyota</**option**>

<**option** value = "honda">Honda</**option**>

<**option** value = "mazda">Mazda</**option**>

</**select**><**br**/><**br**/>

<**button** class = "w3-button w3-green w3-border">Confirm <**span** class = "w3-badge w3-black">9</**span**></**button**>

</**form**><**br**/><**br**/>

<**p** id = "updates" class = "w3-padding">

<**span** class = "w3-tag w3-teal w3-spin"> New updates!</**span**>

</**p**>

<**style**>

*/\*#updates {transform:rotate(-5deg)}\*/*

</**style**>

</**body**>

</**html**>

**Code 6: form.html**

With W3.CSS we can use the icon library you like, such as:

* Font Awesome Icons
* Google Material Design Icons
* Bootstrap Icons

## Responsive

W3.CSS's grid system is responsive, and the columns will re-arrange automatically depending on the screen size

**w3-half** Occupies 1/2 of the window (on medium and large screens)

**w3-third** Occupies 1/3 of the window (on medium and large screens)

**w3-twothird** Occupies 2/3 of the window (on medium and large screens)

**w3-quarter** Occupies 1/4 of the window (on medium and large screens)

**w3-threequarter** Occupies 3/4 of the window (on medium and large screens)

**w3-rest**  Occupies the rest of the column width

**w3-col** Defines one column in a 12-column responsive grid

**w3-mobile** Adds mobile-first responsiveness to a cell (column).

Displays elements as block elements on mobile devices.

The responsive classes above must be placed inside a **w3-row class (**or **w3-row-padding** class)to be **fully responsive.**

**w3-row** Container for responsive classes, with no padding

**w3-row-padding** Container for responsive classes, with 8px left and right padding

**w3-content** Container for fixed size centered content

**w3-hide-small** Hides content on small screens (less than 601px)

**w3-hide-medium** Hides content on medium screens

**w3-hide-large** Hides content on large screens (larger than 992px)

**l1 - l12**  Responsive sizes for large screens

**m1 - m12** Responsive sizes for medium screens

**s1 - s12** Responsive sizes for small screens

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Responsive</**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-row">

<**div** class = "w3-half w3-container w3-black w3-hover-white">

<**h1** class = "w3-right">Yin</**h1**>

</**div**>

<**div** class = "w3-half w3-container w3-hover-black w3-hover-text-white

w3-text-black w3-hover-text-white">

<**h1** class = "w3-left ">Yang</**h1**>

</**div**>

</**div**>

<**div** class = "w3-row">

<**div** class = "w3-half w3-container">

<**h1** class = "w3-right">Yin</**h1**>

</**div**>

<**div** class = "w3-half w3-container">

<**h1** class = "w3-left">Yang</**h1**>

</**div**>

</**div**>

<**div** class = "w3-row">

<**div** class = "w3-third w3-display-container" style = "height:150px;">

<**h1** class>

<**img** src = "images/yinyang.png" style = "width:20%;" class = "w3-display-left"/>

</**h1**>

</**div**>

<**div** class = "w3-third w3-display-container" style = "height:150px;">

<**h1** class>

<**img** src = "images/ruby.png" style = "width:20%;" class = "w3-display-middle"/>

</**h1**>

</**div**>

<**div** class = "w3-third w3-display-container" style = "height:150px;">

<**h1** class>

<**img** src = "images/yinyang.png" style = "width:20%;" class = "w3-display-right"/>

</**h1**>

</**div**>

</**div**>

<**div** class = "w3-row">

<**div** class = "w3-twothird w3-container w3-border">

<**h1** class = "w3-right">Boom</**h1**>

</**div**>

<**div** class = "w3-third w3-container w3-border">

<**h1** class = "w3-left">Shakalaka</**h1**>

</**div**>

</**div**>

<**div** class = "w3-row">

<**div** class = "w3-quarter w3-container w3-border">

<**h1** class = "w3-right">Boom</**h1**>

</**div**>

<**div** class = "w3-quarter w3-container w3-border">

<**h1** class = "w3-left">Shakalaka</**h1**>

</**div**>

<**div** class = "w3-quarter w3-container w3-border">

<**h1** class = "w3-right">Boom</**h1**>

</**div**>

<**div** class = "w3-quarter w3-container w3-border">

<**h1** class = "w3-left">Shakalaka</**h1**>

</**div**>

</**div**>

</**body**>

</**html**>

**Code 7: responsive.html**

## Cells

"column-like" layout

**w3-cell-row** Container for cells (columns).

**w3-cell**  Layout cell (column).

**w3-cell-top** Aligns content at the top of a cell (column).

**w3-cell-middle** Aligns content at the vertical middle of a cell (column).

**w3-cell-bottom** Aligns content at the bottom of a cell (column).

**w3-mobile** Adds mobile-first responsiveness to a cell (column).

## Animation

**w3-animate-top** 🡪 Slides in an element from the top (-300px to 0)

**w3-animate-bottom** 🡪 Slides in an element from the bottom (-300px to 0)

**w3-animate-left** 🡪 Slides in an element from the left (-300px to 0)

**w3-animate-right** 🡪 Slides in an element from the right (-300px to 0)

**w3-animate-opacity** 🡪 Animates an element's opacity from 0 to 1 in 1.5 seconds

**w3-animate-zoom** 🡪 Animates an element from 0 to 100% in size

**w3-animate-fading** 🡪 Animates an element's opacity from 0 to 1 and 1 to 0

**w3-spin** 🡪 Spins element 360 degrees

## Navigation Bar

**w3-bar, w3-bar-item, w3-dropdown-hover, w3-dropdown-content**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Layout</**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-container">

<**h1** class = "w3-left w3-text-green">Welcome to Boom Shakalaka!</**h2**>

<**h3** class = "w3-right w3-text-gray ">It's party time!</**h3**>

</**div**>

<**div** class = "w3-container w3-bar w3-green w3-margin-bottom">

<**a** href = "#" class = "w3-bar-item w3-button">Home</**a**>

<**div** class = "w3-dropdown-hover">

<**a** href = "#" class = "w3-button">Study</**a**>

<**div** class = "w3-dropdown-content w3-bar-block">

<**a** href = "#" class = "w3-button w3-bar-item">Physics</**a**>

<**a** href = "#" class = "w3-button w3-bar-item">Math</**a**>

<**a** href = "#" class = "w3-button w3-bar-item">Chemistry</**a**>

</**div**>

</**div**>

<**a** href = "#" class = "w3-bar-item w3-button w3-right">About Us</**a**>

</**div**>

<**div** class = "w3-container w3-bar-block w3-right" style="width:5%;">

<**div** class="w3-bar-item">London</**div**>

<**div** class="w3-bar-item">Paris</**div**>

<**div** class="w3-bar-item">Tokyo</**div**>

</**div**>

<**div** class = "w3-dropdown-hover w3-container">Yin

<**div** class = "w3-dropdown-content w3-card-4" style="width:250px">

<**img** src = "images/ruby.png" style="width:100%"/>

<**div** class = "w3-container">

<**p**>Yin Yang</**p**>

</**div**>

</**div**>

</**div**>

</**body**>

</**html**>

**Code 7: layout.html**

## Slideshow

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> SlideShow</**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-content w3-display-container" style = "width:100px;">

<**img** class = "mySlides" src = "images/desktop.ico" style = "width:100%">

<**img** class = "mySlides" src = "images/trash.ico" style = "width:100%">

<**img** class = "mySlides" src = "images/hdd.ico" style = "width:100%">

<**img** class = "mySlides" src = "images/folder.ico" style = "width:100%">

</**div**>

</**body**>

<**script**>

position = 0;

**function** init(){

**var** images = document.getElementsByClassName("mySlides");

**for**(**var** i = 0; i < images.length; i++){

images[i].style.display = "none";

}

position++;

**if**(position > images.length){position = 1;}

images[position-1].style.display = "block";

setTimeout(init, 3000);

}

window.onload = init;

</**script**>

</**html**>

**Code 8: slideshow.html**

## Code

**w3-code 🡪** Write codes in html

**w3-codespan 🡪** Highlight the code

**w3-code jsHigh 🡪** JavaScript code

**w3-code htmlHigh 🡪** HTML code

**w3-code cssHigh 🡪** CSS code

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Code </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**div** class = "w3-code">

fruits[0] = "Apple"; <**br**/>

fruits[1] = "Banana"; <**br**/>

fruits[2] = "Pineapple"; <**br**/>

fruits[3] = "Mango"; <**br**/>

</**div**>

<**div** class = "w3-container">

<**p**>I am using <**code** class = "w3-codespan">HTML</**code**></**p**>

</**div**>

<**div** class = "w3-code jsHigh">

var world = document.getElementById("hello")<**br**/>

world.onclick = doomsday;

</**div**>

<**div** class = "w3-code htmlHigh">

<**p**> Bro this is not cool </**p**>

</**div**>

<**div** class = "w3-code cssHigh">

.boomerang{

background-color: rgba(244,244 244,0);

}

</**div**>

</**body**>

</**html**>

**Code 9: code.html**

## Tool Tips

Tooltips display text (or other content) when you hover over an HTML element.

The **w3-tooltip** class defines the element to hover over (the tooltip container).

The **w3-text** class defines the tooltip text.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Tooltip </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**span** class = "w3-tooltip">

Hello, who are you?

<**span** class = "w3-text w3-badge">Are you the one?</**span**>

</**span**><**br**/>

<**span** class = "w3-tooltip">

Hello, who are you?

<**span** class = "w3-text w3-tag">Are you the one?</**span**>

</**span**><**br**/>

<**span** class = "w3-tooltip">

Hello, who are you?

<**span** class = "w3-text w3-tag w3-animate-opacity">Are you the one?</**span**>

</**span**>

<**div** class = "w3-display-container w3-tooltip w3-card" style="width:150px">

<**img** src = "desktop.ico" style="width:100%"/>

<**span** class = "w3-text">Great Choice</**span**>

</**div**>

</**body**>

</**html**>

**Code 10: tooltips.html**

## Modal

The **w3-modal** class defines a container for a modal.

The **w3-modal-content** class defines the modal content.

Modal content can be any HTML element (divs, headings, paragraphs, images, etc.).

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Modal </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

</**head**>

<**body**>

<**button** id = "firstButton"class = "w3-button"> Log in </**button**>

<**div** class = "w3-modal" id = "modal">

<**div** class = "w3-modal-content w3-padding" style = "width:20%;">

<**form** class = "w3-container w3-animate-opacity">

<**span** class = "w3-button w3-display-topright" id ="secondButton">**&times;**</**span**>

<**label** for = "username" class = "w3-text=green">Username</**label**>

<**input** type = "text" class = "w3-border w3-input"/>

<**label** for = "password" class = "w3-text=green">Password</**label**>

<**input** type = "password" class = "w3-border w3-input"/><**br**/>

<**button** class = "w3-button w3-border w3-green"> Confirm </**button**>

</**form**>

</**div**>

</**div**>

<**script**>

**function** init(){

**var** button = document.getElementById("firstButton");

button.onclick = **function**(){

**var** modal = document.getElementById("modal");

modal.style.display = "block";

};

**var** button2 = document.getElementById("secondButton");

button2.onclick = **function**(){

**var** modal = document.getElementById("modal");

modal.style.display = "none";

}

}

window.onload = init;

</**script**>

</**body**>

</**html**>

**Code 11: modal.html**

# SVG

SVG stands **Scalar Vector Graphics**. SVG defines vector-based graphics in XML format. SVG graphics do not lose any quality if they are zoomed or resized. Every element and every attribute in SVG files can be animated.

* We need to mention the height and width in the <svg> tag

Basic Structure

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Scalar Vector Graphics</**title**>

<**meta** charset = "UTF-8">

</**head**>

<**body**>

<**svg** id = "art">

<**circle** id = "circle"/>

</**svg**>

<**style**>

#art{

**width**: 100px;

**height**: 100px;

}

#circle{

cx: 50;

cy: 50;

r: 40;

stroke: green;

stroke-**width**: 4;

fill: yellow;

}

</**style**>

</**body**>

</**html**>

**Code 1: basic.html**

* An SVG image begins with an <svg> element
* The width and height of the SVG image defines the size
* The <circle> element is used to draw a circle
* The cx and cy attributes define the x and y coordinates of the center of the circle.
* If cx and cy are omitted, the circle's center is set to (0, 0).
* The r attribute defines the radius of the circle
* The stroke and stroke-width attributes control how the outline of a shape appears.
* The fill attribute refers to the color inside the circle.

## Predefined shape

**<rect> 🡪** Rectangle

**<circle> 🡪** Circle

**<ellipse> 🡪** Ellipse

**<line> 🡪** Line

**<polyline>** 🡪 Polyline

**<polygon>** 🡪 Polygon

**<path>** 🡪 path

**<g>** 🡪 group

### Common properties

**stroke:** Color of the border

**stroke-width:** Width of the border

**fill:** Color inside the shape

**fill-opacity:** Changesthe opacity of the color inside the shape

**stroke-opacity:** Changesthe opacity of the border color of the shape

**opacity:** Changes both the fill and stroke opacity

**<rect>** specific properties (Should be mentioned in tag)

**x:** initial position of the rectangle

**y:** initial position of the rectangle

**rx:** needed for the rounded corners of the rectangle

**ry:** needed for the rounded corners of the rectangle

**width:** length of the rectangle

**height:** height of the rectangle

**<circle>** specific properties (Should be mentioned in tag)

**cx:** initial position of the center circle

**cy:** initial position of the center circle

**r:** radius of the circle

**<ellipse>** specific properties (Should be mentioned in tag)

**cx**: initial position of the center ellipse

**cy**: initial position of the center ellipse

**rx:** horizontal radius of the ellipse

**ry:** vertical radius of the ellipse

**<line>** specific properties (Should be mentioned in tag)

**x1**: Initial position

**y1**: Initial position

**x2**: Final position

**y2**: Final position

**<polygon>** specific properties (Should be mentioned in tag)

The <polygon> element is used to create a graphic that contains at least three sides.

Polygons are made of straight lines, and the shape is "closed" (all the lines connect up).

The **points attribute** defines the x and y coordinates for each corner of the polygon

**points:** All the x and y positions

**<polyline>** specific properties (Should be mentioned in tag)

**points:** All the x and y positions

**<path>**

This element is used to define a path.

* M = moveto
* L = lineto
* H = horizontal lineto
* V = vertical lineto
* C = curveto
* S = smooth curveto
* Q = quadratic Bézier curve
* T = smooth quadratic Bézier curveto
* A = elliptical Arc
* Z = closepath

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Scalar Vector Graphics</**title**>

<**meta** charset = "UTF-8">

</**head**>

<**body**>

<**svg** class = "art">

<**circle** id = "circle" cx = "50" cy = "50" r = "40"/>

</**svg**>

<**svg** class = "art">

<**rect** id = "rectangle" x = "0" y = "0" width = "100" height = "100"/>

</**svg**>

<**svg** class = "art">

<**rect** id = "rectangle2" x = "0" y = "0" rx = "10" ry ="10" width = "100" height = "100"/>

</**svg**>

<**svg** class = "art">

<**ellipse** id = "ellipse" cx = "50" cy = "50" rx = "40" ry = "20"/>

</**svg**>

<**svg** class = "art">

<**ellipse** id = "ellipse1" cx = "50" cy = "50" rx = "50" ry = "20"/>

<**ellipse** id = "ellipse2" cx = "50" cy = "40" rx = "40" ry = "15" />

<**ellipse** id = "ellipse3" cx = "50" cy = "30" rx = "30" ry = "10"/>

</**svg**>

<**style**>

**.art**{

**width**: 100px;

**height**: 100px;

}

#circle{

stroke: green;

stroke-**width**: 4;

fill: yellow;

}

#rectangle{

fill: rgba(0, 123, 123, 0.8);

stroke: maroon;

stroke-**width**: 4;

fill-**opacity**: 0.1;

stroke-**opacity**: 0.5;

}

#rectangle2{

fill: rgba(0, 123, 123, 0.8);

stroke: maroon;

stroke-**width**: 4;

**opacity**: 0.5;

}

#ellipse{

stroke: green;

stroke-**width**: 4;

fill: yellow;

}

#ellipse1{

fill: purple;

}

#ellipse2{

fill: lime;

}

#ellipse3{

fill: blue;

}

</**style**>

</**body**>

</**html**>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Scalar Vector Graphics</**title**>

<**meta** charset = "UTF-8">

</**head**>

<**body**>

<**svg** class = "art">

<**line** x1="0" y1="0" x2="50" y2="50" id = "line"/>

</**svg**>

<**svg** class = "art">

<**polygon** id = "polygon" points = "0,0,70,80,10,70,0,0"/>

</**svg**>

<**svg** class = "art">

<**polygon** id = "polygon2" points = "0,30,100,30,20,90,50,0,80,90,0,30"/>

</**svg**>

<**svg** class = "art">

<**polygon** id = "polygon3" points = "0,30,100,30,20,90,50,0,80,90,0,30"/>

</**svg**>

<**svg** class = "art">

<**polyline** id = "polyline" points = "0,0,30,0,30,30,60,30,60,60,90,60,90,90"/>

</**svg**>

<**svg** class = "art">

<**path** id = "path" d = "M50,0 L100,100 L0,100 L50,0 Z"/>

</**svg**>

<**svg** class = "art">

<**path** id = "path2" d = "M50,0 L100,100 L0,100 L50,0 Z"/>

</**svg**>

<**svg** class = "art">

<**path** id = "abc" d = "M50,0 L100,100 Z"/>

<**path** id = "abc" d = "M50,0 L0,100 Z"/>

<**path** id = "abc" d = "M25,50 L75,50 Z"/>

<**path** id = "curve" d = "M0,100 Q50,0 100,100Z"/>

</**svg**>

</**body**>

<**style**>

**.art**{

**width**: 100px;

**height**: 100px;

}

#line{

stroke: blue;

stroke-**width**: 2;

}

#polygon{

stroke: red;

stroke-**width**: 4;

fill: white;

}

#polygon2{

stroke: red;

stroke-**width**: 1;

fill-rule: nonzero;

}

#polygon3{

stroke: red;

stroke-**width**: 1;

fill-rule: evenodd;

}

#polyline{

stroke: red;

stroke-**width**: 1;

fill: white;

}

#path{

stroke: blue;

stroke-**width**: 1;

fill: white;

}

#path2{

stroke: blue;

stroke-**width**: 1;

fill: white;

}

#abc{

stroke: blue;

stroke-**width**: 1;

fill: white;

}

#curve{

stroke: red;

stroke-**width**: 1;

fill: white;

}

</**style**>

</**html**>

**Code 2: shape.html**

## Text

**<text>** 🡪 text

**<tspan>** 🡪 go to the next line

**<text>, <tspan>** specific properties (Should be mentioned in tag)

**x:** initial position of the rectangle

**y:** initial position of the rectangle

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Text</**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**svg**>

<**text** class = "text" x = "50" y = "50">

Several Lines:

<**tspan** x = "50" y = "70"> First Line</**tspan**>

<**tspan** x = "50" y = "90"> Second Line</**tspan**>

</**text**>

</**svg**>

<**svg** xmlns:xlink="http://sphotonkhan.com">

<**a** xlink:href = "http://sphotonkhan.com" target="\_blank">

<**text** class = "text" x = "50" y = "50">

Photon

</**text**>

</**a**>

</**svg**>

</**body**>

<**style**>

**svg**{

**width**: 200px;

**height**: 200px;

}

**.text**{

fill: rgba(0, 255, 0, 0.6);

stroke: rgba(0, 0, 255, 0.3);

stroke-**width**: 2;

transform: rotate(20deg);

*/\*transform: skewY(20deg);\*/*

*/\*transform: scaleY(1.5);\*/*

}

</**style**>

</**html**>

**Code 3: text.html**

## Stroke

SVG offers a wide range of stroke properties.

**stroke 🡪** defines the color of a line, text or outline of an element:

**stroke-width 🡪** defines the thickness of a line, text or outline of an element

**stroke-linecap 🡪** defines different types of endings to an open path

stroke-linecap properties 🡪 butt, round, square

**stroke-dasharray 🡪** is used to create dashed lines

All the stroke properties can be applied to any kind of lines, text and outlines of elements like a circle.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Stroke</**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**svg**>

<**line** id = "line1" x1 = "0" y1 = "20" x2 = "100" y2 = "20"/>

<**line** id = "line2" x1 = "0" y1 = "40" x2 = "100" y2 = "40"/>

<**line** id = "line3" x1 = "0" y1 = "60" x2 = "100" y2 = "60"/>

<**line** id = "line4" x1 = "0" y1 = "80" x2 = "100" y2 = "80"/>

</**svg**>

<**style**>

**svg**{

**width**: 100px;

**height**: 100px;

}

#line1{

stroke: black;

stroke-**width**: 4;

stroke-linecap: butt;

stroke-dasharray: 5,5;

}

#line2{

stroke: black;

stroke-**width**: 4;

stroke-linecap: round;

stroke-dasharray: 10,10;

}

#line3{

stroke: black;

stroke-**width**: 4;

stroke-linecap: **square**;

stroke-dasharray: 25,5,15,5,10,5;

}

#line4{

stroke: black;

stroke-**width**: 4;

stroke-linecap: butt;

}

</**style**>

</**body**>

</**html**>

**Code 4: stroke.html**

## Filters

The available filter elements in SVG are:

<feBlend> 🡪 filter for combining images

<feColorMatrix> 🡪 filter for color transforms

<feComponentTransfer>

<feComposite>

<feConvolveMatrix>

<feDiffuseLighting>

<feDisplacementMap>

<feFlood>

<feGaussianBlur>

<feImage>

<feMerge>

<feMorphology>

<feOffset> 🡪 filter for drop shadows

<feSpecularLighting>

<feTile>

<feTurbulence>

<feDistantLight> 🡪 filter for lighting

<fePointLight> 🡪 filter for lighting

<feSpotLight> 🡪 filter for lighting

**Structure**

<svg>

🡪Predefined Shapes Ex <rect> <line> <polygon> <line> <path>

🡪<defs>

🡪<filter>

🡪Filter Elements Ex: <feGlaussianBlur>

All SVG filters are defined within a <defs> element.

The <defs> element is short for definitions and contains definition of special elements (such as filters).

The <filter> element is used to define an SVG filter.

The <filter> element has a required id attribute which identifies the filter. The graphic then points to the filter to use.

**<defs> 🡪** Definition for the filter

**<filter>** specific properties mention it in the tag

**x:** Initial position

**y:** Initial position

### feGaussianBlur

**<feGaussianBlur>** properties should be in the tag

**in:** SourceGraphic

**stdDeviation:** 15

The in="SourceGraphic" part defines that the effect is created for the entire element

The stdDeviation attribute defines the amount of the blur

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Filters</**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**svg**>

<**defs**>

<**filter** id="f1" x="0" y="0">

<**feGaussianBlur** in="SourceGraphic" stdDeviation="15" />

</**filter**>

</**defs**>

<**rect** width="90" height="90"/>

</**svg**>

</**body**>

<**style**>

**svg**{

**width**: 100px;

**height**: 100px;

}

**rect**{

stroke: green;

stroke-**width**: 4;

fill: yellow;

filter: url(#f1);

}

</**style**>

</**html**>

**Code 4: feGaussianBlur.html**

### feOffset

The **<feOffset>** element is used to create drop shadow effects.

The idea is to take an SVG graphic (image or element) and move it a little bit in the xy plane.

Offset a rectangle (with <feOffset>), then blend the original on top of the offset image (with **<feBlend>**)

**<feOffset>** properties should be in the tag

**in: SourceGraphic**

**result: offOut**

**dx: 20**

**dy: 20**

**<feOffset>** properties should be in the tag

**in: SourceGraphic**

**in2: offOut 🡪 optional**

**mode: normal 🡪 optional**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>feOffset</**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**svg**>

<**defs**>

<**filter** id="f2" x="0" y="0">

<**feOffset** in="SourceGraphic" result = "offOut" dx = "10" dy = "10"/>

<**feBlend** in = "SourceGraphic"/>

</**filter**>

</**defs**>

<**rect** id = "rect2" width="90" height="90"/>

</**svg**>

</**body**>

<**style**>

**svg**{

**width**: 200px;

**height**: 200px;

}

#rect2{

stroke: green;

stroke-**width**: 4;

fill: yellow;

filter: url(#f2);

}

</**style**>

</**html**>

**Code 5: feOffset.html**

### feOffset + feBlur + feBlend + feColorMatrix

**in: SourceGraphic 🡪 Takes data using rgba**

**in: SourceAlpha 🡪 Takes data from the alpha channel**

The **<feColorMatrix>** filter is used to transform the colors in the offset image closer to black. The three values of '0.2' in the matrix all get multiplied by the red, green and blue channels. Reducing their values brings the colors closer to black (black is 0)

Concept of Color Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Red | Green | Blue | A | 0 |
| Red | 1 | 0 | 0 | 0 | 0 |
| Green | 0 | 1 | 0 | 0 | 0 |
| Blue | 0 | 0 | 1 | 0 | 0 |
| A | 0 | 0 | 0 | 1 | 0 |

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Offset Blend Blur</**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**svg**>

<**defs**>

<**filter** id="f1" x="0" y="0">

<**feOffset** in="SourceGraphic" result = "offOut" dx = "10" dy = "10"/>

<**feGaussianBlur** in = "offOut" result = "OffBlur" stdDeviation = "15"/>

<**feBlend** in = "SourceGraphic" in2 = "OffBlur" mode = "normal"/>

</**filter**>

</**defs**>

<**rect** id = "rect1" width="90" height="90"/>

</**svg**>

<**svg**>

<**defs**>

<**filter** id="f2" x="0" y="0">

<**feOffset** in="SourceAlpha" result = "offOut" dx = "10" dy = "10"/>

<**feGaussianBlur** in = "offOut" result = "OffBlur" stdDeviation = "15"/>

<**feBlend** in = "SourceGraphic" in2 = "OffBlur" mode = "normal"/>

</**filter**>

</**defs**>

<**rect** id = "rect2" width="90" height="90"/>

</**svg**>

<**svg**>

<**defs**>

<**filter** id="f3" x="0" y="0">

<**feOffset** in="SourceAlpha" result = "offOut" dx = "10" dy = "10"/>

<**feColorMatrix** result="matrixOut" in="offOut" type="matrix"

values="0.5 0 0 0 0

0 0 0 0 0

0 0 0 0 0

0 0 0 0.6 0" />

<**feGaussianBlur** in = "matrixOut" result = "OffBlur" stdDeviation = "15"/>

<**feBlend** in = "SourceGraphic" in2 = "OffBlur" mode = "normal"/>

</**filter**>

</**defs**>

<**rect** id = "rect3" width="90" height="90"/>

</**svg**>

</**body**>

<**style**>

**svg**{

**width**: 200px;

**height**: 200px;

}

#rect1{

stroke: green;

stroke-**width**: 4;

fill: yellow;

filter: url(#f1);

}

#rect2{

stroke: green;

stroke-**width**: 4;

fill: yellow;

filter: url(#f2);

}

#rect3{

stroke: green;

stroke-**width**: 4;

fill: yellow;

filter: url(#f3);

}

</**style**>

</**html**>

**Code 6: feColorMatrix.html**

## Gradients

A gradient is a smooth transition from one color to another. In addition, several color transitions can be applied to the same element.

There are two main types of gradients in SVG:

Linear

Radial

### Linear Gradient

The **<linearGradient>** element is used to define a linear gradient.

It must be nested within a <defs> tag.

Linear gradients can be defined as **horizontal, vertical or angular gradients**

**Horizontal gradients** are created when y1 and y2 are equal and x1 and x2 differ

**Vertical gradients** are created when x1 and x2 are equal and y1 and y2 differ

**Angular gradients** are created when x1 and x2 differ and y1 and y2 differ

The x1, x2, y1,y2 attributes of the <linearGradient> tag define the start and end position of the gradient

The color range for a gradient can be composed of two or more colors.

Each color is specified with a <stop> tag.

The offset attribute is used to define where the gradient color begins and end

The fill attribute links the ellipse element to the gradient

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Linear Gradient </**title**>

<**meta** charset = UTF-8/>

</**head**>

<**body**>

<**svg**>

<**defs**>

<**linearGradient** id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">

<**stop** offset="0%" class = "off1"/>

<**stop** offset="100%" class = "off2"/>

</**linearGradient**>

</**defs**>

<**rect** id = "rect1" width="100" height="100"/>

</**svg**>

<**svg**>

<**defs**>

<**linearGradient** id="grad2" x1="0%" y1="0%" x2="0%" y2="100%">

<**stop** offset="0%" class = "off1"/>

<**stop** offset="100%" class = "off2"/>

</**linearGradient**>

</**defs**>

<**rect** id = "rect2" width="100" height="100"/>

<**text** x = "35" y = "50"> Bro! </**text**>

</**svg**>

<**style**>

**svg**{

**width**: 100px;

**height**: 100px;

}

#rect1{

fill: url(#grad1);

}

#rect2{

fill: url(#grad2);

}

**.off1**{

stop-**color**:**rgb**(255,255,0);

stop-**opacity**:1;

}

**.off2**{

stop-**color**:**rgb**(255,0,0);

stop-**opacity**:1;

}

**text**{

fill: white;

}

</**style**>

</**body**>

</**html**>

**Code 7: linearGradient.html**

### Radial Gradient

The **<radialGradient>** element is used to define a radial gradient.

This element must be nested within a <defs> tag.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Radial Gradient </**title**>

<**meta** charset = "UTF-8"/>

</**head**>

<**body**>

<**svg**>

<**defs**>

<**radialGradient** id = "grad1" cx = "50%" cy = "50%" r = "50%" fx = "50%" fy = "50%">

<**stop** offset = "0%" id = "off1"/>

<**stop** offset = "100%" id = "off2"/>

</**radialGradient**>

</**defs**>

<**circle** cx = "50" cy = "50" r = "30"/>

</**svg**>

</**body**>

<**style**>

**svg**{

**width**: 100px;

**height**: 100px;

}

#off1{

stop-**color**: **rgb**(255, 255, 0);

stop-**opacity**: 1;

}

#off2{

stop-**color**: **rgb**(255, 0, 0);

stop-**opacity**: 1;

}

**circle**{

fill: url(#grad1);

}

</**style**>

</**html**>

**Code 7: radialGradient.html**

# Angular JS

## Basics

AngularJS extends HTML with new attributes.

AngularJS is perfect for Single Page Applications (SPAs).

AngularJS is a JavaScript framework. It can be added to an HTML page with a <script> tag.

AngularJS extends HTML attributes with Directives, and binds data to HTML with Expressions.

AngularJS extends HTML with ng-directives.

The ng-app directive defines an AngularJS application.

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

The ng-bind directive binds application data to the HTML view.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Angular JS Basics</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

*<!-- <div ng-app = "">*

*<label for = "name">Name</label><br/>*

*<input type = "text" ng-model = "name"/><br/>*

*<p>Hello {{name}}</p>*

*</div> -->*

<**div** ng-app = "">

<**label** for = "fullname">Full Name</**label**><**br**/>

<**input** type = "text" ng-model = "fullname"/><**br**/>

<**p** ng-bind = "fullname"></**p**>

</**div**>

</**body**>

</**html**>

**Code 1: basic.html**

AngularJS starts automatically when the web page has loaded.

The ng-app directive tells AngularJS that the <div> element is the "owner" of an AngularJS application.

The ng-model directive binds the value of the input field to the application variable name.

The ng-bind directive binds the innerHTML of the <p> element to the application variable name.

We can use the ng-bind in the same way as {{expression}}

## AngularJS Directives

As we have already seen, AngularJS directives are HTML attributes with an **ng prefix**.

The **ng-init** directive initializes AngularJS application variables.

**data-ng-init** 🡪 Directive which is similar to ng-init, if we want to make the HTML valid

**data-ng-bind** 🡪 Directive similar to ng-bind

**Two-Way Binding**

The binding goes both ways. If the user changes the value inside the input field, the AngularJS property will also change its value:

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Angular JS Basics</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

*<!-- ng-model & ng-bind -->*

<**div** ng-app = "">

<**label** for = "fullname">Full Name</**label**><**br**/>

<**input** type = "text" ng-model = "fullname"/><**br**/>

<**p** ng-bind = "fullname"></**p**>

*<!-- ng-init -->*

<**div** ng-init="owner='Photon Enterprise'">

<**p**> Powered by <**span** ng-bind = "owner"></**span**></**p**>

</**div**>

*<!-- data-ng-init (Alternative) -->*

<**div** data-ng-init="copy='&copy;'">

<**p**><**span** data-ng-bind = "copy"></**span**> 2018 All Rights Reserved</**p**>

</**div**>

</**div**>

</**body**>

</**html**>

**Code 2: directives.html**

## AngularJS Expressions

AngularJS expressions are written inside double braces: {{expression}}.

AngularJS will "output" data exactly where the expression is written.

AngularJS expressions bind AngularJS data to HTML the same way as the ng-bind directive.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Angular JS Basics</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

*<!-- ng-model & ng-bind -->*

<**div** ng-app = "">

<**label** for = "fullname">Full Name</**label**><**br**/>

<**input** type = "text" ng-model = "fullname"/><**br**/>

<**p** ng-bind = "fullname"></**p**>

*<!-- ng-init -->*

<**div** ng-init="owner='Photon Enterprise'">

<**p**> Powered by <**span** ng-bind = "owner"></**span**></**p**>

</**div**>

*<!-- data-ng-init (Alternative) -->*

<**div** data-ng-init="copy='&copy;'">

<**p**><**span** data-ng-bind = "copy"></**span**> 2018 All Rights Reserved</**p**>

</**div**>

*<!-- {{Expressions}} -->*

<**label** for = "replicate">Replicate</**label**><**br**/>

<**input** type = "text" ng-model = "replicate"/><**br**/>

<**p**> I replicate: {{replicate}} </**p**>

</**div**>

</**body**>

</**html**>

**Code 3: expressions.html**

**AngularJS Numbers**

<div ng-app="" ng-init="quantity=1;cost=5">  
 <p>Total in dollar: {{ quantity \* cost }}</p>  
</div>

**AngularJS Strings**

<div ng-app="" ng-init="firstName='John';lastName='Doe'">  
<p>The name is <span ng-bind="firstName + ' ' + lastName"></span></p>  
</div>

**AngularJS Objects**

<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">  
<p>The name is {{ person.lastName }}</p>  
</div>

**AngularJS Arrays**

div ng-app="" ng-init="points=[1,15,19,2,40]">  
<p>The third result is <span ng-bind="points[2]"></span></p>  
</div>

## AngularJS Applications

AngularJS modules define AngularJS applications.

AngularJS controllers control AngularJS applications.

The ng-app directive defines the application, the ng-controller directive defines the controller.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Angular JS Basics</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-init="cost=5;quantity=5;myColor='orange'">

<**div** ng-controller = "myCtrl">

<**label** for = "firstname"> First Name </**label**><**br**/>

<**input** type = "text" ng-model = "firstname"/><**br**/><**br**/>

<**label** for = "lastname"> Last Name </**label**><**br**/>

<**input** type = "text" ng-model = "lastname"/><**br**/><**br**/>

<**p** style = "color:{{myColor}}">Full Name: {{firstname}} {{lastname}}</**p**>

</**div**>

<**div**>

<**p**> Quantity to be Sold: {{cost \* quantity}}</**p**>

</**div**>

</**div**>

</**body**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl",**function**($scope){

$scope.firstname = "";

$scope.lastname = "";

});

</**script**>

</**html**>

**Code 4: application.html**

## AngularJS Module

An AngularJS module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

Controllers always belong to a module.

### Adding a Directive

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Angular Directive</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" boom-shakalaka>

</**div**>

<**script**>

**var** myApp = angular.module("myApp",[]);

myApp.directive("boomShakalaka", **function**(){

**return** {

template: "I am the boom boom man!"

};

});

</**script**>

</**body**>

</**html**>

**Code 5: directives.html**

### Adding a Controller

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Angular JS Controller</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp">

<**div** ng-controller = "myCtrl">

<**label** for = "first"> First Name</**label**><**br**/>

<**input** type = "text" ng-model = "first"/><**br**/><**br**/>

<**label** for = "second"> Second Name </**label**><**br**/>

<**input** type = "text" ng-model = "second"/><**br**/><**br**/>

<**p**> Full Name: <**span** ng-bind = "first + ' ' + second"></**span**></**p**>

</**div**>

</**div**>

<**script**>

**var** myApp = angular.module("myApp", []);

myApp.controller("myCtrl", **function**($scope){

$scope.first = "";

$scope.second = "";

});

</**script**>

</**body**>

</**html**>

**Code 6: comtroller.html**

### Modules and Controllers in Files

It is common in AngularJS applications to put the module and the controllers in JavaScript files.

In this example, "myApp.js" contains an application module definition, while "myCtrl.js" contains the controller

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Separation </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**label** for = "name"> Name </**label**><**br**/>

<**input** type = "text" ng-model = "name"><**br**/><**br**/>

<**p**>I know what you did there! Mr. <**span** ng-bind = "name"></**span**>!<**p**>

</**div**>

</**body**>

<**script** src = "myApp.js"></**script**>

<**script** src = "myCtrl.js"></**script**>

</**html**>

**Code 7: separation.html (Part 1)**

**var** myApp = angular.module("myApp", []);

**Code 8: myApp.js (Part 2)**

myApp.controller("myCtrl", **function**($scope){

$scope.name = "";

});

**Code 9: myCtrl.js (Part 3)**

**Validate User Input**

**$valid, $dirty, $touched**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> AngularJS Model </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**form** ng-app = "" name = "myForm">

<**label** for = "emailaddress">Email</**label**>

<**input** type = "email" name = "emailaddress" ng-model = "text">

<**p**><**span** ng-show = "myForm.emailaddress.$error.email"> Not a Valid Address </**span**></**p**>

<**h1**> Status</**h1**>

Valid Status: {{myForm.emailaddress.$valid}}<**br**/>

Dirty Status: {{myForm.emailaddress.$dirty}}<**br**/>

Touch Status: {{myForm.emailaddress.$touched}}<**br**/>

</**form**>

<**style**>

**input.ng-invalid** {

**background-color**: red;

}

</**style**>

</**body**>

</**html**>

**Code 10: Validation.html**

The ng-model directive adds/removes the following classes, according to the status of the form field:

**ng-empty, ng-not-empty**

**ng-touched, ng-untouched**

**ng-valid, ng-invalid**

**ng-dirty, ng-pending, ng-pristine**

**Event**

**ng-click**

<!DOCTYPE html>

<html>

<head>

<title> Binding </title>

<meta charset = "UTF-8"/>

<script src = "../Required/angular.js"></script>

</head>

<body>

<div ng-app = "myApp" ng-controller = "myCtrl">

<div ng-click = "changeName()">{{firstname}}</div>

</div>

<script>

var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope){

$scope.firstname = "John";

$scope.changeName = function(){

$scope.firstname = "Bob";

}

});

</script>

</body>

</html>

**Code 11: event.html**

## AngularJS Select

### Repeat

**ng-repeat**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Loop </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**ul**>

<**li** ng-repeat = "x in names">

Name: {{x.name}}; Age: {{x.age}}

</**li**>

</**ul**>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope){

$scope.names = [

{name: 'Sam', age: 15},

{name: 'Bob', age: 28},

{name: 'Mary', age: 41}

];

});

</**script**>

</**body**>

</**html**>

**Code 12: loop.html**

### Options

Dropdowns made with ng-options allows the selected value to be an object, while dropdowns made from ng-repeat has to be a string.

ng-repeat directive has its limitations, the selected value must be a string

ng-options directive, the selected value can be an object.

**ng-options 🡪 ng-options = “x.model for x in cars”**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Repeat Vs Option </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**h1**> Using the ng-repeat method </**h1**>

<**select** ng-model = "selectedCar">

<**option** ng-repeat = "x in cars" value = "{{x.model}}">{{x.model}}</**option**>

</**select**><**br**/>

<**p**> Selected Car is {{selectedCar}}</**p**>

<**h1**> Using the ng-option method </**h1**>

<**select** ng-model = "selectCar" ng-options = "x.model for x in cars"></**select**>

<**p**> Car Model: {{selectCar.model}}</**p**>

<**p**> Color: {{selectCar.color}}</**p**>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope){

$scope.cars = [

{model : "Ford Mustang", color : "red"},

{model : "Fiat 500", color : "white"},

{model : "Volvo XC90", color : "black"}

];

});

</**script**>

</**body**>

</**html**>

**Code 13: options.html**

Using an object as the data source, x represents the key, and y represents the value

**ng-options = “x for (x,y) in cars”**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Options </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**select** ng-model = "selectCar" ng-options = "y.model for (x,y) in cars"></**select**>

<**p**> Car Model: {{selectCar.model}}</**p**>

<**p**> Color: {{selectCar.color}}</**p**>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope){

$scope.cars = [

{model : "Ford Mustang", color : "red"},

{model : "Fiat 500", color : "white"},

{model : "Volvo XC90", color : "black"}

];

});

</**script**>

</**body**>

</**html**>

**Code 14: keyvakye.html**

## AngularJS Scope

The scope is the binding part between the HTML (view) and the JavaScript (controller).

The scope is an object with the available properties and methods.

The scope is available for both the view and the controller.

### Understanding the Scope

If we consider an AngularJS application to consist of:

View, which is the HTML.

Model, which is the data available for the current view.

Controller, which is the JavaScript function that makes/changes/removes/controls the data.

Then the scope is the Model. The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.

$scope in control and the $rootscope cannot have the same variable name;

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Scope </**title**>

<**meta** charset = "UTF-8">

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp">

<**p**>First Name: {{name}} </**p**>

<**div** ng-controller = "myCtrl">

<**p**>Second Name: {{name}}</**p**>

</**div**>

<**p**>Third Name: {{name}} </**p**>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.run(**function**($rootScope){

$rootScope.name = "Bob Marley";

});

app.controller("myCtrl", **function**($scope){

$scope.name = "Jimi Marley";

});

</**script**>

</**body**>

</**html**>

**Code 15: scope.html**

## AngularJS Filters

AngularJS provides filters to transform data:

**currency** 🡪Format a number to a currency format.

**date** 🡪 Format a date to a specified format.

**filter** 🡪 Select a subset of items from an array.

**json** 🡪 Format an object to a JSON string.

**limitTo** 🡪 Limits an array/string, into a specified number of elements/characters.

**lowercase** 🡪 Format a string to lower case.

**number** 🡪 Format a number to a string.

**orderB**y 🡪 Orders an array by an expression.

**uppercase** 🡪 Format a string to upper case.

We can use **app.filter** to make custom filters

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Filter </**title**>

<**meta** charset = "UTF-8">

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**ul**>

<**li** ng-repeat = "x in employees | orderBy: 'age'">

Name: {{x.name}}; Age: {{x.age}}

</**li**>

</**ul**>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope){

$scope.employees = [

{name: "Sam", age: 34},

{name: "Robin", age: 23},

{name: "Tom", age: 67},

];

});

</**script**>

</**body**>

</**html**>

**Code 16: filter.html**

## AngularJS Services

### Location

$location

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

AngularJS has about 30 built-in services. One of them is the **$location** service.

The **$location** service has methods which **return information about the location of the current web page**

**$location.absUrl();**

AngularJS constantly supervises your application, and for it to handle changes and events properly, AngularJS prefers that you use the $location service instead of the window.location object.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Location</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**p**> Url of this page is: {{myUrl}}</p>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope, $location){

$scope.myUrl = $location.absUrl();

});

</**script**>

</**body**>

</**html**>

**Code 17: location.html**

**$location** service is passed in to the controller as an argument. In order to use the service in the controller, it must be defined as a dependency.

### Get

**$http.get**

<?php

$names = [

["name" => "Sam",

"occupation"=> "Engineer"

],

["name" => "Bob",

"occupation"=> "Accountant"

],

["name" => "Mary",

"occupation" => "Choreographer"

]

];

**echo** json\_encode($names);

?>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Location</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**ul**>

<**li** ng-repeat = "x in employees">

Name: {{x.name}}; Occupation: {{x.occupation}}

</**li**>

</**ul**>

</**div**>

</**body**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope, $http){

$http.get("services.php").then(**function**(response){

$scope.employees = response.data;

});

});

</**script**>

</**html**>

**Code 18: services.html**

### Timeout

**$timeout**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Time out </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**p** ng-bind = "sentence"></**p**>

</**div**>

</**body**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope, $timeout){

$scope.sentence = "Hello, my boy!";

$timeout(**function**(){

$scope.sentence = "Good Boy!";

}, 4000);

});

</**script**>

</**html**>

**Code 17: timeout.html**

### Interval

**$interval**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Interval </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**p** ng-bind = "todaydate"></**p**>

</**div**>

</**body**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope, $interval){

$interval(**function**(){

$scope.todaydate = **new** Date().toLocaleTimeString();

}, 1000);

});

</**script**>

</**html**>

**Code 19: interval.html**

### Custom

**app.service(“function\_name”, function(){})**

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>Custom Services</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**p**> {{bro}} </**p**>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.service("hexaconvert", **function**(){

**this**.conversion = **function**(x){

**return** x.toString(16);

};

})

app.controller("myCtrl", **function**($scope, hexaconvert){

$scope.bro = hexaconvert.conversion(1212);

});

</**script**>

</**body**>

</**html**>

**Code 20: custom.html**

## AngularJS Http

**$http 🡪 .delete() .get(), .head() .jsonp(),**

**.patch(), .post() .put()**

Properties

The response from the server is an object with these properties:

**.config** 🡪 the object used to generate the request.

**.data** 🡪 a string, or an object, carrying the response from the server.

**.headers** 🡪 a function to use to get header information.

**.status** 🡪 a number defining the HTTP status.

**.statusText** 🡪 a string defining the HTTP status.

## AngularJS DOM

AngularJS has directives for binding application data to the attributes of HTML DOM elements.

**ng-disabled**

The ng-disabled directive binds AngularJS application data to the disabled attribute of HTML elements.

**ng-show**

The ng-show directive shows or hides an HTML element.

The ng-show directive shows (or hides) an HTML element based on the value of ng-show.

We can use any expression that evaluates to true or false

**ng-hide**

The ng-hide directive hides or shows an HTML element.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>DOM</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = '' ng-init = "mySwitch=true;hour=13">

<**button** ng-disabled = "mySwitch"> Click me! </**button**><**br**/>

<**input** type = "checkbox" ng-model = "mySwitch"/><**br**/>

<**p** ng-bind = "mySwitch"></**p**>

<**p** ng-show = "true"> I can see you!</**p**>

<**p** ng-show = "false"> I can't see you!</**p**>

<**p** ng-show = "hour>12"> Expression Show can see you!</**p**>

<**p** ng-hide = "hour<12"> Expression Hide can see you!</**p**>

<**p** ng-hide = "hour>12"> Expression Hide can see you!</**p**>

</**div**>

</**body**>

</**html**>

**Code 21: dom.html**

## AngularJS Events

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

**ng-copy ng-cut ng-paste**

**ng-keydown ng-keypress ng-keyup**

**ng-dblclick ng-click**

**ng-focus ng-blur ng-change**

**ng-mousedown ng-mouseenter ng-mouseleave**

**ng-mouseover ng-mouseup ng-mousemove**

**ng-click 🡪** Directive will execute when the element is being clicked.

**ng-mouseover 🡪** Directive event changes the HTML element when hovered on

**ng-mouseleave** 🡪 Directive event changes the HTML element when moved away

**$event** Object

We can pass the $event object as an argument when calling the function.

The $event object contains the browser's event object

**ng-mousemove** 🡪 Directive event changes the HTML element when the moue moves

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Event </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**div** ng-mouseover = "myColor='red'" ng-mouseleave = "myColor='black'"

ng-click = "colorChange()">Over me</**div**>

<**p** style = "color: {{myColor}};"> Bro, What is wrong with me!</**p**>

<**button** ng-click = "showGroceries()">Show</**button**>

<**ul** ng-show = "condition">

<**li**> Milk </**li**>

<**li**> Butter</**li**>

<**li**> Bacon </**li**>

</**ul**><**br**/><**br**/>

<**h1** ng-mousemove = "giveCoords($event)"> Check it out </**h1**>

<**p**> ClientX: {{x}}; ClientY {{y}};

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope){

$scope.showGroceries = **function**(){

console.log($scope.condition);

**if**($scope.condition == **undefined** || $scope.condition == **null**){

$scope.condition = **false**;

}**else** **if**($scope.condition){

$scope.condition = **false**;

}**else**{

$scope.condition = **true**;

}

}

$scope.colorChange = **function**(){

$scope.myColor = "blue";

}

$scope.giveCoords = **function**(event){

$scope.x = event.clientX;

$scope.y = event.clientY;

}

});

</**script**>

</**body**>

</**html**>

**Code 22: events.html**

## AngularJS Form

Forms in AngularJS provides data-binding and validation of input controls.

Input controls are the HTML input elements:

input elements, select elements, button elements. textarea elements

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Angular JS Basics</**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**form** ng-app = "myApp" ng-controller = "myCtrl">

<**label** for = "fullname">Name</**label**><**br**/>

<**input** type = "text" ng-model = "fullname"/><**br**/><**br**/>

<**p**>You entered: {{fullname}}</**p**>

<**label** for = "hobbies">Hobbies</**label**><**br**/>

<**input** type = "checkbox" ng-model = "guitar"/>

<**label** for = "guitar">Guitar</**label**>

<**input** type = "checkbox" ng-model = "reading"/>

<**label** for = "reading">Reading</**label**>

<**input** type = "checkbox" ng-model = "gaming"/>

<**label** for = "gaming">Gaming</**label**>

<**p**> You chose: {{guitar}}</**p**>

<**p**> You chose: {{reading}}</**p**>

<**p**> You chose: {{gaming}}</**p**><**br**/>

<**label** for = "gender"> Gender </**label**><**br**/>

<**input** type = "radio" ng-model = "gender" value = "male"/>

<**label** for = "male"> Male </**label**>

<**input** type = "radio" ng-model = "gender" value = "female"/>

<**label** for = "female"> Female </**label**><**br**/>

<**p**> You chose: {{gender}}</**p**>

<**label** for = "cars">Cars</**label**><**br**/>

<**select** ng-options = "x.name for x in cars" ng-model = "car"></**select**>

<**p**> You chose: {{car.name + " " + car.model}}</**p**><**br**/><**br**/>

</**form**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl",**function**($scope){

$scope.fullname = "";

$scope.cars = [

{name: "Toyota", model: "Corolla"},

{name: "Mazda", model: "RX-7"},

{name: "Nissan", model: "350Z"}

];

});

</**script**>

</**body**>

</**html**>

**Code 23: form.html**

## AngularJS 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.

We can use standard HTML5 attributes to validate input, or we can make your own validation functions.

**$untouched** 🡪 The field has not been touched yet

**$touched** 🡪 The field has been touched

**$pristine** 🡪The field has not been modified yet

**$dirty** 🡪 The field has been modified

**$invalid** 🡪The field content is not valid

**$valid** 🡪 The field content is valid

**$pristine** 🡪 No fields have been modified yet

**$dirty** 🡪 One or more have been modified

**$invalid** 🡪 The form content is not valid

**$valid** 🡪 The form content is valid

**$submitted** 🡪 The form is submitted

CSS Classes

AngularJS adds CSS classes to forms and input fields depending on their states.

The following classes are added to, or removed from, input fields:

**ng-untouched** 🡪 The field has not been touched yet

**ng-touched** 🡪 The field has been touched

**ng-pristine** 🡪 The field has not been modified yet

**ng-dirty** 🡪 The field has been modified

**ng-valid** 🡪 The field content is valid

**ng-invalid** 🡪 The field content is not valid

**ng-valid-key** 🡪One key for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated

**ng-invalid-key** 🡪 Example: ng-invalid-required

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Validation </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**form** ng-app = "myApp" ng-controller = "myCtrl" name = "myForm">

<**label** for = "myemail">Email</**label**><**br**/>

<**input** type = "email" ng-model = "myemail" name = "myemail" required/><**br**/>

<**span** ng-show = "myForm.myemail.$touched &&

myForm.myemail.$invalid">The name field is required</**span**>

</**form**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope){

});

</**script**>

<**style**>

**input.ng-valid**{

**background-color**: rgba(0, 255, 0, 0.5);

}

**input.ng-invalid**{

**background-color**: rgba(255, 0, 0, 0.5);

}

</**style**>

</**body**>

</**html**>

**Code 24: validation.html**

### Custom Validation

<!DOCTYPE html>

<**html**>

<**head**>

<**title**></**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**form** ng-app = "myApp" name = "myForm">

<**label** for = "regex"> Phone Number </**label**>

<**input** ng-model = "regex" name = "regex" required my-directive/>

<**p**> Valid: {{myForm.regex.$valid}} </**p**>

<**p** ng-show = "myForm.regex.$invalid">Pattern do no match [XXX-XXX-XXXX]</**p**>

</**form**>

<**script**>

**var** app = angular.module("myApp", []);

app.directive("myDirective", **function**(){

**return** {

require: "ngModel",

link: **function**(scope, element, attr, mCtrl){

**function** myValidation(value){

**var** pattern = /^\d{3}-\d{3}-\d{4}$/;

**if**(pattern.test(value)){

console.log(pattern.test(value));

mCtrl.$setValidity('charE', **true**);

}**else**{

mCtrl.$setValidity('charE', **false**);

}

**return** value;

}

mCtrl.$parsers.push(myValidation);

}

};

});

</**script**>

</**body**>

</**html**>

**Code 24: customvalidation.html**

## AngularJS Global API

The AngularJS Global API is a set of global JavaScript functions for performing common tasks.

API stands for Application Programming Interface.

Comparing objects, iterating objects, Converting data

The Global API functions are accessed using the angular object.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Global API </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body** >

<**div** ng-app = "myApp" ng-controller = "myCtrl">

Input: <**input** type = "text" ng-model = "inputNumber"/><**br**/>

<**p**>Display: {{inputNumber}}</**p**>

<**p**>Lower: {{lower}}</**p**>

<**p**>Upper: {{upper}}</**p**>

<**p**>String: {{isitstring}}</**p**>

<**p**>number: {{isitnumber}}</**p**>

</**div**>

</**body**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope, $interval){

$scope.inputNumber = "";

$interval(**function**(){

$scope.lower = angular.lowercase($scope.inputNumber);

$scope.upper = angular.uppercase($scope.inputNumber);

$scope.isitstring = angular.isString($scope.inputNumber);

$scope.isitnumber = angular.isNumber($scope.inputNumber);

},1000);

});

</**script**>

</**html**>

**Code 24: globalapi.html**

## AngularJS Include

<?php

$names = [

["name" => "Sam",

"occupation"=> "Engineer"

],

["name" => "Bob",

"occupation"=> "Accountant"

],

["name" => "Mary",

"occupation" => "Choreographer"

]

];

**echo** json\_encode($names);

?>

<**table** class = "w3-table-all">

<**tr**>

<**th**>Name</**th**>

<**th**>Occupation</**th**>

</**tr**>

<**tr** ng-repeat = "x in employees">

<**td**>{{x.name}}</**td**>

<**td**>{{x.occupation}}</**td**>

</**tr**>

</**table**>

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Including a File </**title**>

<**meta** charset = "UTF-8"/>

<**link** rel = "stylesheet" href = "../Required/w3.css"/>

<**script** src = "../Required/angular.js"></**script**>

</**head**>

<**body**>

<**div** ng-app = "myApp" ng-controller = "myCtrl">

<**div** ng-include = "'include.html'"></**div**>

</**div**>

<**script**>

**var** app = angular.module("myApp", []);

app.controller("myCtrl", **function**($scope, $http){

$http.get("services.php").then(**function**(response){

$scope.employees = response.data;

});

});

</**script**>

</**body**>

</**html**>

**Code 25: include.html**

### Include Cross Domains

By default, the ng-include directive does not allow you to include files from other domains.

To include files from another domain, we can add a whitelist of legal files and/or domains in the config function of our application.

<body ng-app="myApp">  
<div ng-include="'https://tryit.w3schools.com/angular\_include.php'"></div>  
<script>

var app = angular.module('myApp', [])  
app.config(function($sceDelegateProvider) {  
    $sceDelegateProvider.resourceUrlWhitelist([  
        'https://tryit.w3schools.com/\*\*'  
    ]);  
});

</script>  
  
</body>

**Code 26: crossdomain.html**

## AngularJS Animation

An animation is when the transformation of an HTML element gives you an illusion of motion.

First include the library and then refer to the **ngAnimate** module in your application

<body ng-app = "**ngAnimate**"> or if our application has a name, add ngAnimate as a dependency in your application module.

<script>

var app = angular.module('myApp', ['ngAnimate']);

</script>

### ngAnimate

The ngAnimate module adds and removes classes.

When ngAnimate notice certain events, like hide or show of an HTML element, the element gets some pre-defined classes which can be used to make animations.

The directives in AngularJS who add/remove classes are:

**ng-show, ng-hide**

The ng-show and ng-hide directives adds or removes a ng-hide class value.

**ng-class, ng-view. ng-include. ng-repeat. ng-if. ng-switch**

The other directives add a ng-enter class value when they enter the DOM, and a ng-leave attribute when they are removed from the DOM.

The ng-repeat directive also adds a ng-move class value when the HTML element changes position.

In addition, during the animation, the HTML element will have a set of class values, which will be removed when the animation has finished. Example: the ng-hide directive will add these class values:

**ng-animate**

**ng-hide-animate**

**ng-hide-add (if the element will be hidden)**

**ng-hide-remove (if the element will be showed)**

**ng-hide-add-active (if the element will be hidden)**

**ng-hide-remove-active (if the element will be showed)**

### CSS Animations

CSS Animations allows us to change CSS property values smoothly, from one value to another, over a given duration.

@keyframes myChange {  
    from {  
        height: 100px;  
    } to {  
        height: 0;  
    }  
}

.ng-hide{

animation: 0.5s myChange;

height: 0px;

}

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Global API </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

<**script** src = "../Required/angular-animate.js"></**script**>

</**head**>

<**body**>

<**diV** ng-app = "myApp" ng-controller = "myCtrl">

<**button** ng-click = "toggle()">Animation</**button**>

<**div** ng-show = "check"></**div**>

</**diV**>

</**body**>

<**script**>

**var** app = angular.module("myApp",["ngAnimate"]);

app.controller("myCtrl", **function**($scope){

$scope.check = **false**;

$scope.toggle = **function**(){

**if**($scope.check){

$scope.check = **false**;

}**else**{

$scope.check = **true**;

}

};

});

</**script**>

<**style**>

**.ng-hide**{

**height**: 0px;

}

**div** > **div**{

**height**: 100px;

**background-color**: lightblue;

transition: all linear 0.5s;

}

</**style**>

</**html**>

**Code 27: animation.html**

## AngularJS Routing

The **ngRoute** module helps your application to become a Single Page Application.

If we want to navigate to different pages in our application, but want the application to be a SPA (Single Page Application), with no page reloading, we can use the **ngRoute** module.

The **ngRoute** module routes our application to different pages without reloading the entire application.

We must add the ngRoute as a dependency in the application module:

var app = angular.module("myApp", ["ngRoute"]);

Now our application has access to the route module, which provides the $routeProvider.

Use the $routeProvider to configure different routes in your application:

Our application needs a container to put the content provided by the routing.

This container is the ng-view directive.

There are three different ways to include the ng-view directive in your application

<div ng-view></div>

<ng-view></ng-view>

<div class="ng-view"></div>

Applications can only have one ng-view directive, and this will be the placeholder for all views provided by the route.

<!DOCTYPE html>

<**html**>

<**head**>

<**title**> Routing </**title**>

<**meta** charset = "UTF-8"/>

<**script** src = "../Required/angular.js"></**script**>

<**script** src = "../Required/angular-route.js"></**script**>

</**head**>

<**body** ng-app = "myApp">

<**p**><**a** href="#/!">Main</**a**></**p**>

<**a** href="#!london">City 1</**a**>

<**a** href="#!paris">City 2</**a**>

<**div** ng-view></**div**>

</**body**>

<**script**>

**var** app = angular.module("myApp", ["ngRoute"]);

app.config(**function**($routeProvider) {

$routeProvider

.when("/", {

templateUrl : "main.html"

})

.when("/london", {

templateUrl : "london.html",

controller : "londonCtrl"

})

.when("/paris", {

templateUrl : "paris.html",

controller : "parisCtrl"

});

});

app.controller("londonCtrl", **function** ($scope) {

$scope.msg = "I love London";

});

app.controller("parisCtrl", **function** ($scope) {

$scope.msg = "I love Paris";

});

</**script**>

</**html**>

**Code 28: routing.html**

<**h1**>London</**h1**>

<**h3**>London is the capital city of England.</**h3**>

<**p**>It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</**p**>

<**p**>{{msg}}</**p**>

**Code 29: london.html**

<**h1**>Paris</**h1**>

<**h3**>Paris is the capital city of France.</**h3**>

<**p**>The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants.</**p**>

<**p**>{{msg}}</**p**>

**Code 30: paris.html**

## Angular Keywords

**ng-app** 🡪 Directive defines an AngularJS application

**ng-model** 🡪 Directive binds the value of HTML controls (input, select) to application data

**ng-bind** 🡪 Directive binds application data to the HTML view

**ng-init** 🡪 Directive initializes AngularJS application variables

**data-ng-init** 🡪 Directive which is similar to ng-init, if we want to make the HTML valid

**data-ng-bind** 🡪 Directive similar to ng-bind

**{{expression}} 🡪** Expressions

**ng-controller** 🡪 Directive defines the controller.

**ng-click 🡪** Directive will execute when the element is being clicked.

**ng-repeat 🡪** Directive loops through the array

**ng-options 🡪** Directive specialized for select options (Give data in object)

**ng-disabled 🡪** Directive disables the HTML element

**ng-hide 🡪** Directive hides or shows an HTML element

**ng-show 🡪** Directive hides or shows an HTML element

**ng-mouseover 🡪** Directive event changes the HTML element when hovered on

**ng-mouseleave** 🡪 Directive event changes the HTML element when moved away

**ng-mousemove** 🡪 Directive event changes the HTML element when the moue moves

**ng-include** 🡪 Directive to include HTML element

**Functions**

**$untouched** 🡪 The field has not been touched yet

**$touched** 🡪 The field has been touched

**$pristine** 🡪The field has not been modified yet

**$dirty** 🡪 The field has been modified

**$invalid** 🡪The field content is not valid

**$valid** 🡪 The field content is valid

**$pristine** 🡪 No fields have been modified yet

**$dirty** 🡪 One or more have been modified

**$invalid** 🡪 The form content is not valid

**$valid** 🡪 The form content is valid

**$submitted** 🡪 The form is submitted

**$http 🡪** Returns response from the server

**$location 🡪** Returns information about the location of the current web page

**$timeout 🡪** Returnsfunction after a certain time

**$interval 🡪** Returns function after a certain interval of time

**$sceDelegateProvider 🡪** Returns cross domain html content

**CSS Classes**

**ng-untouched** 🡪 The field has not been touched yet

**ng-touched** 🡪 The field has been touched

**ng-pristine** 🡪 The field has not been modified yet

**ng-dirty** 🡪 The field has been modified

**ng-valid** 🡪 The field content is valid

**ng-invalid** 🡪 The field content is not valid

**ng-valid-key** 🡪One key for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated

**ng-invalid-key** 🡪 Example: ng-invalid-required

**CSS Animation Classes**

**ng-animate**

**ng-hide-animate**

**ng-hide-add (if the element will be hidden)**

**ng-hide-remove (if the element will be showed)**

**ng-hide-add-active (if the element will be hidden)**

**ng-hide-remove-active (if the element will be showed)**

**Global API**

**angular.lowercase()**🡪 Converts a string to lowercase

**angular.uppercase()**🡪 Converts a string to uppercase

**angular.isString()** 🡪 Returns true if the reference is a string

**angular.isNumber()** 🡪 Returns true if the reference is a number

**APP**

**app.directive (Example)**

var app = angular.module("myApp", []);

app.directive("myDirective", function(){

return{

require: "ngModel",

link: function(scope, element, attr, mCtrl){

function myValidation(value){

var pattern = /^\d{3}-\d{3}-\d{4}$/;

if(pattern.test(value)){

console.log(pattern.test(value));

mCtrl.$setValidity('charE', true);

}else{

mCtrl.$setValidity('charE', false);

}

return value;

}

mCtrl.$parsers.push(myValidation);

}

};

});

**app.config (Example)**

**var** app = angular.module("myApp", ["ngRoute"]);

app.config(**function**($routeProvider) {

$routeProvider

.when("/", {

templateUrl : "main.html"

})

.when("/london", {

templateUrl : "london.html",

controller : "londonCtrl"

})

.when("/paris", {

templateUrl : "paris.html",

controller : "parisCtrl"

});

});

app.controller("londonCtrl", **function** ($scope) {

$scope.msg = "I love London";

});

app.controller("parisCtrl", **function** ($scope) {

$scope.msg = "I love Paris";

});