Unit 6

Menus, Navigation and Web Page Protection

Marks: 14 (R-2, U-4, A-6)

Course Outcome: Create menus and navigations in web pages.

Unit Outcome:

1) Develop Javascript to manage the given status bar.
2) Develop javascript to create the given banner.
3) Develop javascript to create the given slideshow.
4) Develop javascript to create the given menu.
5) Develop javascript to protect a webpage in a specified manner.

Topics and Sub-topics:

6.1 Status bar-builds a static message, changing the message using rollover, moving the message using rollover
6.2 Banner-loading and displaying banner advertisement. Linking a banner advisement to url
6.3 Slide show – creating a slideshow
6.4 Menus-creating a pulldown menu, dynamically changing a menu, validating a menu selection, Floating menu, chain select menu, Tab menu, Pop-up menu, sliding menu, Highlighted menu, Folding a tree menu, context menu, scrollable menu, side bar menu
6.5 Protective web page-Hiding your code, disabling the right mouse button, javascript, concealing e-mail address

6.6 Frameworks of javascript and its application.

6.1. Status Bar

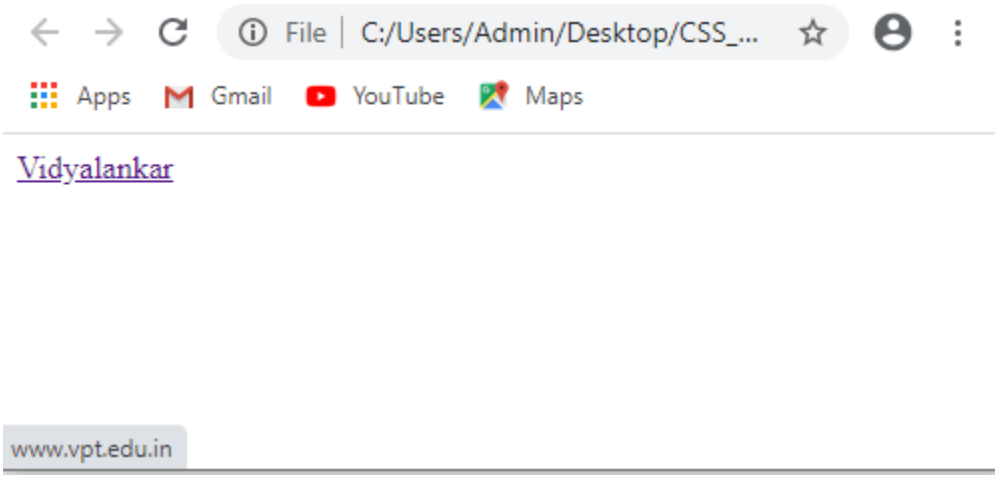
The **status** property of the Window interface was originally intended to set the text in the status bar at the bottom of the browser window. However, the HTML standard now requires setting window.status to have no effect on the text displayed in the status bar.

Syntax:

```
window.status = string;
var value = window.status;
```

```
 <html>
<head>
<title>JavaScript Status Bar</title></head>
<body>
  <a href="http://www.vpt.edu.in"
   onMouseOver="window.status='Vidyalankar';return true"
   onMouseOut="window.status='';return true">
Vidyalankar
  </a>
</body>
</html>
```

Output:



Note: "window.status" does not supported any browser.

6.2 Banner

The banner advertisement is the hallmark of every commercial web page. It is typically positioned near the top of the web page, and its purpose is to get the visitor's attention by doing all sorts of clever things.
Nearly all banner advertisements are in a file format such as a GIF, JPG, TIFF, or other common graphic file format. Some are animated GIFs, which is a series of images contained in one file that rotate automatically on the screen. Some are Flash movies that require the visitor to have a browser that includes a Flash plug-in. Many banner advertisements consist of a single graphical image that does not contain any animation and does not require any special plug-in.
You need to do three things to incorporate a banner advertisement in your web page:
1. Create several banner advertisements using a graphics tool such as Photoshop. You'll want to make more than one advertisement so you can rotate them on your web page using a JavaScript.
2. Create an <img> element in your web page with the height and width necessary to display the banner advertisement.
3. Build a JavaScript that loads and displays the banner advertisements in conjunction with the <img> element.

6.2.1 Loading and Displaying Banner Advertisements

The banners should all be the same size so they look professional as they rotate on your web page. The best way to do this is to create an empty banner and then copy it for each banner advertisement that you want to build. This assures that all the banners will be the same size.
You can then use each copy to design each ad. Next, create an image element on your web page using the <img> tag. You'll need to set four attributes of the <img> tag: src, width, height, and name. Set
the src attribute to the file name of the first banner advertisement that you want to display. Set the width and height attributes to the width and height of the banner. Set the name attribute to a unique name for the image element. You'll be using the name attribute in the JavaScript when you change from one banner to the next. The image element (banner) should be centered in the page using the tag within the <body> tag of your web page, The final step is to build the JavaScript that will rotate the banners on your web page. You'll define the JavaScript in the <head> tag of the web page.

The JavaScript must do the following to load the banner:
1. Load banner advertisements into an array.
2. Determine whether the browser supports the image object.

3. Display a banner advertisement.
4. Pause before displaying the next banner advertisement.

You load the banner advertisements into an array by declaring an Array() object and initializing it with the file name of each banner advertisement. For example, suppose you have three banner advertisements that are contained in the 1.jpg, 2.jpg, and 3.jpg files. Here's how you'd load them into an Array() object:

Banners = new Array('1.jpg','2.jpg','3.jpg')

Example:

```
<html >
<head>
<title>Banner Ads</title>
<script>
Banners = new Array('1.jpg','2.jpg','3.jpg');
CurrentBanner = 0;
function DisplayBanners()
{
if (document.images);
 {
CurrentBanner++;
if (CurrentBanner == Banners.length)
{
CurrentBanner = 0;
}
document.RotateBanner.src= Banners[CurrentBanner];
setTimeout("DisplayBanners()",1000);
}
}
</script>
</head>
<body onload="DisplayBanners()" >
<center>
<img src="1.jpg" width="400"
```

```
height="75" name="RotateBanner" />
</center>
</body>
</html>
```

6.2.2 Linking Banner Advertisements to URLs

A banner advertisement is designed to encourage the visitor to learn more information about a product or service that is being advertised. To get additional information, the visitor is expected to click the banner so that a new web page opens. You can link a banner advertisement to a web page by inserting a hyperlink into your web page that calls a JavaScript function rather than the URL of a web page. The JavaScript then determines the URL that is associated with the current banner and loads the web page that is associated with the URL.

```
<html>
<head>
<title>Link Banner Ads</title>
<script language="Javascript" type="text/javascript">

Banners = new Array('1.jpg','2.jpg','3.jpg')
BannerLink = new Array(
'google.com/','vpt.edu.in/', 'msbte.org.in/');
CurrentBanner = 0;
NumOfBanners = Banners.length;
function LinkBanner()
{
document.location.href =
"http://www." + BannerLink[CurrentBanner];
}
function DisplayBanners() {
if (document.images) {
CurrentBanner++
if (CurrentBanner == NumOfBanners) {
CurrentBanner = 0
}
```

```
document.RotateBanner.src= Banners[CurrentBanner]
setTimeout("DisplayBanners()",1000)
}
}
</script>
</head>
<body onload="DisplayBanners()" >
<center>
<a href="javascript: LinkBanner()"><img src="1.jpg"
width="400" height="75" name="RotateBanner" /></a>
</center>
</body>
</html>
```

6.3 Slideshow:

A slideshow is similar in concept to a banner advertisement in that a slideshow rotates multiple images on the web page. However, unlike a banner advertisement, a slideshow gives the visitor the ability to change the image that's displayed: the visitor can click the Forward button to see the next image and the Back button.

Creating a slideshow:

First, set the **slideIndex** to 1. (First picture)
Then call **showDivs()** to display the first image.
When the user clicks one of the buttons call **plusDivs()**.
The plusDivs() function **subtracts** one or **adds** one to the slideIndex.
The **showDiv()** function hides (**display="none"**) all elements with the class name "mySlides", and displays (**display="block"**) the element with the given slideIndex.
If the slideIndex is **higher than** the number of elements (x.length), the slideIndex is set to zero.
If the slideIndex is **less than** 1 it is set to number of elements (x.length).

```
<html>
<title>slideshow</title>
<body>
<h2 class="w3-center">Manual Slideshow</h2>
<div class="w3">
```

```
  <img class="mySlides" src="y.jpg" style="width:50%">
  <img class="mySlides" src="yy.jpg" style="width:50%">
  <img class="mySlides" src="yyy.jpg" style="width:50%">
  <img class="mySlides" src="yyyy.jpg" style="width:50%">

  <button class="aa" onclick="plusDivs(-1)">&#10094;Back</button>
  <button class="bb" onclick="plusDivs(1)">&#10095;Forward</button>
</div>

<script>
var slideIndex = 1;
showDivs(slideIndex);

function plusDivs(n)
{
  showDivs(slideIndex += n);
}

function showDivs(n)
 {
  var i;
  var x = document.getElementsByClassName("mySlides");
  if (n > x.length)
     {
     slideIndex = 1
     }

  if (n < 1)
     {
     slideIndex = x.length
     }
  for (i = 0; i < x.length; i++)
     {
     x[i].style.display = "none";
```

```
    }
  x[slideIndex-1].style.display = "block";
}
</script>
</body>
</html>
```

Automatic Slideshow:

```
<html>
<head>
<title>Automatic Slideshow</title>
</head>
<body>
<h2 class="aa">Automatic Slideshow</h2>
<div class="aa" style="max-width:500px">
  <img class="mySlides" src="y.jpg" style="width:200%">
  <img class="mySlides" src="yy.jpg" style="width:200%">
  <img class="mySlides" src="yyy.jpg" style="width:200%">
  <img class="mySlides" src="yyyy.jpg" style="width:200%">
</div>
<script>
var myIndex = 0;
auto_slide_show();

function auto_slide_show()
{
  var i;
  var x = document.getElementsByClassName("mySlides");
  for (i = 0; i < x.length; i++) {
    x[i].style.display = "none";
  }
  myIndex++;
  if (myIndex > x.length) {myIndex = 1}
```

```
  x[myIndex-1].style.display = "block";
  setTimeout(auto_slide_show, 2000); // Change image every 2 seconds
}
</script>
</body>
</html>
```

6.4 Menus:

A menu consists of a set of options which are presented to the user. Menus are common in graphical user interface such as Windows or Mac OS.

The Menu component provides the pull-down menu element that's common in most graphical user interfaces (GUIs). Using a familiar GUI element will reduce the interface learning curve of your web site or application for new users, as well as help all users more easily find what they're looking for. Having menus that contain links to sections at various levels in your web site can improve both the navigation of the site and the real estate of your web pages.

6.4.1 Creating pull-down menu:

- Also known as drop-down menus.
- Clicking a menu title causes the menu items to appear to drop down from that position and be displayed.

Code: Create a simple menu: If user not selected any menu, alert should be displayed.

```
Select Fruit:
<select id="ddlFruits">
   <option value=""></option>
   <option value="1">Apple</option>
   <option value="2">Mango</option>
   <option value="3">Orange</option>
</select>
<input type="submit" value="Validate" onclick="return Validate()" />
<script type="text/javascript">
   function Validate()
{
      var ddlFruits = document.getElementById("ddlFruits");
```

```
    if (ddlFruits.value == "") {
        //If the "Please Select" option is selected display error.
        alert("Please select an option!");
        return false;
    }
    return true;
  }
</script>
```

Code: Design a drop-down menu for various colours. After selecting any colour from menu, background colour should be changed.

1) demo2.html
2) demo2-script.js
3) demo2-style.css

HTML file:

```
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Demo 2</title>
  <link rel="stylesheet" href="demo2-style.css">
</head>
<body>
<div class="container">
  <label for="color">Choose a Background Color:</label>
  <select name="color" id="color" class="color" onchange="changeColor()">
    <option value="white">White</option>
    <option value="black">Black</option>
    <option value="blue">Blue</option>
    <option value="red">Red</option>
    <option value="yellow">Yellow</option>
  </select>
</div>
<script type="text/javascript" src="demo2-script.js"></script>
</body>
```

```
</html>
```

JavaScript file:

```
changeColor = () => {
  var color = document.getElementById("color").value;
  switch(color){
    case "white":
      document.body.style.backgroundColor = "white";
      document.body.style.color = "black";
      document.getElementById("color").style.backgroundColor = "white";
      document.getElementById("color").style.color = "black";
      break;
    case "black":
      document.body.style.backgroundColor = "black";
      document.body.style.color = "white";
      document.getElementById("color").style.backgroundColor = "black";
      document.getElementById("color").style.color = "white";
      break;
    case "blue":
      document.body.style.backgroundColor = "blue";
      document.body.style.color = "white";
      document.getElementById("color").style.backgroundColor = "blue";
      document.getElementById("color").style.color = "white";
      break;
    case "red":
      document.body.style.backgroundColor = "red";
      document.body.style.color = "white";
      document.getElementById("color").style.backgroundColor = "red";
      document.getElementById("color").style.color = "white";
      break;
    case "yellow":
      document.body.style.backgroundColor = "yellow";
      document.body.style.color = "black";
      document.getElementById("color").style.backgroundColor = "yellow";
```

```
            document.getElementById("color").style.color = "black";
            break;
        default:
            document.body.style.backgroundColor = "white";
            document.body.style.color = "black";
            document.getElementById("color").style.backgroundColor = "white";
            document.getElementById("color").style.color = "black";
            break;
    }
}
```

CSS file:

```
* {
    box-sizing: border-box;
}
body {
    font-family: "Calibri", "Roboto", sans-serif;
    -ms-overflow-style: none;  /* IE and Edge */
    scrollbar-width: none;  /* Firefox */
}
body::-webkit-scrollbar {
    display: none;
}
.container{
    margin: 10%;
    text-align: center;
}
.color{
    width: 30%;
    outline: none;
    height: 30px;
    background: transparent;
}
```

6.4.2 Dynamically changing a Menu:

Code: Following example provides two radio buttons to the user one is for fruits and another is for vegetables.

When user will select the fruits radio button, the option list should present only the fruits names to user and when user will select the vegetable radio button, the option list should present only the vegetable names to user so that user can select an appropriate element of interest.

```
<html>
<body>
<html>
<script type="text/javascript">
function modifyList(x)
{
with(document.forms.myform)
{
if(x ==1)
{
optionList[0].text="Kiwi";
optionList[0].value=1;
optionList[1].text="Pine-Apple ";
optionList[1].value=2;
optionList[2].text="Apple";
optionList[2].value=3;
}

if(x ==2)
{
optionList[0].text="Tomato";
optionList[0].value=1;
optionList[1].text="Onion ";
optionList[1].value=2;
optionList[2].text="Cabbage ";
optionList[2].value=3;
```

```
}
}
}

</script>
</head>
</body>
<form name="myform" action=" " method="post">
<select name="optionList" size="3">
<option value=1>Kiwi
<option value=1>Pine-Apple
<option value=1>Apple
</select>
<br>
<input type="radio" name="grp1" value=1 checked="true"
onclick="modifyList(this.value)"> Fruits

<input type="radio" name="grp1" value=2 onclick="modifyList(this.value)">
Vegitables
</form>
</body>
</html>
```

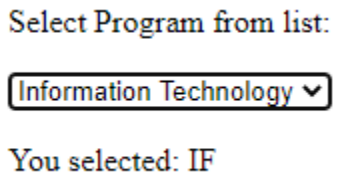
Output:

6.4.3 Validating Menu Selections:

Code: Following example provides four list elements as name of branches. When you select a branch from list, selected branch will be displayed as output.

```
<html>
<body>
<p>Select Program from list:</p>
<select id="mySelect" onchange="myFunction()">
  <option value="CO">Computer Engg</option>
  <option value="IF">Information Technology</option>
  <option value="EJ">Electronics and Tele</option>
  <option value="CE">Chemical Engg</option>
</select>
<p id="demo"></p>
<script>
function myFunction()
{
  var x = document.getElementById("mySelect").value;
  document.getElementById("demo").innerHTML = "You selected: " + x;
}
</script>
</body>
</html>
```

Output:

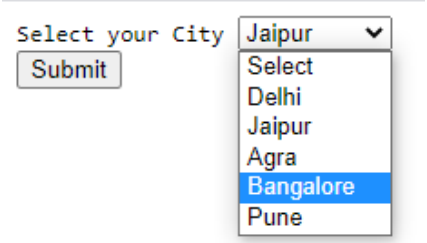


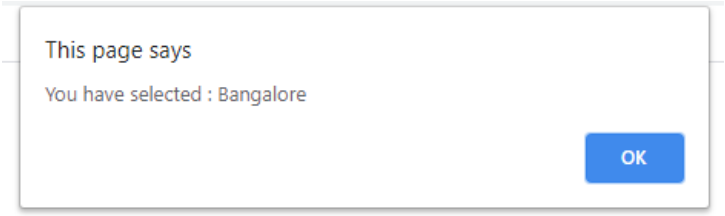
Code:

```
<html>
<script language="Javascript">
function validate()
```

```
{
if(document.form.city.selectedIndex=="")
{
alert ( "Please select city!");
return false;
}
var sel = document.getElementById("city");
//get the selected option
var selectedText = sel.options[sel.selectedIndex].text;
alert("You have selected : "+selectedText);
return true;
}
</script>
<form name="form" method="post" onSubmit="return validate()"><pre>
Select your City <select name="city" id="city">
<option value="Select">Select</option>
<option value="Delhi">Delhi</option>
<option value="Jaipur">Jaipur</option>
<option value="Agra">Agra</option>
<option value="Bangalore">Bangalore</option>
<option value="Pune">Pune</option>
</select>
<input type="submit" name="Submit" value="Submit">
</pre></form>
</html>
```

Output:





6.4.4. Floating Menu

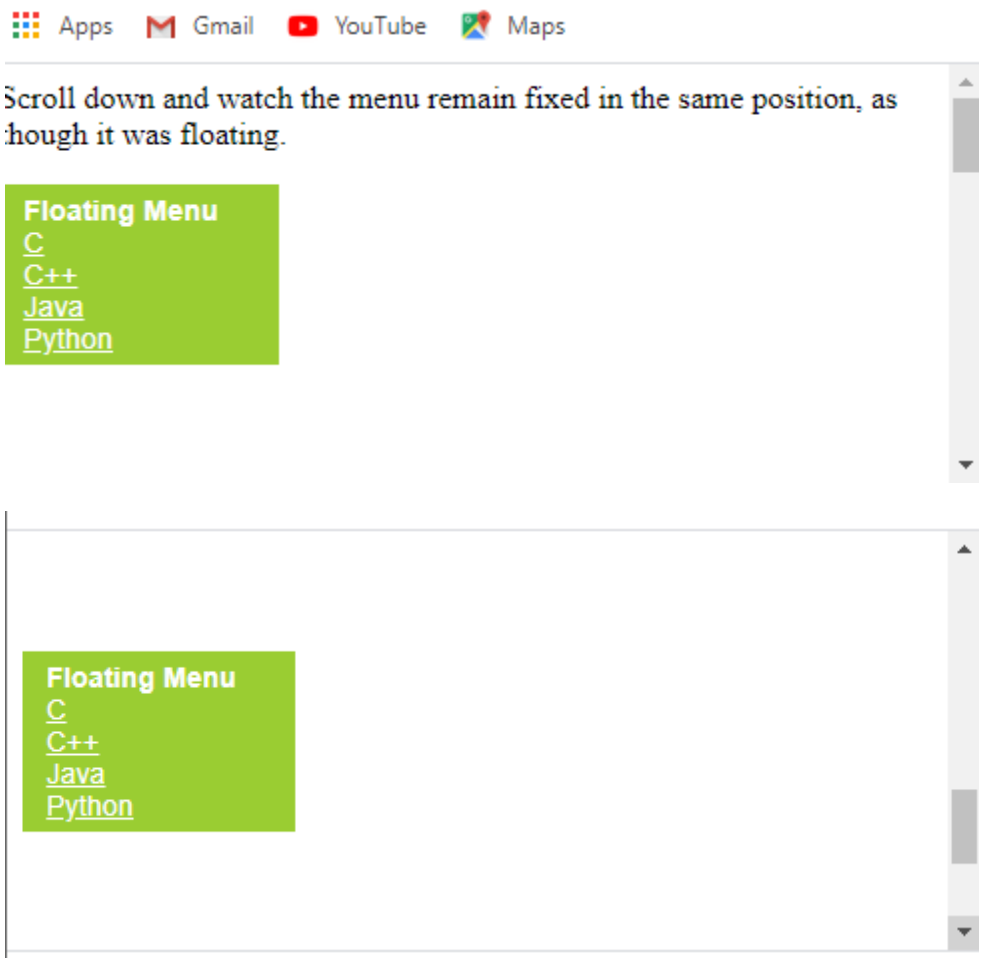
Also known as "fixed menus" and "hovering menus", floating menus stay in a fixed position when you scroll the page. They appear to "float" on top of the page as you scroll.

Code:

```
<html>
<title>Example</title>
<style>
 body {
  background-image: url('/pix/samples/bg1.gif');
 }
 main {
  margin-bottom: 200%;
 }
 .floating-menu {
  font-family: sans-serif;
  background: yellowgreen;
  padding: 5px;;
  width: 130px;
  z-index: 100;
  position: fixed;
 }
 .floating-menu a,
 .floating-menu h3 {
  font-size: 0.9em;
  display: block;
  margin: 0 0.5em;
  color: white;
 }
</style>
<main>
 <p>Scroll down and watch the menu remain fixed in the same position, as though it was floating.</p>
```

```
  <nav class="floating-menu">
    <h3>Floating Menu</h3>
    <a href="c_sub.txt">C</a>
    <a href="C++_sub.txt">C++</a>
    <a href="java_sub.txt">Java</a>
    <a href="python_sub.txt">Python</a>
  </nav>
</main>
```

Output: scroll down the page to observe the output:



6.4.5. Chain Select Menu

Chained selects menu lets you "chain" multiple form selects list together so that the selection in a "parent" list can tailor the options available in a "child" list.

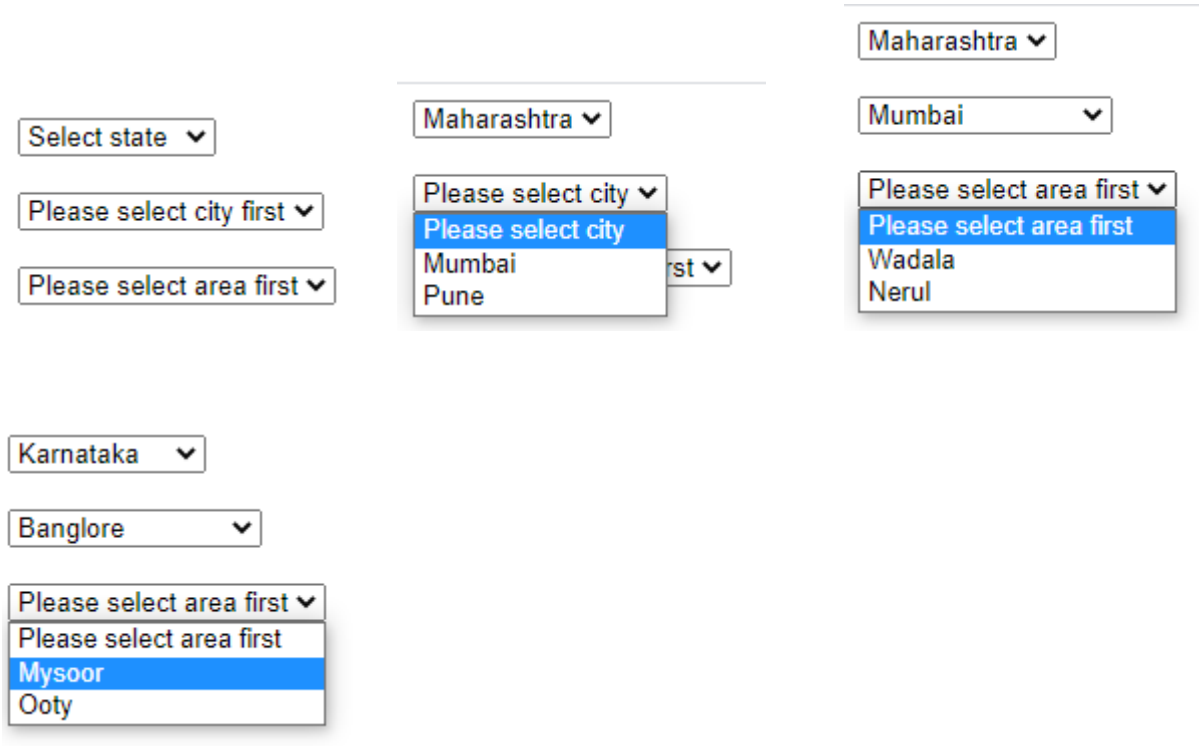
Code:

```
<html>
<head><title>chained menu</title></head>
<script>
var stateObject = {
   "Maharashtra": {
      "Mumbai": ["Wadala", "Nerul"],
      "Pune": ["Aundh","Kothrud"]
   },
   "Karnataka": {
      "Banglore": ["Mysoor", "Ooty"],
   }
}
window.onload = function ()
 {
   var aaa = document.getElementById("aaa"),
      bbb = document.getElementById("bbb"),
      ccc = document.getElementById("ccc");
   for (var state in stateObject)
   {
      aaa.options[aaa.options.length] = new Option(state, state);
   }
   aaa.onchange = function ()
     {
      bbb.length = 1; // remove all options bar first
      ccc.length = 1; // remove all options bar first
      if (this.selectedIndex < 1) {
        bbb.options[0].text = "Please select city first"
        ccc.options[0].text = "Please select area first"
        return; // done
      }
     bbb.options[0].text = "Please select city"
      for (var citi_name in stateObject[this.value]) {
```

```
          bbb.options[bbb.options.length] = new Option(citi_name, citi_name);
       }
       if (bbb.options.length==2)
       {
         bbb.selectedIndex=1;
         bbb.onchange();
       }
        }
   aaa.onchange(); // reset in case page is reloaded
  bbb.onchange = function ()
     {
      ccc.length = 1; // remove all options bar first
       if (this.selectedIndex < 1)
       {
        ccc.options[0].text = "Please select area first"
         return; // done
       }
       ccc.options[0].text = "Please select area first"
        var cities = stateObject[aaa.value][this.value];
       for (var i = 0; i < cities.length; i++) {
          ccc.options[ccc.options.length] = new Option(cities[i], cities[i]);
       }
       if (ccc.options.length==2)
       {
         ccc.selectedIndex=1;
         ccc.onchange();
       }
          }
}
</script>
</body>
<form name="myform" id="myForm">
   <select name="optone" id="aaa" size="1">
      <option value="" selected="selected">Select state</option>
```

```
    </select>
    <br>
    <br>
    <select name="opttwo" id="bbb" size="1">
        <option value="" selected="selected">Please select city first</option>
    </select>
    <br>
    <br>
    <select name="optthree" id="ccc" size="1">
        <option value="" selected="selected">Please select area first</option>
    </select>
</form>
</body>
</html>
```

Output:



6.4.6 Tab Menu

Using tab menu, more complete description is displayed below the tab bar as the visitor clicks the mouse cursor over the tabs.

2-ways to create tab menu:

a) Using button
b) Using target selector

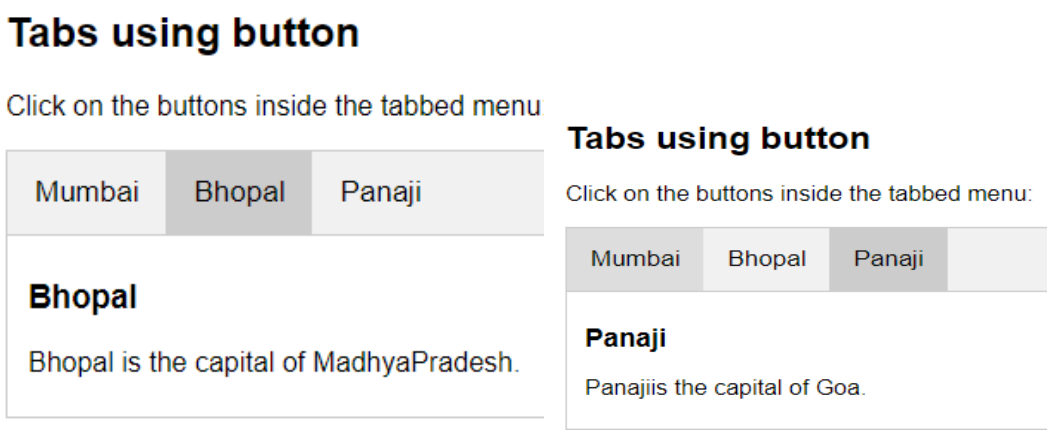
Code: In following example, created 3 buttons using <button>

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial;}

/* Style the tab */
.tab {
  overflow: hidden;
  border: 1px solid #ccc;
  background-color: #f1f1f1;
}

/* Style the buttons inside the tab */
.tab button {
  background-color: inherit;
  float: left;
  border: none;
  outline: none;
  cursor: pointer;
  padding: 14px 16px;
  transition: 0.3s;
  font-size: 17px;
}
```

```
/* Change background color of buttons on hover */
.tab button:hover {
  background-color: #ddd;
}

/* Create an active/current tablink class */
.tab button.active {
  background-color: #ccc;
}

/* Style the tab content */
.tabcontent {
  display: none;
  padding: 6px 12px;
  border: 1px solid #ccc;
  border-top: none;
}
</style>
</head>
<body>

<h2>Tabs using button </h2>
<p>Click on the buttons inside the tabbed menu:</p>

<div class="tab">
  <button class="tablinks" onclick="openCity(event, 'Mumbai')">Mumbai</button>
  <button class="tablinks" onclick="openCity(event, 'Bhopal')">Bhopal</button>
  <button class="tablinks" onclick="openCity(event, 'Panaji')">Panaji</button>
</div>

<div id="Mumbai" class="tabcontent">
  <h3>Mumbai</h3>
  <p>Mumbai is the capital city of Maharashtra.</p>
```

```
</div>

<div id="Bhopal" class="tabcontent">
  <h3>Bhopal</h3>
  <p>Bhopal is the capital of MadhyaPradesh.</p>
</div>

<div id="Panaji" class="tabcontent">
  <h3>Panaji</h3>
  <p>Panajiis the capital of Goa.</p>
</div>
<script>
function openCity(evt, cityName)
{
  var i, tabcontent, tablinks;
  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++)
  {
    tabcontent[i].style.display = "none";
  }
  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++)
  {
    tablinks[i].className = tablinks[i].className.replace(" active", "");
  }
  document.getElementById(cityName).style.display = "block";
  evt.currentTarget.className += " active";
}
</script>
   </body>
</html>
```

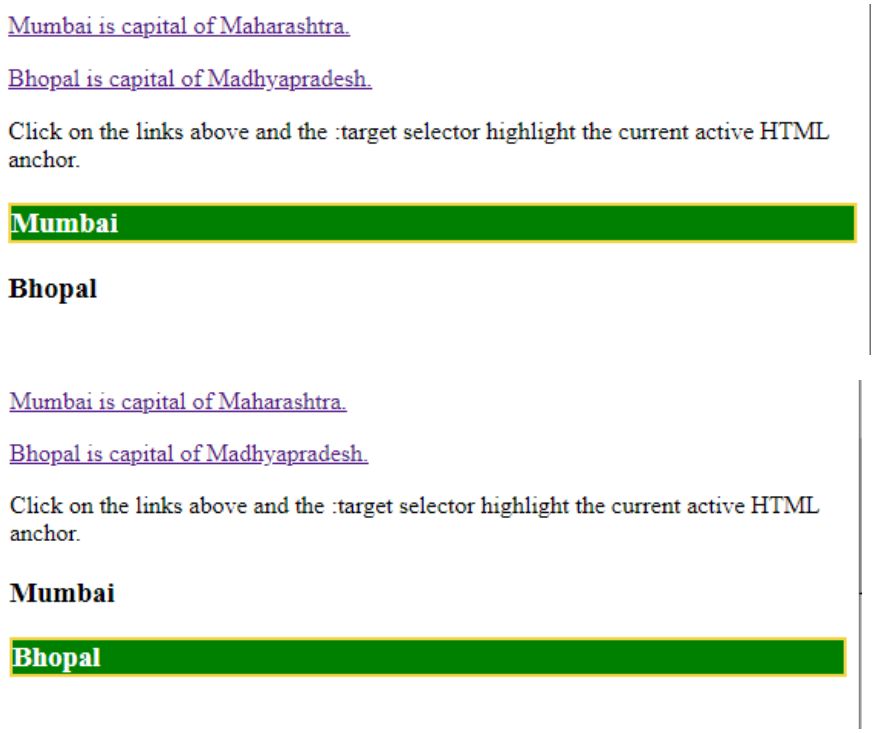
Output:



Code: Following example shows how to create tab menu by using target selector<a>.

```
<html>
<head>
<style>
:target
 {
  color:white;
  border: 2px solid #F4D444;
  background-color:green;
}
</style>
</head>
<body>
<p><a href="#news1">Mumbai is capital of Maharashtra.</a></p>
<p><a href="#news2">Bhopal is capital of Madhyapradesh.</a></p>
<p>Click on the links above and the :target selector highlight the current active HTML anchor.</p>
<h3>
<p id="news1"><b>Mumbai</b></p>
<p id="news2"><b>Bhopal</b></p>
</h3>
</body>
</html>
```

Output:

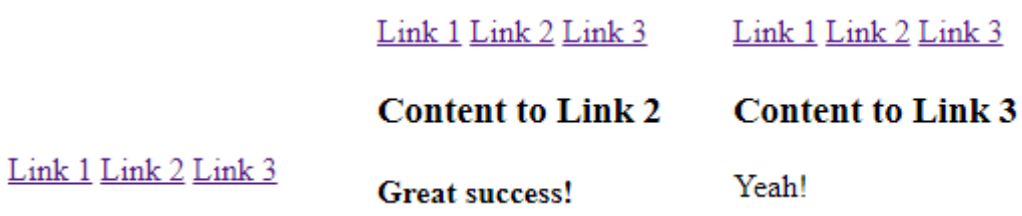


Code:

```
<html>
<head>
<style>
.tab div {
  display: none;
}

.tab div:target {
  display: block;
}
</style>
</head>
<body>

<div class="tab">
```

```
<a href="#link1">Link 1</a>
<a href="#link2">Link 2</a>
<a href="#link3">Link 3</a>

<div id="link1">
  <h3>Content to Link 1</h3>
  <p>Hello World!</p>
</div>

<div id="link2">
  <h3>Content to Link 2</h3>
  <h4>Great success!</h4>
</div>

<div id="link3">
  <h3>Content to Link 3</h3>
  <p>Yeah!</p>
</div>

</div>

</body>
</html>
```

Output:



6.4.7. Popup Menu:

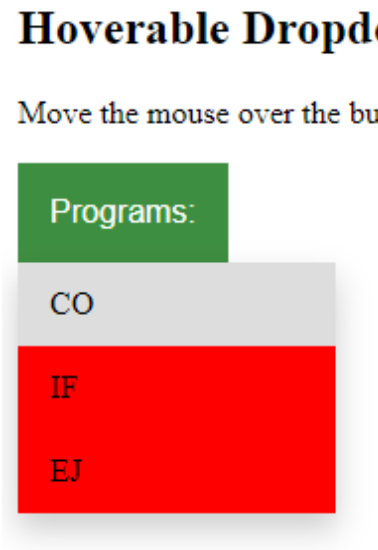
A popup menu appears as the user moves the mouse cursor over a parent menu item. The popup menu contains child menu items that are associated with the parent menu item.

Code:

```
<html>
<head>
<style>
.dropbtn {
  background-color: Blue;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
}
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: red;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}

.dropdown-content a {
  color: black;
  padding: 12px 16px;
```

```
  text-decoration: none;
  display: block;
}

.dropdown-content a:hover {background-color: #ddd;}

.dropdown:hover .dropdown-content {display: block;}

.dropdown:hover .dropbtn {background-color: #3e8e41;}
</style>
</head>
<body>
<h2>Hoverable Dropdown</h2>
<p>Move the mouse over the button to open the dropdown menu.</p>
<div class="dropdown">
  <button class="dropbtn">Programs:</button>
  <div class="dropdown-content">
    <a href="#">CO</a>
    <a href="#">IF</a>
    <a href="#">EJ</a>
  </div>
</div>
</body>
</html>
```

Output:



6.4.8. Sliding Menu:

The slide-in menu appears as a block that floats on the left /right side of the web page. It seems to come alive when the user moves the mouse over the block.

Code:

```
<html>
<head>
<style>
#menu
{
  position: fixed;
  right: -8.5em;
  top: 20%;
  width: 8em;
  background: pink;
  color: red;
  margin: -1;
  padding: 0.5em 0.5em 0.5em 2.5em;
}
#menu:hover
{
right: 0
```

```
}
#menu
{
transition: 0.2s
}
#menu a
{
  position: relative;
  left: 0;
}
#menu a:focus
{
  left: -7em;

}
#menu a { transition:  0.1s }
#menu:hover a:focus {
  left: 0;
  background: none;
}

</style>
</head>
<body>
<h3>
<ul id=menu>
<li><a href="#home">Home</a>
<li><a href="#prog">Programs</a>
<li><a href="#vision">Vision</a>
<li><a href="#mission">Mission</a>
</ul>
</div>
</body>
</html>
```

6.4.9. Highlighted Menu:

User can highlight menu by using following methods:

1) When user performs onmouseover()
2) When user performs onclick()

Code:

```
<html>
<head>
<title>Highlighted Menu Effect</title>
<style>
.link
{
text-decoration: none;
padding: 10px 16px;
background-color:pink;
font-size: 20px;
}

.active, .link:hover
{
background-color:gray;
color:white;
}
</style>
</head>
<body>
Move the mouse over menus:<br><br>
<div id="me">
        <a href="#file" class="link">File</a>
        <a href="#edit" class="link">Edit</a>
        <a href="#view" class="link">View</a>
        <a href="#exit" class="link">Exit</a>
```

```
</div>
</body>
</html>
```

Output:

6.4.10. Folding Tree Menu:

Also known as cascading tree.

The folding tree menu looks like a tree which consists of one or more closed folders, each of these folders further consist of some menu items.

Code:

```
<html>
<head>
<style>
ul, #myUL {
  list-style-type: none;
}

#myUL {
  margin: 0;
  padding: 0;
}

.caret {
  cursor: pointer;
  -webkit-user-select: none; /* Safari 3.1+ */
  -moz-user-select: none; /* Firefox 2+ */
  -ms-user-select: none; /* IE 10+ */
  user-select: none;
}
```

```
.caret::before {
  content: "\25B6";
  color: black;
  display: inline-block;
  margin-right: 6px;
}

.caret-down::before {
  -ms-transform: rotate(90deg); /* IE 9 */
  -webkit-transform: rotate(90deg); /* Safari */'
  transform: rotate(90deg);
}

.nested {
  display: none;
}

.active {
  display: block;
}
</style>
</head>
<body>

<h2>Folding Tree Menu</h2>
<p>A tree menu represents a hierarchical view of information, where each item can have a number of subitems.</p>
<p>Click on the arrow(s) to open or close the tree branches.</p>

<ul id="myUL">
  <li><span class="caret">India</span>
    <ul class="nested">
      <li>Karnataka</li>
```

```
      <li>Tamilnaadu</li>
      <li><span class="caret">Maharashtra</span>
        <ul class="nested">
          <li>Mumbai</li>
          <li>Pune</li>
          <li><span class="caret">Navi Mumbai</span>
            <ul class="nested">
              <li>Nerul</li>
              <li>Vashi</li>
              <li>Panvel</li>

            </ul>
          </li>
        </ul>
      </li>
    </ul>
  </li>
</ul>
<script>
var toggler = document.getElementsByClassName("caret");
var i;
for (i = 0; i < toggler.length; i++) {
  toggler[i].addEventListener("click", function() {
    this.parentElement.querySelector(".nested").classList.toggle("active");
    this.classList.toggle("caret-down");
  });
}
</script>

</body>
</html>
```

Output:

Tree Menu

A tree menu represents a hierarchical view of information, where each item can have a number of subitems.

Click on the arrow(s) to open or close the tree branches.

▼ India
- Karnataka
- Tamilnaadu
- ▼ Maharashtra
  - Mumbai
  - Pune
  - ▼ Navi Mumbai
    - Nerul
    - Vashi
    - Panvel

6.4.11. Context Menu:

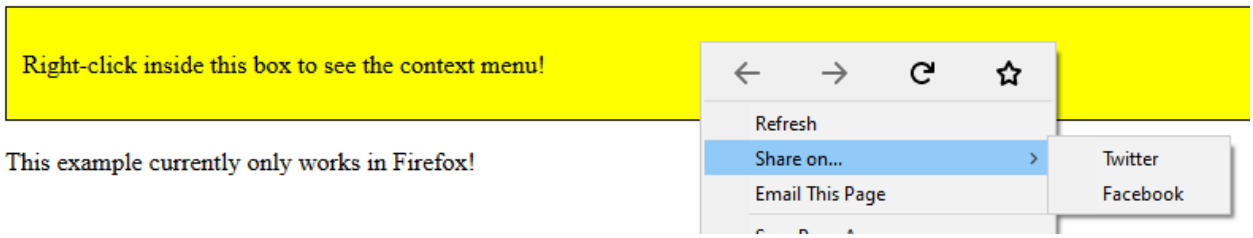
The context menu appears on the web page when the user clicks the right button on the screen.

Code:

```
<html>
<head>
<style>
div {
  background: yellow;
  border: 1px solid black;
  padding: 10px;
}
</style>
</head>
<body>

<div contextmenu="mymenu">
<p>Right-click inside this box to see the context menu!

<menu type="context" id="mymenu">
```

```
  <menuitem label="Refresh" onclick="window.location.reload();"
icon="ico_reload.png"></menuitem>
  <menu label="Share on...">
    <menuitem label="Twitter" icon="ico_twitter.png"
onclick="window.open('//twitter.com/intent/tweet?text=' +
window.location.href);"></menuitem>
    <menuitem label="Facebook" icon="ico_facebook.png"
onclick="window.open('//facebook.com/sharer/sharer.php?u=' +
window.location.href);"></menuitem>
  </menu>
  <menuitem label="Email This Page"
onclick="window.location='mailto:?body='+window.location.href;"></menuitem>
</menu>

</div>

<p>This example currently only works in Firefox!</p>

</body>
</html>
```

Output:



6.4.12. Scrollable Menu:

Scrollbar is different from other menu as it provides two arrowheads.

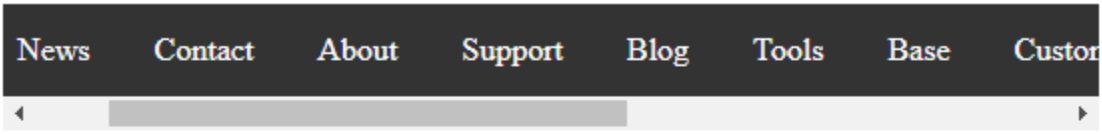
2-ways to implement this type of menu:

1) Horizontal Scrollable Menu:
   Code:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
div.scrollmenu {
  background-color: #333;
  overflow: auto;
  white-space: nowrap;
}

div.scrollmenu a {
  display: inline-block;
  color: white;
  text-align: center;
  padding: 14px;
  text-decoration: none;
}

div.scrollmenu a:hover {
  background-color: #777;
}
</style>
</head>
<body>

<div class="scrollmenu">
  <a href="#home">Home</a>
  <a href="#news">News</a>
  <a href="#contact">Contact</a>
  <a href="#about">About</a>
  <a href="#support">Support</a>
  <a href="#blog">Blog</a>
  <a href="#tools">Tools</a>
```

```
  <a href="#base">Base</a>
  <a href="#custom">Custom</a>
  <a href="#more">More</a>
  <a href="#logo">Logo</a>
  <a href="#friends">Friends</a>
  <a href="#partners">Partners</a>
  <a href="#people">People</a>
  <a href="#work">Work</a>
</div>

<h2>Horizontal Scrollable Menu</h2>
<p>Resize the browser window to see the effect.</p>
</body>
</html>
```

Output:



Horizontal Scrollable Menu

Resize the browser window to see the effect.

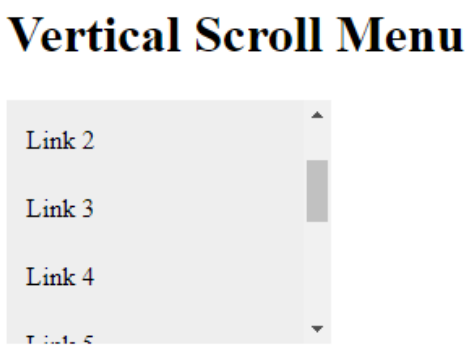
2) Vertical Scrollable Menu:
   Code:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
.vertical-menu {
  width: 200px;
  height: 150px;
  overflow-y: auto;
}
```

```
.vertical-menu a {
  background-color: #eee;
  color: black;
  display: block;
  padding: 12px;
  text-decoration: none;
}

.vertical-menu a:hover {
  background-color: #ccc;
}

.vertical-menu a.active {
  background-color: #4CAF50;
  color: white;
}
</style>
</head>
<body>
<h1>Vertical Scroll Menu</h1>
<div class="vertical-menu">
  <a href="#" class="active">Home</a>
  <a href="#">Link 1</a>
  <a href="#">Link 2</a>
  <a href="#">Link 3</a>
  <a href="#">Link 4</a>
  <a href="#">Link 5</a>
  <a href="#">Link 6</a>
  <a href="#">Link 7</a>
  <a href="#">Link 8</a>
  <a href="#">Link 9</a>
  <a href="#">Link 10</a>
</div>
```

```
</body>
</html>
```

Output:



6.4.13. Side Bar Menu:

The side bar menu displays a menu on the side of the web page.

Code:

```
<html>
<head>
<style>
.sidebar
{
height: 100%;
width:100px;
position:fixed;
background-color: pink;
padding-top:20px;
}

.sidebar a
{
text-deocration:none;
font-size:20px;
color:red;
display:block;
```

```
}

.sidebar a:hover
{
color:white;
}

.main
{
margin-left:160px;
padding:0px 10px;
}

</style>
</head>
<body>

<div class="sidebar">
<a href="#home"> Home</a>
<a href="#vision">Vision </a>
<a href="#mission"> Mission</a>
<a href="#prog">Programs</a>
</div>

<div class="main">
<h2> Side bar Menu </h2>
</div>
</body>
</html>
```

Output:

6.5 Protecting your webpage:

There is nothing secret about your web page. Anyone with a little computer knowledge can use a few mouse clicks to display your HTML code, including your JavaScript, on the screen. In this, you'll learn how to hide your JavaScript and make it difficult for malicious hackers to extract e-mail addresses from your web page.

6.5.1 Hiding Your Code

- Every developer has to admit that, on occasion, they've peeked at the code of a web page or two by right-clicking and choosing View Source from the context menu.
- In fact, this technique is a very common way for developers to learn new techniques for writing HTML and Javascript. However, some developers don't appreciate a colleague snooping around their code and then borrowing their work without permission. This is particularly true about javascript, which are typically more time-consuming to develop than using HTML to build a web page.
- In reality, you cannot hide your HTML code and JavaScript from prying eyes, because a clever developer can easily write a program that pretends to be a browser and calls your web page from your web server, saving the web page to disk, where it can then be opened using an editor. Furthermore, the source code for your web page—including your JavaScript—is stored in the cache, the part of computer memory where the browser stores web pages that were requested by the visitor.

- A sophisticated visitor can access the cache and thereby gain access to the web page source code.
- However, you can place obstacles in the way of a potential peeker. First, you can disable use of the right mouse button on your site so the visitor can't access the View Source menu option on the context menu. This hide both your HTML code and your JavaScript from the visitor. Nevertheless, the visitor can still use the View menu's Source option to display your source code. In addition, you can store your JavaScript on your web server instead of building it into your web page. The browser calls the JavaScript from the web server when it is needed by your web page.
- Using this method, the JavaScript isn't visible to the visitor, even if the visitor views the source code for the web page.

6.5.1.1 Disabling the Right Mouse Button

The following example shows you how to disable the visitor's right mouse button while the browser displays your web page. All the action occurs in the JavaScript that is defined in the <head> tag of the web page.

```
<html>
<head>
<script>
```

```
window.onload = function()
{
document.addEventListener("contextmenu", function(e)
{
e.preventDefault();
}, false);}
</script>
<body>
<h3>Right click on screen,Context Menu is disabled</h3>
</body>
</html>
```

The preventDefault() method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

For example, this can be useful when:

- Clicking on a "Submit" button, prevent it from submitting a form
- Clicking on a link, prevent the link from following the URL

Syntax

```
event.preventDefault()
```

```
<html>
<body>
<a id="myAnchor" href="https://w3schools.com/">Go to W3Schools.com</a>
<script>
document.getElementById("myAnchor").addEventListener("click", function(event){
  event.preventDefault()
});
</script>
</body>
</html>
```

6.5.1.2 Hiding JavaScript

You can hide your JavaScript from a visitor by storing it in an external file on your web server. The external file should have the .js file extension. The browser then calls the external fi le whenever the browser encounters a JavaScript element in the web page. If you look at the source code for the web page, you'll see reference to the external .js file, but you won't see the source code for the JavaScript.

The next example shows how to create and use an external JavaScript fi le. First you must tell the browser that the content of the JavaScript is located in an external fi le on the web server rather than built into the web page. You do this by assigning the fi le name that contains the JavaScript to the *src* attribute of the <script> tag.

Next, you need to define empty functions for each function that you define in the external JavaScript file.

webpage.html

```
<html>
<head>
<script src="mycode.js" languages="javascript" type="text/javascript">
</script>
<body>
<h3> Right Click on screen, Context Menu is disabled</h3>
</body>
</html>
```

mycode.js

```
window.onload=function()
{
document.addEventListener("contextmenu", function(e)
{
e.preventDefault();
}, false);
}
```

6.5.2 Concealing Your E-mail Address

- Many of us have endured spam at some point and have probably blamed every merchant we ever patronized for selling our e-mail address to spammers.
- While e-mail addresses are commodities, it's likely that we ourselves are the culprits who invited spammers to steal our e-mail addresses.
- Here's what happens: Some spammers create programs called bots that surf the
- Net looking for e-mail addresses that are embedded into web pages, such as those placed there by developers to enable visitors to contact them. The bots then strip these e-mail addresses from the web page and store them for use in a spam attack.
- This technique places developers between a rock and a hard place. If they place their e-mail addresses on the web page, they might get slammed by spammers.
- If they don't display their e-mail addresses, visitors will not be able to get in touch with the developers.
- The solution to this common problem is to conceal your e-mail address in the source code of your web page so that bots can't find it but so that it still appears on the web page.
- Typically, bots identify e-mail addresses in two ways: by the mailto: attribute that tells the browser the e-mail address to use when the visitor wants to respond to the web page, and by the @ sign that is required of all e-mail addresses. Your job is to confuse the bots by using a JavaScript to generate the e-mail address dynamically. However, you'll still need to conceal the e-mail address in your JavaScript, unless the JavaScript is contained in an external JavaScript file, because a bot can easily recognize the mailto: attribute and the @ sign in a JavaScript.
- Bots can also easily recognize when an external file is referenced.
- To conceal an e-mail address, you need to create strings that contain part of the e-mail address and then build a JavaScript that assembles those strings into the e-mail address, which is then written to the web page.
- The following example illustrates one of many ways to conceal an e-mail address.
- It also shows you how to write the subject line of the e-mail. We begin by creating four strings:
    - The first string contains the addressee and the domain along with symbols &, *, and _ (underscore) to confuse the bot.
    - The second and third strings contain portions of the mailto: attribute name. Remember that the bot is likely looking for mailto:
    - The fourth string contains the subject line. As you'll recall from your HTML training, you can generate the TO, CC, BCC, subject, and body of an e-mail from within a web page.
- You then use these four strings to build the e-mail address. This process starts by using the replace() method of the string object to replace the & with the @ sign

and the * with a period (.). The underscores are replaced with nothing, which is the same as simply removing the underscores from the string.

- All the strings are then concatenated and assigned to the variable b, which is then assigned the location attribute of the window object. This calls the e-mail program on the visitor's computer and populates the TO and Subject lines with the strings generated by the JavaScript.

```
<html >
<head>
<title>Conceal Email Address</title>
<script>

function CreateEmailAddress()
{
var x = 'abcxyz*c_o_m'
var y = 'mai'
var z = 'lto'
var s = '?subject=Customer Inquiry'
x = x.replace('&','@')
x = x.replace('*','.')
x = x.replace('_','')
x = x.replace('_','')
var b = y + z +':'+ x + s
window.location=b;
}

</script>
</head>
<body>
<input type="button" value="send" onclick="CreateEmailAddress()">
</body>
</html>
```

6.6 Frameworks of JavaScript and its application

JavaScript is a multi-paradigm language that supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. JavaScript was initially used only for the client-side. However, these days, JavaScript is used as a server-side programming language as well. To summarize, in just a simple sentence - JavaScript is the language of the web.

JavaScript framework is an application framework written in JavaScript where the programmers can manipulate the functions and use them for their convenience.

Frameworks are more adaptable for the designing of websites, and hence, most of the website developers prefer it. JavaScript frameworks are a type of tool that makes working with JavaScript easier and smoother. These frameworks also make it possible for the programmer to code the application as a device responsive.
Following are the most used framework of JavaScript:

1) React

React is not among the earliest disruptive JavaScript-based Web frameworks. But it is the most disruptive and influential JavaScript-based Web framework. Jordan Walke and a group of Facebook Engineers created React in 2013 as a Component-based Web Framework with one-way data flow and changed the Front-end Web Development forever. It also introduced many other concepts like functional, declarative programming, immutable state, which was uncommon in Front-end development. The other breakthrough of React was to introduce the Virtual DOM, which gives better user experience and performance gain.

Features

- **Declarative:** Creates interactive and dynamic UI for websites and mobile applications. React updates efficiently and render the right components when data changes. Declarative views make the code readable and easy to debug.

- **Virtual DOM**: For every DOM object, there is a corresponding "virtual DOM object." It creates a virtual copy of the original DOM and is a representation of a DOM object,
- **Event handling**: React has its fully compatible W3C object model event system created. It also provides a cross-browser interface to a native event, meaning no need to worry about incompatible event names and fields. React reduces memory head by as event system is implemented through event delegation and has a pool of event objects.
- **JSX:** JSX is a markup syntax that closely resembles HTML. JSX makes writing React components easier by making the syntax almost identical to the HTML injected into the web page.
- **Performance:** React uses one-way data binding with an application architecture called Flux controls. ReactJS helps update the View for the user and, Flux controls the application workflow. Virtual DOM adds advantages as it compares the new data with original DOM and updates the View automatically.
- **React Native**: React Native is a custom renderer for React; it uses native components instead of web components like React as building blocks. It also serves access to these platforms' features, apart from transforming React code to work on iOS and Android.
- **Component-Based**: In React, everything is a component of the web page divided into small components to create a view(or UIs). Every part of the application visuals would be wrapped inside a self-contained module known as a component. Components in ReactJS use to define the visuals and interactions in applications.

2. Node.js

- In 2009, Ryan Dahl created the asynchronous, event-driven Server-Side JavaScript runtime Node.js and brought JavaScript in the uncharted territory of Back-end development.
- Ryan Dahl has used the popular JavaScript Engine V8 and C++ libraries. Since then, the popularity of both Node.js and JavaScript has skyrocketed.
- With Node Package Manager NPM and countless numbers of frameworks/libraries, Node.js has surpassed many other established Server-side frameworks.

- Because of its Asynchronous Event-Driven nature and lightweight, fast runtime, Node.js is especially suited for I/O heavy applications like Web, IoT, Serverless.
- Node.js is one of the primary driving force to improve JavaScript as a programming language and to increase the popularity of JavaScript.

Features:

- **It is swift:**
  The library of Node.js is fast when it comes to code execution, as it is built on the V8 JavaScript engine of Google Chrome.
- **I/O is asynchronous and Event-Driven:**
  All the APIs are asynchronous, which means that its server does not wait for the API to come back with data. Here the server calls the APIs one by one and keeps moving to the next one while using a notification mechanism of Events to generate a response from the API, called previously. This makes it fast too.
- **Single-threaded:**
  Node.js, along with event looping, follows a single-threaded model.
- **Highly scalable:**
  Node.js follows an event mechanism that makes it possible for the server to respond in a non-blocking manner, which eventually makes it scalable.
- **No buffering:**
  When it comes to uploading audio and video files, Node.js cuts down the processing time significantly. It does not buffer any data, and here the application gets out the data in chunks.
- **Open source:**
  Being open-source, Node.js's community has come up with several amazing models that can be used to add better capabilities to the Node.js applications.
- **License:**
  It has been released under MIT license.

3. Vue.js

In modern days where Web frameworks are backed by Giant Tech companies, Vue.js is an exception. In 2014, an ex-Google Engineer Evan You decided to combine the good parts of AngularJS (View Layer) and the good parts of React (Virtual DOM) and created Vue.js. Today, Vue.js is one of the most popular JavaScript-based Web frameworks. One of the

key design goals of Evan You was to lower the barrier into JavaScript-based front-end development. Vue.js is one of the easiest Front-end frameworks where developers can write SPA applications with minor effort.
Developers can use Vue.js as an End-to-End framework with Routing, State management like Angular, or as only a view layer like React. It also offers Angular like two-way data-binding with additional Reactivity and React like rendering using Virtual DOM.

Features:

- **Virtual DOM**: Vue.js utilizes virtual DOM. Virtual DOM is a clone of the principal DOM element. The virtual DOM absorbs every change intended for the DOM presents in the form of JavaScript data structures, which are compared with the original data structure.
- The viewers view final changes that reflect in the real DOM. The method is creative and cost-effective; also, the changes are done quickly.
- **Data Binding:** This feature facilitates to manipulate or assign values to HTML attributes., change the style, assign classes with v-bind available, which is a binding directive.
- **CSS Transitions and Animations:** This feature provides several methods to apply a transition to HTML elements when added, updated, or removed from the DOM. Its features consist of a built-in component that wraps the element responsible for returning the transition effect.
- **Template:** It provides HTML-based templates that bind the DOM with the Vue.js instance data. The templates are compiled into Virtual DOM Render functions. A developer can use the render functions template and can replace the template with the render function.
- **Methods:** We use methods when an event occurs that isn't necessarily related to the instance data being mutated or want to change a component's state. Methods do not keep records of any dependencies but can take arguments.
- **Complexity:** Vue.js is simpler in terms of API and design. A web developer builds simple applications in a single day.

4. Angular

In AngularJS, Google had created one of the earliest hot JavaScript-based Front-end frameworks in 2010. But once Facebook released React, it exposed the design flaws of AngularJS, and it quickly became an outdated framework. As a result, the Google team has created an entirely new SPA framework and released it as Angular in 2016. Although Angular and AngularJS have similar names, in reality, they are two different frameworks. Unlike React, it is an end-to-end Framework with "Out-of-the-box" support of everything one needs to develop an Enterprise-grade Web App. Also, Angular is the first significant framework that has embraced TypeScript and played a considerable role in making TypeScript popular.

Features:

- Angular.js is an end-to-end framework with "out of the box" support to develop Enterprise Application. In Angular CLI, it has one of the best Command-Line Tool in the JavaScript landscape.
- With TypeScript and separating the template from styling and business logic, it is especially suited for the enterprise-grade large code-base.
- It is inherently the most secure Front-end framework with built-in features like DOM sanitization.
- Although Google is not backing Angular the same way as Facebook is backing React, it is still putting enough resources so that Angular remains an attractive and innovative framework. Recently it has added Lazy Loading, Differential loading to improve loading time of modules.
- In Angular 9, it releases a new rendering Engine Ivy to improve startup time, response time, and to reduce bundle size.

5. Express

When Node.js appeared in 2009, TJ Holowaychuk has created Express.js based on the minimalistic Web Framework Sinatra. It is a minimalistic Web framework to develop Web application and REST API. It is also less opinionated and very fast. Many other JavaScript-based Web frameworks are based on Express.Today, Express.js is the most popular JavaScript-based Web application framework hands down.

Features:

- Express.js is almost the default JavaScript Server Side framework.
- Express is the complete Application framework with middleware, routing, template.
- Express supports MVC pattern with View system supporting 14+ templating engines.
- It also offers robust routing.
- Express also supports content negotiation.

6. Next.js

React is a very unopinionated framework where React-Core just offers the view layer. There was always a need for an end-to-end, opinionated framework based on React. Tim Neutkens and a group of Software Engineers from the Dutch company Zeit has created Next.js as an end-to-end, higher-level Web Framework on top of React and Node.js. Next.js offers both Server-Rendered and Static Web sites for Web, Desktop, and Mobile devices.

Features:

- Next.js is built upon the two most popular and battle-hardened JavaScript frameworks: React and Node.js.
- It also offers "Build once, runs everywhere," i.e., a Next.js can run on Web, Mobile, and Desktop.
- Next.js offers excellent Server-Side rendering with exceptional SEO support and fast startup.

- It offers automatic code splitting and filesystem-based routing.
- It also supports easy-to-use data fetching and built-in CSS support.

7.Meteor

In 2012, a group of Engineers had created Meteor as an isomorphic, open-source full-stack JavaScript framework based on Node.js. It also supports building end-to-end applications for Web, Mobile, Desktop platform and integrates well with popular front-end frameworks like React, Vue.js, Angular, Svelte. It is also a "Batteries Included" framework with "Out-of-the-box" support for Enterprise-grade App development.

Features:

- Meteor is a full-stack framework to develop the complete stack: Frontend-to-Backend.
- For front-end development, it has its own template engine. But developers can use Meteor with other popular front-end frameworks like Angular, React, Vue.js or Svelte.
- It is a cross-platform framework and can develop an application for Web, Mobile, and Desktop.
- Meteor has integrated JavaScript stack, which enables different integrating technologies (e.g., MongoDB database, React front-end) with minimum effort.
- It is an Isomorphic platform sharing the same API on client-side and server-side.

8. Svelte

In 2016, a Guardian Software Engineer Rich Harris had the groundbreaking idea to develop a JavaScript framework with no framework-specific Runtime and released Svelte. The idea was to use the Svelte compiler, which would compile framework-specific code

to plain JavaScript, HTML, CSS, and render the compiled code to the browser. Although the concept was not new in software development, it was uncharted territory in Front-end development. The other significant contribution of Svelte is to add first-class support of reactivity, which leads to faster, improved performance without Virtual DOM. Today, it is arguably the hottest Front-end framework with tremendous traction and interest in the industry.

Features:

- It is a compile-time framework and does not need any framework-specific runtime. It has the smallest bundle size among all frameworks.
- Svelte performs the DOM rendering via reactive programming, which is faster than Virtual DOM most times. As a result, Svelte gives the fastest rendering among all frameworks.
- Svelte is just a View layer like React-Core, and it is an unopinionated framework.
- Svelte supports both client-side and server-side rendering with excellent SEO support.
- Developers can use Svelte to develop a Web app, Cross-platform Mobile App development, or Desktop app development.

9. Koa

In 2013, the core members of Express.js led by TJ Holowaychuk had created Koa as a lightweight, modern, expressive, and robust middleware framework for Web Applications and APIs. Koa is hugely modular with tiny Core with no middleware. However, middleware is available as separate modules.

Features:

- Koa has a lightweight, smaller Core with no out-of-the-box Middleware bundle.
- Koa has a highly modular architecture and offers pluggable middleware Modules.
- Koa supports cascading middleware in a stack-like manner, which allows to perform actions downstream then and manipulate the response upstream.
- Koa uses async/await instead of callback and supports cleaner, expressive code with better error handling.
- In terms of performance, it outperforms Express.js.

10. Ember.js.

Inspired by the Ruby on Rails principle "Convention over Configuration," Yehuda Katz from Apple has created Ember.js as a highly opinionated, end-to-end framework in 2012. Ember.js is a strictly backward compatible framework introducing no significant breaking changes since its inception. Where other frameworks from that era (Backbone.js, AngularJS) are diminishing in popularity, Ember.js is still giving a reliable, productive framework to fulfill the need of modern Front-end development.

Features:

- End-to-end opinionated cohesive framework focusing on "Convention over Configuration."
- Instead of one Tech giant, Ember is backed by several Tech Giant like LinkedIn, Yahoo. As a result, it is not driven by one corporation's needs.
- Ember's Data library is the best to access data across multiple sources at once, set up asynchronous relationships.
- In Ember CLI, it has the best CLI among all JavaScript frameworks, which helps to scaffold and generating all the necessary codes with the right structure, including all dependencies.
- In its latest release Ember Octane, it has introduced HTML first and component first approach with improved support for state management and reactivity.

11. Backbone.js

It is one of the most popular JavaScript frameworks. It is effortless to understand and learn. It can be used to create Single Page Applications. The development of this framework involves the idea that all the server-side functions must flow through an API, which would help achieve complex functionalities by writing less code.

Features:

- BackboneJS uses JavaScript functions, making the development of applications and the frontend in a much easier.

- Building blocks such as models, views, events, routers, and collections are provided for assembling the client-side web applications.
- It is a simple library that helps in separating business and user interface logic.
- It is a free and open-source library and contains over 100 available extensions.
- It is a backbone for any project and helps in the organization of the code.
- BackboneJS has a soft dependency on jQuery and a hard dependency on Underscore.js.
- It allows us to create client-side web applications or mobile applications in a well-structured and organized format.

12.Aurelia

Aurelia framework is the latest version of JavaScript, which can be used to implement any interface. It is the next generation of the framework for developing far more robust websites. The framework of Aurelia can extend the HTML for various purposes, including data binding. Also, its modern architecture ensures that the purpose of toll is for interpretation client-side and server-side at a time.

Features:

- Components: Components are building blocks of the Aurelia framework and are composed of JavaScript view-model pairs and HTML views.
- Web Standards: It is one of the cleanest modern frameworks. It completely focuses on web standards without unnecessary abstractions.
- Extensible: The framework facilitates an easy way to integrate with the other needed tools.
- Commercial Support: This framework offers commercial and enterprise support.
- License: Aurelia is open-sourced and licensed under MIT license.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<script>
var subjectObject = {
```

```
  "Front-end": {
    "HTML": ["Links", "Images", "Tables", "Lists"],
    "CSS": ["Borders", "Margins", "Backgrounds", "Float"],
    "JavaScript": ["Variables", "Operators", "Functions", "Conditions"]
  },
  "Back-end": {
    "PHP": ["Variables", "Strings", "Arrays"],
    "SQL": ["SELECT", "UPDATE", "DELETE"]
  }
}
window.onload = function() {
  var subjectSel = document.getElementById("subject");
  var topicSel = document.getElementById("topic");
  var chapterSel = document.getElementById("chapter");
  for (var x in subjectObject) {
    subjectSel.options[subjectSel.options.length] = new Option(x, x);
  }
  subjectSel.onchange = function() {
    //empty Chapters- and Topics- dropdowns
    chapterSel.length = 1;
    topicSel.length = 1;
    //display correct values
    for (var y in subjectObject[this.value]) {
      topicSel.options[topicSel.options.length] = new Option(y, y);
    }
  }
  topicSel.onchange = function() {
    //empty Chapters dropdown
    chapterSel.length = 1;
    //display correct values
    var z = subjectObject[subjectSel.value][this.value];
    for (var i = 0; i < z.length; i++) {
      chapterSel.options[chapterSel.options.length] = new Option(z[i], z[i]);
    }
  }
```

```
}
</script>
</head>
<body>

<h1>Cascading Dropdown Example</h1>

<form name="form1" id="form1" action="/action_page.php">
Subjects: <select name="subject" id="subject">
    <option value="" selected="selected">Select subject</option>
  </select>
  <br><br>
Topics: <select name="topic" id="topic">
    <option value="" selected="selected">Please select subject first</option>
  </select>
  <br><br>
Chapters: <select name="chapter" id="chapter">
    <option value="" selected="selected">Please select topic first</option>
  </select>
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML and CSS Slideshow</title>
    <style>
    body {
        font-family: Helvetica, sans-serif;
        padding: 5%;
        text-align: center;
        font-size: 50;
    }

    /* Styling the area of the slides */

    #slideshow {
        overflow: hidden;
        height: 510px;
        width: 728px;
        margin: 0 auto;
    }

    /* Style each of the sides
    with a fixed width and height */

    .slide {
        float: left;
        height: 510px;
        width: 728px;
    }

    /* Add animation to the slides */

    .slide-wrapper {

        /* Calculate the total width on the
      basis of number of slides */
        width: calc(728px * 4);

        /* Specify the animation with the
      duration and speed */
        animation: slide 10s ease infinite;
    }

    /* Set the background color
    of each of the slides */

    .slide:nth-child(1) {
        background: green;
    }

    .slide:nth-child(2) {
        background: pink;
```

```
    }

    .slide:nth-child(3) {
        background: red;
    }

    .slide:nth-child(4) {
        background: yellow;
    }

    /* Define the animation
    for the slideshow */

    @keyframes slide {

        /* Calculate the margin-left for
      each of the slides */
        20% {
            margin-left: 0px;
        }
        40% {
            margin-left: calc(-728px * 1);
        }
        60% {
            margin-left: calc(-728px * 2);
        }
        80% {
            margin-left: calc(-728px * 3);
        }
    }
    </style>
</head>

<body>

    <!-- Define the slideshow container -->
    <div id="slideshow">
        <div class="slide-wrapper">

            <!-- Define each of the slides
         and write the content -->
            <div class="slide">
                <h1 class="slide-number">
                    GeeksforGeeks
                </h1>
            </div>
            <div class="slide">
                <h1 class="slide-number">
                    A computer science portal
                </h1>
            </div>
            <div class="slide">
                <h1 class="slide-number">
                    This is an example of
                </h1>
```

```
            </div>
            <div class="slide">
                <h1 class="slide-number">
                    Slideshow with HTML and CSS only
                </h1>
            </div>
        </div>
    </div>
</body>
</html>
```