

# Chapter 6:

## Menus, navigation and web page protection

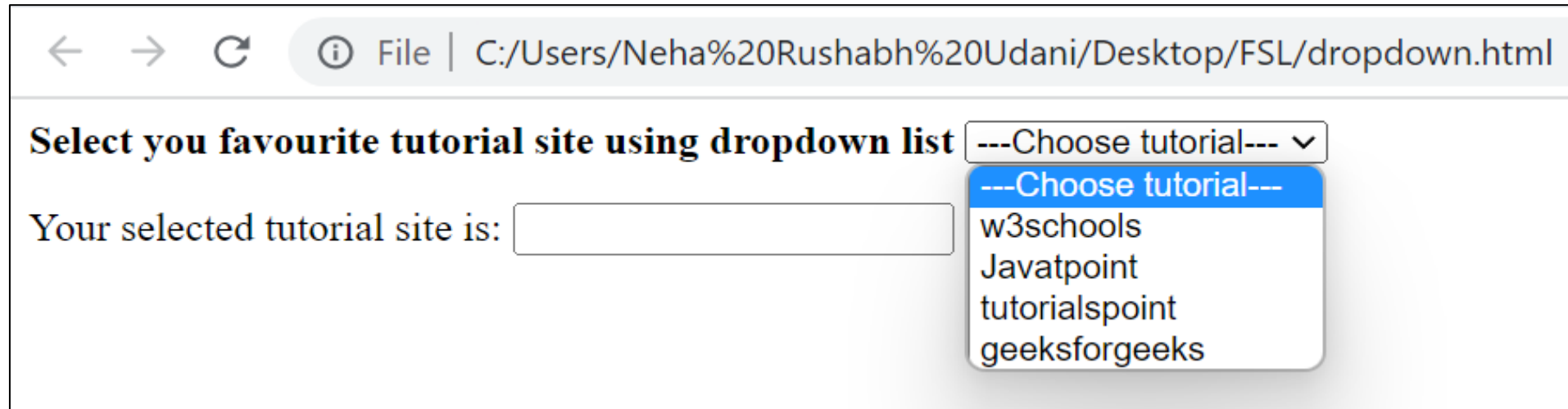
## dropdown list using <select> tab

```
<html><head><title>dropdown menu using select tab</title></head>
<script>
function favTutorial() {
let myList1 = document.getElementById("myList");
document.getElementById("favourite").value = myList1.options[myList1.selectedIndex].text;
}
</script><body><form>
<b> Select you favourite tutorial site using dropdown list </b>
<select id = "myList" onchange = "favTutorial()" >
<option> ---Choose tutorial--- </option>
<option> w3schools </option>
<option> Javatpoint </option>
<option> tutorialspoint </option>
<option> geeksforgeeks </option>
</select>
<p> Your selected tutorial site is:
<input type = "text" id = "favourite" size = "20" </p>
</form>
</body>
</html>
```

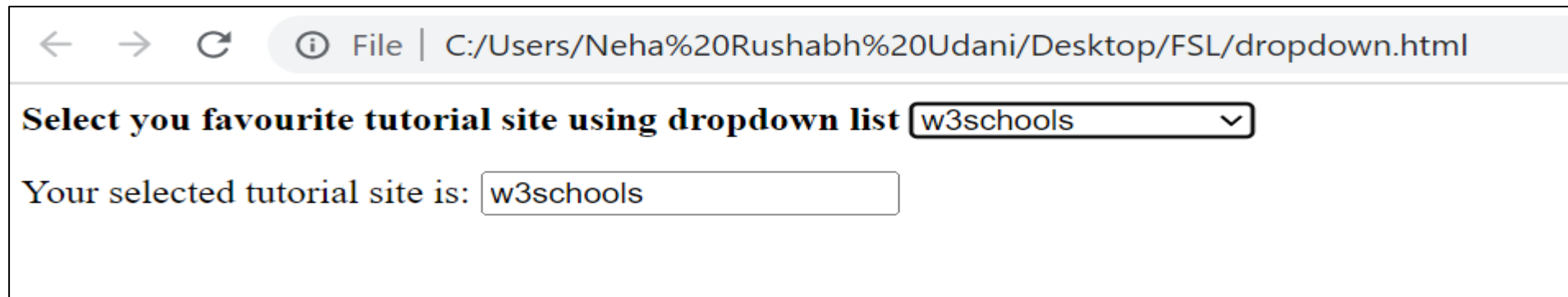
**The selectedIndex** property sets or returns the index of the selected option in a drop-down list.

The index starts at 0.

**Note:** If the drop-down list allows multiple selections it will only return the index of the first option selected.



A screenshot of a web browser window displaying a web page titled "Select you favourite tutorial site using dropdown list". The browser's address bar shows the file path "C:/Users/Neha%20Rushabh%20Udani/Desktop/FSL/dropdown.html". The page contains a label "Your selected tutorial site is:" followed by an empty text input field. To the right of the input field is a dropdown menu with the placeholder text "---Choose tutorial---". The dropdown menu is open, showing a list of options: "w3schools", "Javatpoint", "tutorialspoint", and "geeksforgeeks". The first option, "w3schools", is highlighted in blue.

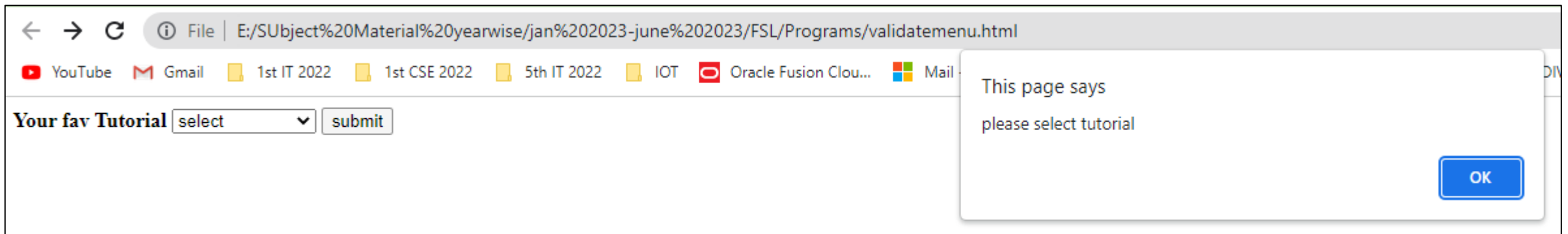
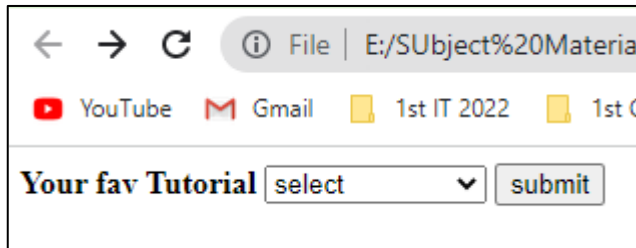


A screenshot of the same web browser window showing the same web page. The dropdown menu is now closed, and the selected option "w3schools" is visible in the dropdown box. The text input field below the label "Your selected tutorial site is:" now contains the text "w3schools".

## validating menu selection

```
<html><title>validating menu selection</title>
<script>
function validate()
{
    var s1=document.getElementById('myList').value;
    if(s1=="select")
    {
        alert("please select tutorial");
        return false;
    }
    return true;
}
</script>
```

```
<body>
<form name="form1" onsubmit = "return validate()"
action="google.com">
<b>Your fav Tutorial</b>
<select id = "myList">
<option>select</option>
<option> w3schools </option>
<option> Javatpoint </option>
<option> tutorialspoint </option>
<option> geeksforgeeks </option>
</select>
<input type="submit" value="submit">
</body></html>
```



# Dropdown list using button and div tab

## Block vs. Inline

HTML tags like <div>, <p>, <ul> take full-width of space and each starts with a new line, whereas other HTML tags like <span>, <img> or <a> don't need a new line and can be placed side by side.

This is because of the different display behaviors: Block or inline.

```
<body>
  <p>I'm a paragraph</p>
  <p>I'm a paragraph too</p>
  <span>I'm a word</span>
  <span>I'm a word too.</span>
</body>
```

I'm a paragraph

I'm a paragraph too

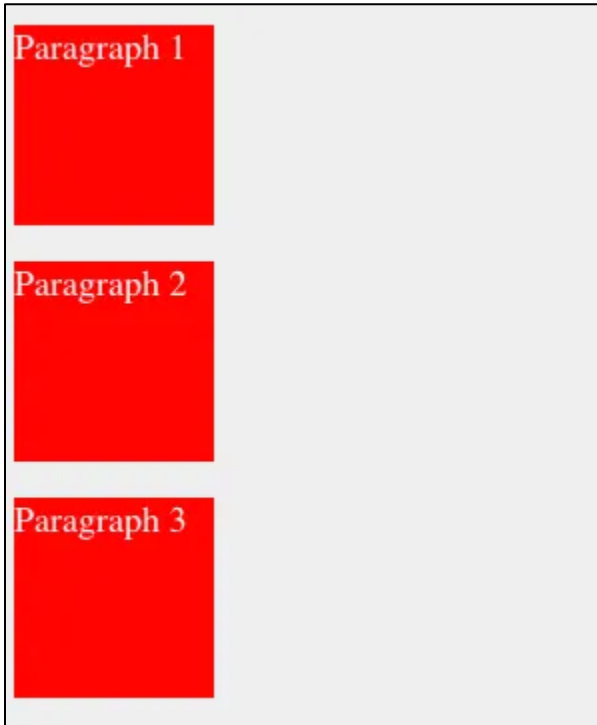
I'm a word I'm a word too.

### **Block-level elements**

- Take full-width (100% width) by default
- Each gets displayed in a new line
- Width & height properties can be set

Since <p> tags are block-level elements, width & height properties can be set:

```
p {  
  height: 100px;  
  width: 100px;  
  background: red;  
  color: white;  
  display: block;  
}
```



If no width was declared here, then the default width of <p> would be 100%. However, We declared a 100px width and the next <p> element still starts with a new line:

## Inline elements

- Take only as much space as they need
- Displayed side by side
- Don't accept width or height properties, and top-bottom margin

```
p {  
  height: 100px;  
  width: 100px;  
  background: red;  
  color: white;  
  display: inline;  
}
```

Since our <p> tag is now an inline element, they will be placed side by side and the width & height properties have no effect anymore:

Paragraph 1Paragraph 2Paragraph 3

## Display: Inline-block

In some cases, both of the display values may not be enough for better web design. At that point, a third display behavior comes to the rescue and also makes alignment much easier: `display: inline-block`.

`display: inline-block` declaration shows both the characteristics of inline and block-level elements.

```
p {  
  display: inline-block;  
  height: 100px;  
  width: 100px;  
  background: red;  
  color: white;  
}
```





## Dropdown list using button and div tab

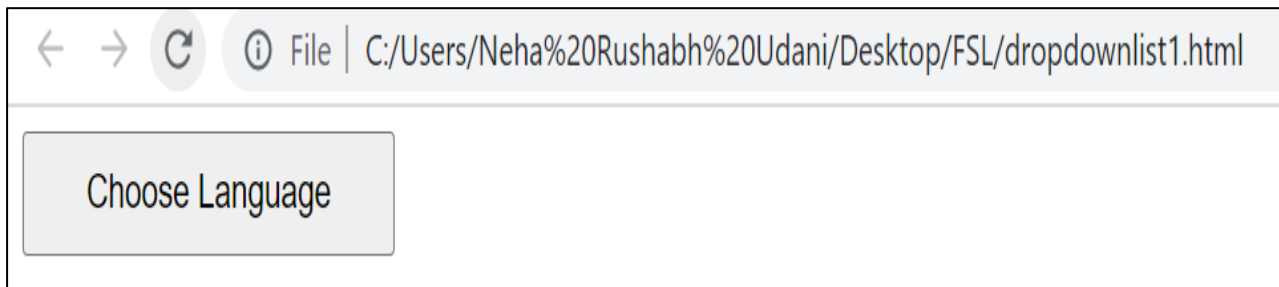
```
<html> <head> <title>dropdown menu using  
button</title> </head> <style>  
  
/* set the position of dropdown */  
.dropdown {  
    position: relative;  
    display: inline-block;  
}  
/* set the size and position of button on web */  
.button {  
    padding: 10px 30px;  
    font-size: 15px;  
}  
/* provide css to background of list items */  
#list-items {  
    display: none;  
    position: absolute;  
    background-color: white;  
    min-width: 185px;  
}
```

```
/* provide css to list items */  
#list-items a {  
    display: block;  
    font-size: 18px;  
    background-color: #ddd;  
    color: black;  
    text-decoration: none;  
    padding: 10px;  
}  
</style>
```

```
<script>
  //show and hide dropdown list item on button click
  function show_hide() {
    var click = document.getElementById("list-items");
    if(click.style.display === "none") {
      click.style.display = "block";
    } else {
      click.style.display = "none";
    }
  }
</script>
```

```
<body>
<div class="dropdown">
  <button onclick="show_hide()" class="button">Choose
  Language</button>
  <center>
    <!-- dropdown list items will show when we click the
    drop button -->
    <div id="list-items">
      <a href="#"> Hindi </a>
      <a href="#"> English </a>
      <a href="#"> Spanish </a>
      <a href="#"> Chinese </a>
      <a href="#"> Japanese </a>
    </div>
  </center>

</body>
</html>
```



## dynamically changing a menu-Chain Menu:

Chain of pull-down menus in which the option selected from the first pull-down menu determines the options that are available in the second pull-down menu.

### Document createElement()

The createElement() method creates an element node.

```
<html>
<body>
<h1>The Document Object</h1>
<h2>The createElement() Method</h2>
<p>Create a p element with some text:</p>
<script>
// Create element:
const para = document.createElement("p");
para.innerText = "This is a paragraph.";
// Append to body:
document.body.appendChild(para);
</script>
</body>
</html>
```

The Document Object  
The createElement() Method  
Create a p element with some text:

This is a paragraph.

```
const para = document.createElement("h1");
para.innerText = "This is a paragraph.";
```

The Document Object  
The createElement() Method  
Create a p element with some text:

**This is a paragraph.**

## dynamically changing a menu

```
<html>
<script>
function populate(s1,s2){
    var sel1 = document.getElementById(s1);
    var sel2 = document.getElementById(s2);
    sel2.innerHTML = "";
    if(sel1.value == "Chevy"){
        var optionArray = ["|", "camaro | Camaro", "corvette | Corvette", "impala | Impala"];
    } else if(sel1.value == "Dodge"){
        var optionArray = ["|", "avenger | Avenger", "challenger | Challenger", "charger | Charger"];
    } else if(sel1.value == "Ford"){
        var optionArray = ["|", "mustang | Mustang", "shelby | Shelby"];
    }
    for(var option in optionArray){
        var pair = optionArray[option].split("|");
        var newOption = document.createElement("option");
        newOption.value = pair[0];
        newOption.innerHTML = pair[1];
        sel2.options.add(newOption);
    }
}
</script>
```

```
<body>
<h2>Choose Your Car</h2>
<hr />
Choose Car Make:
<select id="slct1" name="slct1" onchange="populate(this.id,'slct2')">
  <option value=""></option>
  <option value="Chevy">Chevy</option>
  <option value="Dodge">Dodge</option>
  <option value="Ford">Ford</option>
</select>
<hr />
Choose Car Model:
<select id="slct2" name="slct2"></select>
<hr />
</body>
</html>
```

← → ↻ ⓘ File | C:/Users/Neha%2

## Choose Your Car

Choose Car Make:

Choose Car Model:

← → ↻ ⓘ File | C:/Users/Ne

## Choose Your Car

Choose Car Make:

Choose Car Model:

Mustang

Shelby

← → ↻ ⓘ File | C:/Users/Neha%

## Choose Your Car

Choose Car Make:

Choose Car Model:

Avenger

Challenger

Charger

← → ↻ ⓘ File | C:/Users/Neha%

## Choose Your Car

Choose Car Make:

Choose Car Model:

Camaro

Corvette

Impala

## Status bar in javascript:

The window.statusbar property returns a Statusbar object representing the status bar of browser. It appears at the bottom of browser. However, it is almost impossible for you to interact with the Statusbar via Javascript

`window.statusbar`

// Or simple:

Statusbar

The tendency of modern browsers is to make the Viewport window as wide as possible, therefore, they remove other components like Statusbar, or make Menubar smaller



For modern browsers, you see the status bar appear only when a user moves the mouse on the surface of a link.

## statusbar.visible

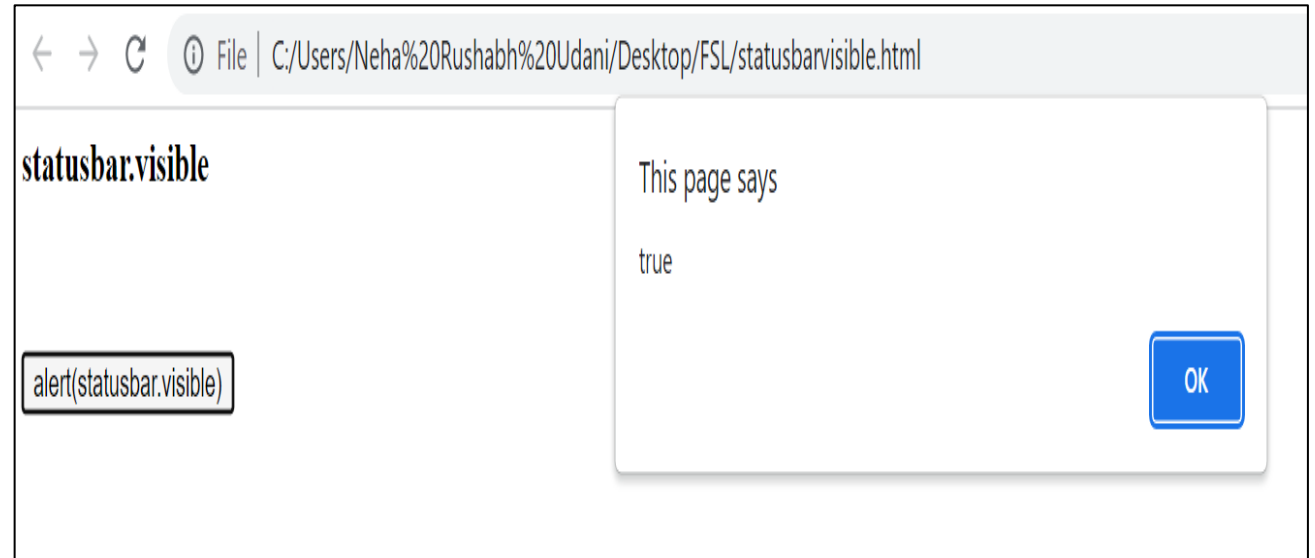
The statusbar.visible property returns true if the status bar is displayed on browser. However, this is unreliable property. You get a true value, which does not mean that you are seeing the status bar.

```
<html>
<head>
  <title>Statusbar</title>
  <meta charset="UTF-8">

</head>
<body>
  <h3>statusbar.visible</h3>

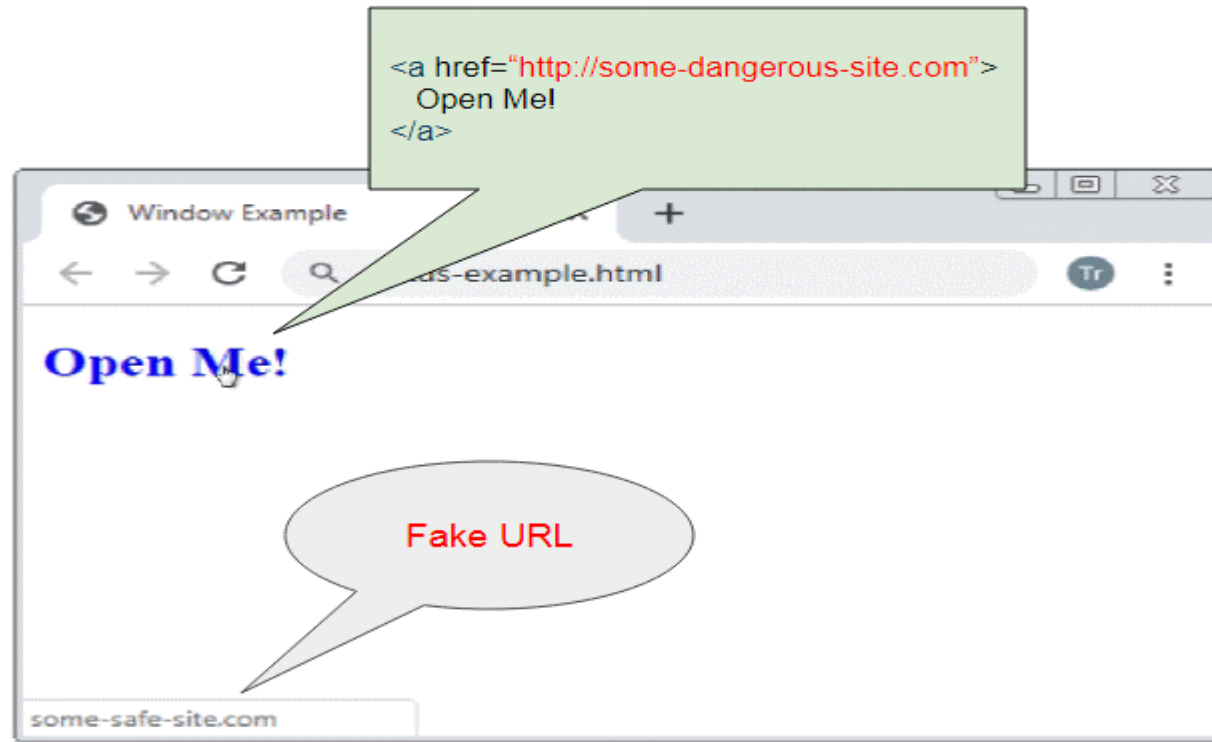
  <br/><br/>
  <button onclick="alert(statusbar.visible)">
    alert(statusbar.visible)
  </button>

</body>
</html>
```





The status property of the window object help you establish a content of text shown on the status bar. By default, for security reasons, most browsers disable this feature for JavaScript. However, if an user wants, they are able to enable this feature for JavaScript by entering the "Options" of the browser.



Before clicking on a link, users often move the mouse over the link's surface to preview its address displayed in the status bar, and only click this link when they feel safe. Some websites may take advantage of `window.status` to display a fake content.

**The status property is deprecated.**  
**It should be avoided to prevent RUN-TIME ERRORS in the future.**

## Status Bar Example:1

```
<html>
  <head>
    <title>JavaScript Status Bar</title></head>
    <body onLoad="window.status='Welcome!';return true">
  </body>
</html>
```

onLoad tells the browser that as soon as your page finished loading, it will display in your current window's status bar (window.status) the message "Welcome!". The return true is necessary because without it, it won't work.

## Status Bar Example: 2

```
<html>
  <head>
    <title>JavaScript Status Bar</title>
  </head>
  <body>
    <a href="http://www.htmlcenter.com"
      onMouseOver="window.status='HTMLcenter';return true"
      onMouseOut="window.status= '';return true">

      HTMLcenter

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

When the user moves his mouse over the link, it will display “HTMLcenter” in the status bar. When he moves his mouse away from the link the status bar will display nothing.

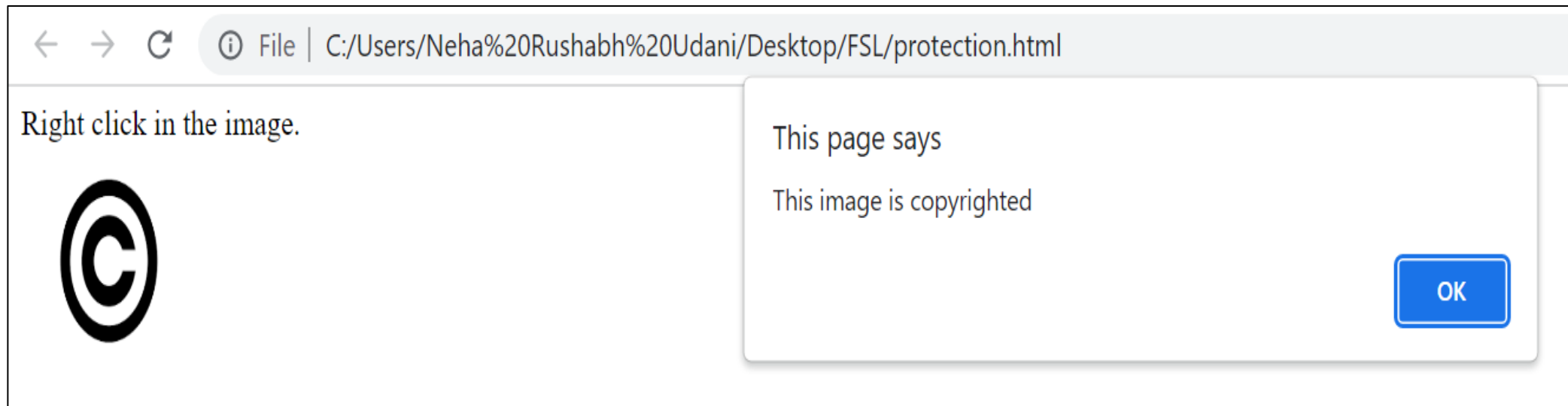
## Moving the message along the status bar

```
<html>  <head>  <title>Javascript ScrollText</title>
  <script language="javascript">
    msg="This is an example of scrolling message";
    spacer="..... ";
    pos=0;
    function ScrollMessage()
    {
      window.status=msg.substring(pos,msg.length)+spacer;
      pos++;
      if(pos>msg.length)
        pos=0;
      window.setTimeout("ScrollMessage()",100);
    }
    ScrollMessage();
  </script>
</head>
<body>
  <p>Scrolling Message Example</p>
  <p>Look at the status line at the bottom of the page</p>
</body>
</html>
```

## Protection web page

```
<html>
<head>
<script language="JavaScript">
function function2()
{
alert("This image is copyrighted")
}
</script>
</head>
<body oncontextmenu="function2()">
<p>Right click in the image.</p>

</body>
</html>
```



## Folding tree menu:

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

For UL

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

::before selector inserts something before the content of each selected element(s).

[Black right pointing triangle.](#)

```
.caret-down::before {
  transform: rotate(90deg);
}
```

The rotate() CSS function defines a transformation that rotates an element around a fixed point on the 2D plane

```
.nested {
  display: none;
}
```

```
.active {
  display: block;
}
```

```
</style></head>
```

## <h2>Tree View</h2>

<p>A tree view 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">Beverages</span>
```

```
    <ul class="nested">
```

```
      <li>Water</li>
```

```
      <li>Coffee</li>
```

```
      <li><span class="caret">Tea</span>
```

```
        <ul class="nested">
```

```
          <li>Black Tea</li>
```

```
          <li>White Tea</li>
```

```
          <li><span class="caret">Green Tea</span>
```

```
            <ul class="nested">
```

```
              <li>Sencha</li>
```

```
              <li>Gyokuro</li>
```

```
              <li>Matcha</li>
```

```
              <li>Pi Lo Chun</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");
  });
}
</script>

</body>
</html>
```

The `addEventListener()` method attaches an event handler to a document.  
The `querySelector()` method returns the first element that matches a CSS selector.

# HTML DOM Document addEventListener()

```
<html>
<body>

<h1>The Document Object</h1>
<h2>The addEventListener() Method</h2>

<p>Click anywhere in the document to display "Hello World!".</p>

<p id="demo"></p>

<script>
document.addEventListener("click", function(){
  document.getElementById("demo").innerHTML = "Hello World!";
});
</script>

</body>
</html>
```

**The Document Object**

**The addEventListener() Method**

Click anywhere in the document to display "Hello World!".



**The Document Object**

**The addEventListener() Method**

Click anywhere in the document to display "Hello World!".

Hello World!

## HTML DOM parentNode Property

```
<html>
<body>
<p>Example list:</p>
<ul>
  <li id="myLI">Coffee</li>
  <li>Tea</li>
</ul>
<p>Click the button to get the node
name of the parent element of the li element in the list.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var x = document.getElementById("myLI").parentNode.nodeName;
  document.getElementById("demo").innerHTML = x;
}
</script>
</body>
</html>
```

Example list:

- Coffee
- Tea

Click the button to get the node name of the parent element of the li element in the list.

Try it

Example list:

- Coffee
- Tea

Click the button to get the node name of the parent element of the li element in the list.

Try it

UL

## HTML DOM Document `querySelector()`

The `querySelector()` method returns the first element that matches a CSS selector.

```
<html>
<body>
<h1>The Document Object</h1>
<h2>The querySelector() Method</h2>

<h3>Add a background color to the first p element:</h3>
<p>This is a p element.</p>
<p>This is a p element.</p>

<script>
document.querySelector("p").style.backgroundColor = "red";
</script>

</body>
</html>
```

### The Document Object

#### The `querySelector()` Method

Add a background color to the first p element:

This is a p element.

This is a p element.

HTML DOM Element `classList`: The `classList` property returns the CSS classnames of an element.

```
<style>
.myStyle {
  background-color: coral;
  padding: 16px;
}
</style>

<body>
<h1>The DOMToken Object</h1>
<h2>The toggle() Method</h2>

<button onclick="myFunction()">Toggle</button>
<p>Click "Toggle" to toggle "myStyle" on and off.</p>

<div id="myDIV">
<p>I am myDIV.</p>
</div>

<script>
function myFunction() {
  document.getElementById("myDIV").classList.toggle("myStyle");
}
</script>
```

## The DOMToken Object

### The toggle() Method

Toggle

Click "Toggle" to toggle "myStyle" on and off.

I am myDIV.

## The DOMToken Object

### The toggle() Method

Toggle

Click "Toggle" to toggle "myStyle" on and off.

I am myDIV.

## Tab Menu:

Tabs are used to navigate and display different content around the website. We use tabs to manage space and to make the website more attractive.

```
<html><head><style>
```

```
/* Style the tab */
```

```
.tab {
```

```
  overflow: hidden;----- With the hidden value, the overflow is clipped, and the rest of the content is hidden:
```

```
  border: 1px solid #ccc;
```

```
  background-color: #f1f1f1;
```

```
}
```

```
/* Style the buttons inside the tab */
```

```
.tab button {
```

```
  background-color: inherit;-----The inherit keyword specifies that a property should inherit its value from its parent element.
```

```
  float: left;
```

```
  border: none;
```

```
  outline: none;
```

```
  cursor: pointer;-----The cursor is a pointer and indicates a link
```

```
  padding: 14px 16px;
```

```
  font-size: 17px;
```

```
}
```

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

```
<h2>Tabs</h2>
```

```
<p>Click on the buttons inside the tabbed menu:</p>
```

```
<div class="tab">
```

```
  <button class="tablinks" onclick="openCity(event, 'London')">London</button>
```

```
  <button class="tablinks" onclick="openCity(event, 'Paris')">Paris</button>
```

```
  <button class="tablinks" onclick="openCity(event, 'Tokyo')">Tokyo</button>
```

```
</div>
```

```
<div id="London" class="tabcontent">
```

```
  <h3>London</h3>
```

```
  <p>London is the capital city of England.</p>
```

```
</div>
```

```
<div id="Paris" class="tabcontent">
```

```
  <h3>Paris</h3>
```

```
  <p>Paris is the capital of France.</p>
```

```
</div>
```

```
<div id="Tokyo" class="tabcontent">
```

```
  <h3>Tokyo</h3>
```

```
  <p>Tokyo is the capital of Japan.</p>
```

```
</div>
```

```
<script>
```



```
function openCity(event, cityName) {  
  var i, tabcontent;  
  tabcontent = document.getElementsByClassName("tabcontent");  
  for (i = 0; i < tabcontent.length; i++)  
  {  
    tabcontent[i].style.display = "none";  
  }  
  document.getElementById(cityName).style.display = "block";  
}  
</script>  
  
</body>  
</html>
```

## Protecting web page

- ✓ JavaScript is designed as an open scripting language. It is not intended to replace proper security measures, and should never be used in place of proper encryption.
- ✓ JavaScript has its own security model, but this is not designed to protect the Web site owner or the data passed between the browser and the server.
- ✓ The security model is designed to protect the user from malicious Web sites, and as a result, it enforces strict limits on what the page author is allowed to do.
- ✓ They may have control over their own page inside the browser, but that is where their abilities end.
- JavaScripts cannot read or write files on users' computers ([File API](#) - not currently supported by many browsers - allows files to be read by scripts if the user specifically chooses to allow it), though they can cause the browser to load remote pages and resources like scripts or images, which the browser may choose to cache locally.
- They are allowed to interact with other pages in a frameset, if those frames originate from the same Web site, but not if they originate from another Web site
- JavaScript cannot be used to set the value attribute of a file input, and will not be allowed to use them to upload files without permission.
- JavaScript cannot read what locations a user has visited by reading them from the location object, although it can tell the browser to jump back or forward any number of steps through the browser history. It cannot see what other Web pages the user has open.
- JavaScript cannot access the cookies or variables from other sites.
- It cannot see when the user interacts with other programs, or other parts of the browser window.
- It cannot open windows out of sight from the user or too small for the user to see, and in most browsers, it cannot close windows that it did not open.

# addEventListener()

The addEventListener() method attaches an event handler to an element.

```
<h1>The Document Object</h1>
<h2>The addEventListener() Method</h2>

<p>Click anywhere in the document to display "Hello World!".</p>

<p id="demo"></p>

<script>
document.addEventListener("click", myFunction);

function myFunction() {
  document.getElementById("demo").innerHTML = "Hello World";
}
</script>
```

element.addEventListener("click", myFunction);

The function to run when the event occurs.

**Event**

Use "click" not "onclick".

## preventDefault() Event Method

- 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

```
<a id="myAnchor" href="https://w3schools.com/">Go to W3Schools.com</a>

<p>The preventDefault() method will prevent the link above from following the URL.</p>

<script>
document.getElementById("myAnchor").addEventListener("click", function(event){
  event.preventDefault()
});
</script>
```

## Example 2

```
<html>
<body>
Try to check this box: <input type="checkbox" id="myCheckbox">
<p>Toggling a checkbox is the default action of clicking on a checkbox. The preventDefault() method
prevents this from happening.</p>
<script>
document.getElementById("myCheckbox").addEventListener("click", function(event){
    event.preventDefault()
});
</script>
</body>
</html>
```

Try to check this box: ☐

Toggling a checkbox is the default action of clicking on a checkbox. The preventDefault() method prevents this from happening.

## Disabling the right mouse button:

```
<html>
<head>
  <title>Example</title>
</head>
<body>
  <h2>Disabling right-clicking on a webpage using JavaScript</h2>
  <p> Right Click is disabled</p>
  <script>
    document.addEventListener("contextmenu", (event) => {
      event.preventDefault();
    });
  </script>
</body>
</html>
```

**Step 1** – Create an HTML page with a script element.

**Step 2** – In the script element, use the `addEventListener` method to attach an event listener function to the `contextmenu` event.

**Step 3** – In the event listener function, use the `preventDefault` method to cancel the default action of the event.

The above JavaScript code uses the `addEventListener` method to attach a `contextmenu` event listener to the document object. The event listener function cancels the default behavior of the right-click by calling the `preventDefault` method of the event object. This will disable right-clicking on the page and prevent the context menu from appearing.

## **concealing email address**    Hide the Email Id in a form using HTML and JavaScript

- 1.Input field: We are going to enter the email id in this field.
- 2.Button: On clicking this button hidden email id will be displayed.
- 3.Output field: Field to display hidden email id.

### **Examples:**

Input :robin\_singh@gmail.com

Output :robin\*\*\*\*\*@gmail.com

Input :geeksforgeeks@gmail.com

Output :geeks\*\*\*\*\*@gmail.com

## Concealingemail

```
<html>
  <head>
    <title></title>
    <script type="text/javascript">
      function test_str(){
        var str = document.getElementById("t1").value ;
        var idx = str.indexOf('@');
        var res = str.replace(str.slice(5, idx), "*".repeat(5));
        document.getElementById("t2").value = res;
      }
    </script>
  </head>
  <body>
    <p>
      Email: <input type="text" placeholder="abc" id="t1"/><br/>
      <input type="button" value="Protect" onclick="test_str()"/><br/>
      Output: <input type="text" id="t2" readonly/>
    </p>
  </body>
</html>
```

Email: neha.more@sbmp.ac.in

Protect

Output: neha.\*\*\*\*\*@sbmp.ac.in

### JavaScript String repeat()

The repeat() method returns a string with a number of copies of a string.

```
let text = "Hello world!";
let result = text.repeat(4);
```

Hello world!Hello world!Hello world!Hello world!



## JavaScript frameworks

- JavaScript frameworks are a collection of libraries containing code written in JavaScript, making life a lot easier for software developers.
- Each JavaScript framework offers pre-built codes for different areas and different purposes in software development, saving time for the developer.

## JavaScript frameworks

JavaScript frameworks provide structure for your code, so in that sense, they're pretty essential to your programming. There are many useful JavaScript frameworks that developers regularly use, and we will cover some of these in this article. A JavaScript framework provides a blueprint, so you have a guide to follow rather than having to start coding your website from scratch.

Most frameworks are open-source, meaning they are constantly being improved by the community that uses them, so they are always up to date. They are by no means set in stone either; you are free to tweak the framework you choose to suit your own website or application.

## Why use a JavaScript framework?

- Developers created JavaScript frameworks to make life easier for themselves. They allow programmers to use the most up-to-date JavaScript features and tools without having to go through the arduous task of coding them from scratch by themselves.
- These frameworks are templates that provide a foundation for software applications. It collects shared resources like libraries, reference documents, images and more and packages them for developers to use. With these frameworks, programmers can add better functionality and more to a web page and website.

## Popular JavaScript Framework libraries

### AngularJS

AngularJS is an open-source framework that came into being in October 2010 and is the oldest available. It's a good one to choose when you're thinking about which framework would be best to go for; brilliantly, it is supported by Google! There are even apps built into cars made by General Motors that have been developed in Angular – it has emerged as a bit of a market leader in JavaScript frameworks. Google's lead Angular developer, Igor Minar, believes that Angular is the most widely used framework because it, more so than others, encourages regular updates and developments.

## **FRONT-END FRAMEWORKS**

### **REACT**

React.js is an efficient and flexible JavaScript library for building user interfaces created by Facebook. Technically, React is a JS library, but it is often discussed as a web framework and is compared to any other open source JavaScript framework.

React makes it easy to create interactive user interfaces because it has predictable JavaScript code that is easy to debug. Furthermore, it provides a REACT component system where blocks of JavaScript code can be written once and reused repeatedly in different parts of the application or even other applications.

### **ANGULAR**

AngularJS is a popular enterprise-level JavaScript framework used for developing large and complex business applications. It is an open-source web framework created by Google and supported by both Google and Microsoft.

### **VUE**

Vue.js is a progressive framework for building user interfaces. It is an up-and-coming framework that helps developers in integrating with other libraries and existing projects. It has an ecosystem of libraries that allow developers to create complex and solid single-page applications.

## **BACK-END FRAMEWORKS**

### **EXPRESS**

Express.js is a flexible, minimalistic, lightweight, and well-supported framework for Node.js applications. It is likely the most popular framework for server-side Node.js applications. Express provides a wide range of HTTP utilities, as well as high-performance speed. It is great for developing a simple, single-page application that can handle multiple requests at the same time.

### **NEXT.JS**

Next.js is a minimalistic framework that allows a JavaScript developer to create a server-side rendering and static web applications using React.js. It is one of the newest and hottest frameworks that takes pride in its ease of use. Many of the problems developers experience while building applications using React.js are solved using Next.js. It has many important features included “out of the box,” and makes development a JavaScript breeze.