ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Lecture 8: Document Object Model (DOM)

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

Document Object Model (DOM)

1. **Introduction**
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

What is the DOM?

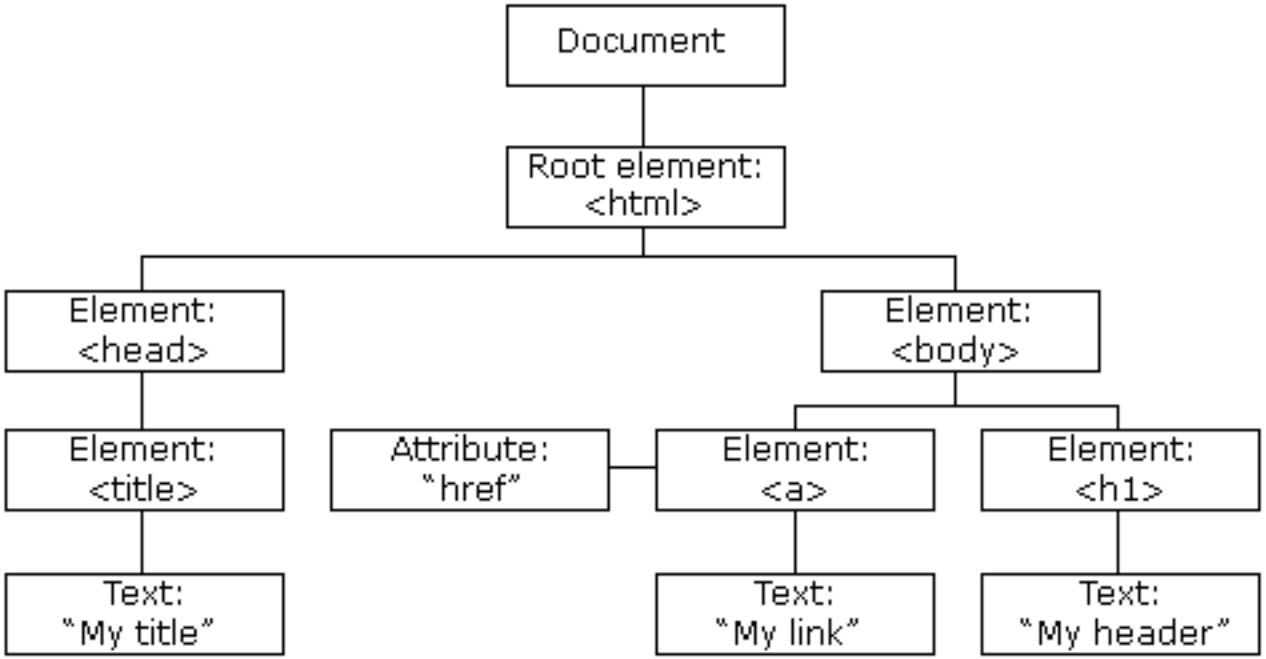
- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- The W3C DOM standard is separated into 3 different parts:
  - Core DOM - standard model for all document types
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents

What is the HTML DOM?

- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements
- In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

The HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:



With the object model, JavaScript gets all the power it needs to create dynamic HTML

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

HTML DOM APIS

- DOM Attributes
- DOM Document
- DOM Element
- DOM Events
- DOM Event Objects
- DOM HTMLCollection
- DOM Location
- DOM Navigator
- DOM Screen
- DOM Style
- DOM Window

Document Object Model (DOM)

1. Introduction
2. **DOM Methods**
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

HTML DOM Methods and HTML DOM properties

- HTML DOM methods are **actions** you can perform (on HTML Elements).
- HTML DOM properties are **values** (of HTML Elements) that you can set or change.

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).

Example

- In the example belows, getElementById is a **method**, while innerHTML is a **property**.

```
<html>
<body>

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

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

My First Page

Hello World!

- The getElementById Method: the most common way to access an HTML element is to use the id of the element.
- The innerHTML Property: The easiest way to get the content of an element is by using the innerHTML property.

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. **DOM Document**
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

JavaScript HTML DOM Document

- The HTML DOM document object is the owner of all other objects in your web page.
- The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Property	**Description**
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
Method	**Description**
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Finding HTML Objects

- The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5.
- Later, in HTML DOM Level 3, more objects, collections, and properties were added.

Property	**Description**	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
…	…	…
…	…	…

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. **DOM Elements**
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

JavaScript HTML DOM Elements

- Often, with JavaScript, you want to manipulate HTML elements.
- To do so, you have to find the elements first. There are several ways to do this:
  - Finding HTML elements by id
  - Finding HTML elements by tag name
  - Finding HTML elements by class name
  - Finding HTML elements by CSS selectors
  - Finding HTML elements by HTML object collections

Finding HTML Element by Id

- The easiest way to find an HTML element in the DOM, is by using the element id.
- This example finds the element with id="intro":

```
<!DOCTYPE html>
<html>

<body>
  <h2>Finding HTML Elements by Id</h2>
  <p id="intro">Hello World!</p>
  <p>This example demonstrates the <b>getElementsById</b> method. </p>
  <p id="demo"></p>
  <script>
    var myElement = document.getElementById("intro");
    document.getElementById("demo").innerHTML = "The text from the intro paragraph is " +
myElement.innerHTML;
  </script>
</body>

</html>
```

Finding HTML Elements by Id

Hello World!

This example demonstrates the **getElementsById** method.

The text from the intro paragraph is Hello World!

Finding HTML Elements by Tag Name

- This example finds all <p> elements:

```
<!DOCTYPE html>
<html>
<body>
  <h2>Finding HTML Elements by Tag Name</h2>

  <p>Hello World!</p>
  <p>This example demonstrates the <b>getElementsByTagName</b> method.</p>

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

  <script>
    var x = document.getElementsByTagName("p");
    document.getElementById("demo").innerHTML =
      'The text in first paragraph (index 0) is: ' + x[0].innerHTML;
  </script>
</body>
</html>
```

Finding HTML Elements by Tag Name

Hello World!

This example demonstrates the **getElementsByTagName** method.

The text in first paragraph (index 0) is: Hello World!

Finding HTML Elements by Class Name

- If you want to find all HTML elements with the same class name, use getElementsByClassName().

- This example returns a list of all elements with class="intro".

```
var x =
document.getElementsByClassName("intro");
```

Finding HTML Elements by CSS Selectors

- If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

- This example returns a list of all <p> elements with class="intro".

```
var x =
document.querySelectorAll("p.intro");
```

Finding HTML Elements by HTML Object Collections

```
<html>
<body>
  <h2>Finding HTML Elements Using document.forms</h2>
  <form id="frm1" action="/action_page.php">
    First name: <input type="text" name="fname" value="Donald"><br>
    Last name: <input type="text" name="lname" value="Duck"><br><br>
    <input type="submit" value="Submit">
  </form>
  <p>Click "Try it" to display the value of each element in the form.</p>
  <button onclick="myFunction()">Try it</button>
  <p id="demo"></p>
  <script>
    function myFunction() {
      var x = document.forms["frm1"];
      var text = "";
      var i;
      for (i = 0; i < x.length; i++) {
        text += x.elements[i].value + "<br>";
      }
      document.getElementById("demo").innerHTML = text;
    }
  </script>
</body>
</html>
```

Finding HTML Elements by HTML Object Collections

```
<html>
<body>
  <h2>Finding HTM
  <form id="frm1"
    First name: <in
    Last name: <in
    <input type="s
  </form>
  <p>Click "Try it" t
  <button onclick=
  <p id="demo"></
  <script>
    function myFu
      var x = docu
      var text = "";
      var i;
      for (i = 0; i <
        text += x.e
      }
      document.ge
    }
  </script>
</body>
</html>
```

Finding HTML Elements Using document.forms

First name: Donald
Last name: Duck

Submit

Click "Try it" to display the value of each element in the form.

Try it

Finding HTML Elements Using document.forms

First name: Donald
Last name: Duck

Submit

Click "Try it" to display the value of each element in the form.

Try it

Donald
Duck
Submit

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. **DOM - Changing HTML**
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

Changing HTML

- The HTML DOM allows JavaScript to change the content of HTML elements:
  - Changing the HTML Output Stream
  - Changing HTML Content
  - Changing the Value of an Attribute

Changing the HTML Output Stream

- In JavaScript, document.write() can be used to write directly to the HTML output stream:

```
<!DOCTYPE html>
<html>
<body>

<script>
document.write(Date());
</script>

</body>
</html>
```

Date: Wed May 12 2021 09:06:41 GMT+0700 (Indochina Time)

Changing HTML Content

- The easiest way to modify the content of an HTML element is by using the innerHTML property.
- To change the content of an HTML element, use this syntax:

```
document.getElementById(id).innerHTML = new HTML
```

- This example changes the content of a <p> element:

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

JavaScript can Change HTML

New text!

The paragraph above was changed by a script.

Changing HTML Content

- This example changes the content of an <h1> element:

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id01">Old Heading</h1>

<script>
var element = document.getElementById("id01");
element.innerHTML = "New Heading";
</script>

</body>
</html>
```

New Heading

JavaScript changed "Old Heading" to "New Heading".

Changing the Value of an Attribute

- To change the value of an HTML attribute, use this syntax:

```
document.getElementById(id).attribute = new value
```

- This example changes the value of the src attribute of an <img> element:

```
<!DOCTYPE html>
<html>
<body>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

Changing the Value of an Attribute

- To change the value of an HTML attribute, use this syntax:

```
document.getElementById(id).attribute = new value
```

- This example changes the value of the src attribute of an <img> element:

```
<!DOCTYPE html>
<html>
<body>

<img id

<script
documer
</scrip

</body>
</html>
```

The original image was smiley.gif, but the script changed it to landscape.jpg

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. **DOM - Changing CSS**
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

Changing CSS

- The HTML DOM allows JavaScript to change the style of HTML elements.
- To change the style of an HTML element, use this syntax:

```
document.getElementById(id).style.property = new style
```

- The following example changes the style of a <p> element:

```
<html>
<body>
<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

Changing CSS

- The HTML DOM allows JavaScript to change the style of HTML elements.
- To change the style of an HTML element, use this syntax:

```
document.getElementById(id).style.property = new style
```

- The following example changes the style of a <p> element:

```
<html>
<body>
<p id="p2">Hello World!</p>

<script>
document.getElementBy
</script>

<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

Hello World!

Hello World!

The paragraph above was changed by a script.

Using Events

- The HTML DOM allows you to execute code when an event occurs.
  - Events are generated by the browser when "things happen" to HTML elements:
  - An element is clicked on
  - The page has loaded
  - Input fields are changed

Using Events - Example

- This example changes the style of the HTML element with id="id1", when the user clicks a button:

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

</body>
</html>
```

My Heading 1

Click Me!

My Heading 1

Click Me!

Using Events – Example 2

```
<!DOCTYPE html>
<html>
<body>
  <p id="p1">
    This is a text.
    This is a text.
    This is a text.
    This is a text.
  </p>
  <input type="button" value="Hide text"
onclick="document.getElementById('p1').style.visibility='hidden'">
  <input type="button" value="Show text"
onclick="document.getElementById('p1').style.visibility='visible'">
</body>
</html>
```

This is a text. This is a text. This is a text. This is a text.

Hide text Show text

Hide text Show text

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. **DOM Animation**
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

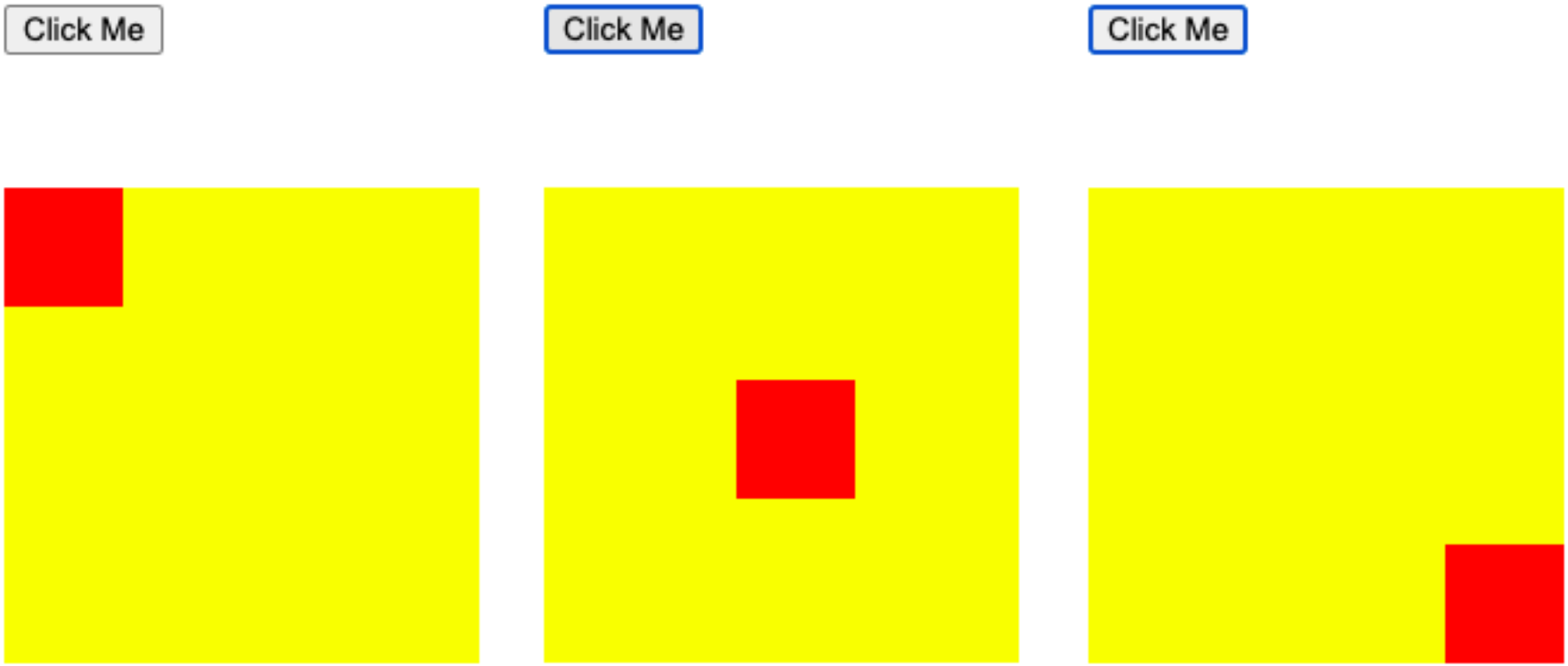
JavaScript HTML DOM Animation

- JavaScript animations are done by programming gradual changes in an element's style.
- The changes are called by a timer. When the timer interval is small, the animation looks continuous.
- The basic code is:

```
var id = setInterval(frame, 5);

function frame() {
  if (/* test for finished */) {
    clearInterval(id);
  } else {
    /* code to change the element style */
  }
}
```

Example



Example

```
<!DOCTYPE html>
<html>
<style>
   #container {
      width: 200px;
      height: 200px;
      position: relative;
      background: yellow;
   }
   #animate {
      width: 50px;
      height: 50px;
      position: absolute;
      background-color: red;
   }
</style>
<body>
   <p><button onclick="myMove()">Click Me</button></p>
   <div id="container">
      <div id="animate"></div>
   </div>
```

Example

```
<script>
  var id = null;
  function myMove() {
    var elem = document.getElementById("animate");
    var pos = 0;
    clearInterval(id);
    id = setInterval(frame, 5);
    function frame() {
      if (pos == 150) {
        clearInterval(id);
      } else {
        pos++;
        elem.style.top = pos + "px";
        elem.style.left = pos + "px";
      }
    }
  }
</script>
</body>
</html>
```

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. **DOM Events**
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

JavaScript HTML DOM Events

- A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.
- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute

```
onclick=JavaScript
```

- Examples of HTML events:
  - When a user clicks the mouse
  - When a web page has loaded
  - When an image has been loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted
  - When a user strokes a key

Reacting to Events

- In this example, the content of the <h1> element is changed when a user clicks on it:

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="this.innerHTML = 'Ooops!'">Click on this text!</h1>

</body>
</html>
```

Click on this text! **Ooops!**

Reacting to Events

- In this example, a function is called from the event handler:

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
  id.innerHTML = "Ooops!";
}
</script>

</body>
</html>
```

Click on this text! **Ooops!**

HTML Event Attributes

- To assign events to HTML elements you can use event attributes.
- In the example, a function named displayDate will be executed when the button is clicked.

```
<!DOCTYPE html>
<html>
<body>
  <p>Click the button to display the date.</p>
  <button onclick="displayDate()">The time is?</button>
  <script>
    function displayDate() {
      document.getElementById("demo").innerHTML = Date();
    }
  </script>
  <p id="demo"></p>
</body>
</html>
```

Click the button to display the date.

The time is?

Click the button to display the date.

The time is?

Wed May 12 2021 09:43:10 GMT+0700 (Indochina Time)

Assign Events Using the HTML DOM

- The HTML DOM allows you to assign events to HTML elements using JavaScript:
- In the example, a function named displayDate is assigned to an HTML element with the id="myBtn".
- The function will be executed when the button is clicked

```
<!DOCTYPE html>
<html>
<body>
  <p>Click "Try it" to execute the displayDate() function.</p>
  <button id="myBtn">Try it</button>
  <p id="demo"></p>
  <script>
    document.getElementById("myBtn").onclick = displayDate;
    function displayDate() {
      document.getElementById("demo").innerHTML = Date();
    }
  </script>
</body>
</html>
```

Assign Events Using the HTML DOM

- The HTML DOM allows you to assign events to HTML elements using JavaScript:
- In the example, a function named displayDate is assigned to an HTML element with the id="myBtn".
- The function will be executed when the button is clicked

```
<!DOCTYPE html>
<html>
<body>
  <p>Click "Try it"
  <button id="myE
  <p id="demo"></p>
  <script>
    document.get
    function displa
      document.g
    }
  </script>
</body>
</html>
```

Click "Try it" to execute the displayDate() function.

Try it

Click "Try it" to execute the displayDate() function.

Try it

Wed May 12 2021 09:45:17 GMT+0700 (Indochina Time)

The onload and onunload Events

- The onload and onunload events are triggered when the user enters or leaves the page.
- The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.
- The onload and onunload events can be used to deal with cookies.

The onload and onunload Events

```
<!DOCTYPE html>
<html>
<body onload="checkCookies()">
  <p id="demo"></p>
  <script>
    function checkCookies() {
      var text = "";
      if (navigator.cookieEnabled == true) {
        text = "Cookies are enabled.";
      } else {
        text = "Cookies are not enabled.";
      }
      document.getElementById("demo").innerHTML = text;
    }
  </script>
</body>
</html>
```

Cookies are enabled.

The onchange Event

- The onchange event is often used in combination with validation of input fields.
- Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function myFunction() {
      var x = document.getElementById("fname");
      x.value = x.value.toUpperCase();
    }
  </script>
</head>
<body>
  Enter your name: <input type="text" id="fname" onchange="myFunction()">
  <p>When you leave the input field, a function is triggered which transforms the input text to upper case.</p>
</body>
</html>
```

The onchange Event

- The onchange event is often used in combination with validation of input fields.
- Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function myFunction() {
      var x = document.getElementById("fname");
```

Enter your name: MY TEST

When you leave the input field, a function is triggered which transforms the input text to upper case.

```
<body>
  Enter your name: <input type="text" id="fname" onchange="myFunction()">
  <p>When you leave the input field, a function is triggered which transforms the input text to upper case.</p>
</body>
</html>
```

The onmouseover and onmouseout Events

- The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element:

```
<!DOCTYPE html>
<html>
<body>
  <div onmouseover="mOver(this)" onmouseout="mOut(this)"
    style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
    Mouse Over Me</div>
  <script>
    function mOver(obj) {
      obj.innerHTML = "Thank You"
    }
    function mOut(obj) {
      obj.innerHTML = "Mouse Over Me"
    }
  </script>
</body>
</html>
```

The onmousedown, onmouseup and onclick Events

- The onmousedown, onmouseup, and onclick events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

The onmousedown, onmouseup and onclick Events

```
<!DOCTYPE html>
<html>
<body>
  <div onmousedown="mDown(this)" onmouseup="mUp(this)"
    style="background-color:#D94A38;width:90px;height:20px;padding:40px;">
    Click Me</div>
  <script>
    function mDown(obj) {
      obj.style.backgroundColor = "#1ec5e5";
      obj.innerHTML = "Release Me";
    }
    function mUp(obj) {
      obj.style.backgroundColor = "#D94A38";
      obj.innerHTML = "Thank You";
    }
  </script>
</body>
</html>
```

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. **DOM EventListener**
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

JavaScript HTML DOM EventListener

- The addEventListener() method attaches an event handler to the specified element.
- The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
- You can add many event handlers to one element.
- You can add many event handlers of the same type to one element, i.e two "click" events.
- You can add event listeners to any DOM object not only HTML elements. i.e the window object.
- The addEventListener() method makes it easier to control how the event reacts to bubbling.
- When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.
- You can easily remove an event listener by using the removeEventListener() method.

JavaScript HTML DOM EventListener

- Syntax

```
element.addEventListener(event, function, useCapture);
```

- The first parameter is the type of the event (like "click" or "mousedown" or any other HTML DOM Event.)
- The second parameter is the function we want to call when the event occurs.
- The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.
-

Add an Event Handler to an Element

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript addEventListener()</h2>
  <p>This example uses the addEventListener() method to attach a click event to a button.</p>
  <button id="myBtn">Try it</button>
  <script>
    document.getElementById("myBtn").addEventListener("click", function () {
      alert("Hello World!");
    });
  </script>
</body>
</html>
```

JavaScript addEventListener()

This example uses the addEventListener() method to attach a click event to a button.

Try it

www.w3schools.com says

Hello World!

OK

Add an Event Handler to an Element

- You can also refer to an external "named" function:

```
element.addEventListener("click", function(){ alert("Hello World!");});
```

```
element.addEventListener("click", myFunction);

function myFunction() {
  alert ("Hello World!");
}
```

Add Many Event Handlers to the Same Element

- The addEventListener() method allows you to add many events to the same element, without overwriting existing events:

```
element.addEventListener("click", myFunction);
element.addEventListener("click", mySecondFunction);
```

- You can add events of different types to the same element:

```
element.addEventListener("mouseover", myFunction);
element.addEventListener("click", mySecondFunction);
element.addEventListener("mouseout", myThirdFunction);
```

Add an Event Handler to the window Object

- The addEventListener() method allows you to add event listeners on any HTML DOM object such as HTML elements, the HTML document, the window object, or other objects that support events, like the xmlHttpRequest object.
- Example: Add an event listener that fires when a user resizes the window:

```
window.addEventListener("resize", function(){
  document.getElementById("demo").innerHTML = sometext;
});
```

Passing Parameters

- When passing parameter values, use an "anonymous function" that calls the specified function with the parameters:

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript addEventListener()</h2>
  <p>Click the button to perform a calculation.</p>
  <button id="myBtn">Try it</button>
  <p id="demo"></p>
  <script>
    var p1 = 5;
    var p2 = 7;
    document.getElementById("myBtn").addEventListener("click", function () {
      myFunction(p1, p2);
    });

    function myFunction(a, b) {
      var result = a * b;
      document.getElementById("demo").innerHTML = result;
    }
  </script>
</body>
</html>
```

Passing Parameters

- When passing parameter values, use an "anonymous function" that calls the specified function with the parameters:

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript addEventListener()</h2>
  <p>Click the button to perform a calculation.</p>
  <button id="myBtn">Try it</button>
  <p id="demo"></p>
  <script>
    var p1 = 5;
    var p2 = 7;
    document.getElementById("myBtn").addEvent
      myFunction(p1, p2);
    });

    function myFunction(a, b) {
      var result = a * b;
      document.getElementById("demo").innerHT
    }
  </script>
</body>
</html>
```

JavaScript addEventListener()

Click the button to perform a calculation.

Try it

JavaScript addEventListener()

Click the button to perform a calculation.

Try it

35

Event Bubbling or Event Capturing?

- There are two ways of event propagation in the HTML DOM, bubbling and capturing.
- Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a element, and the user clicks on the <p> element, which element's "click" event should be handled first?
- In *bubbling* the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the element's click event.
- In *capturing* the outer most element's event is handled first and then the inner: the element's click event will be handled first, then the <p> element's click event.
- With the addEventListener() method you can specify the propagation type by using the "useCapture" parameter:

```
addEventListener(event, function, useCapture);
```

- The default value is false, which will use the bubbling propagation, when the value is set to true, the event uses the capturing propagation.

Event Bubbling

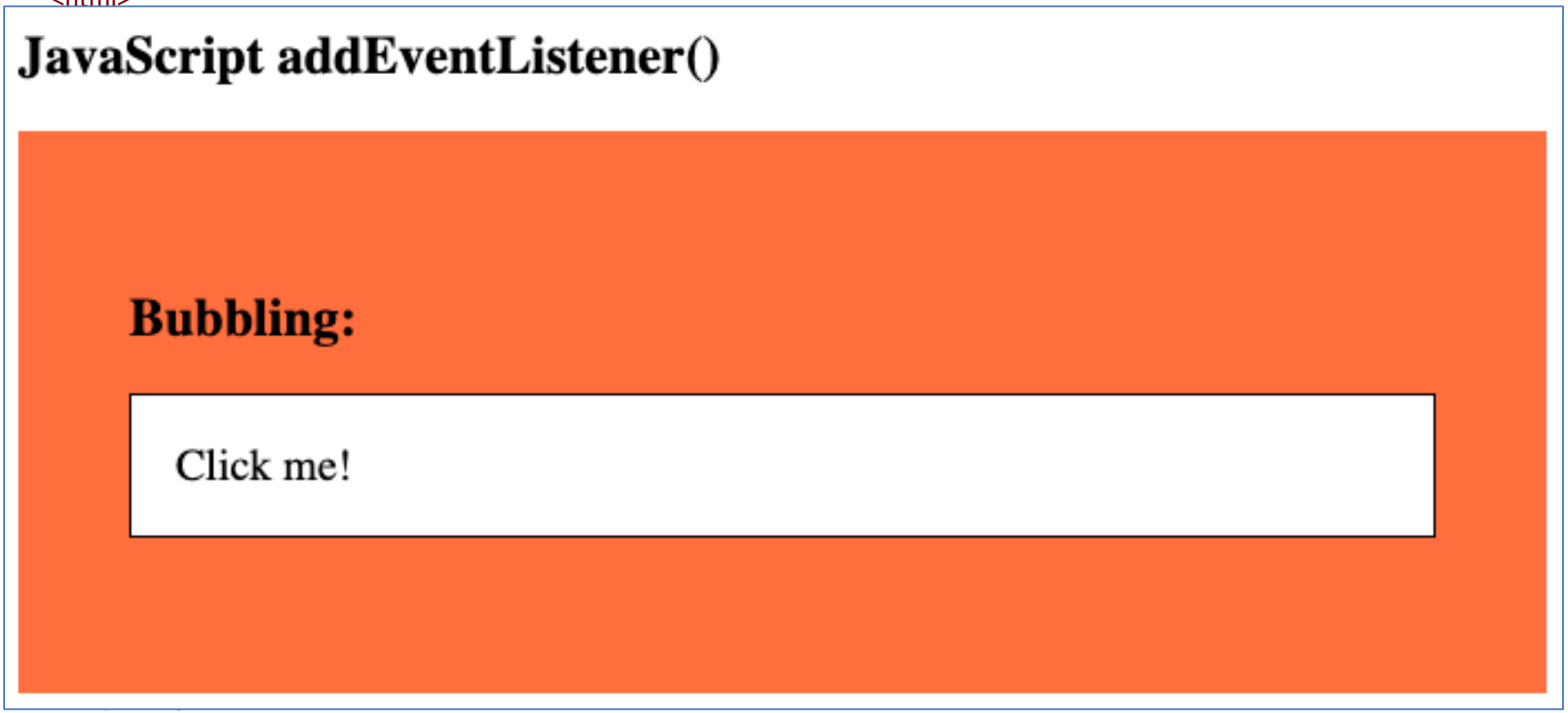
```
document.getElementById("myP").addEventListener("click", myFunction, true);
document.getElementById("myDiv").addEventListener("click", myFunction, true);
```

```
<html>
<head>
  <style>
    #myDiv1 {background-color: coral; padding: 50px;}
    #myP1 {
      background-color: white; font-size: 20px; border: 1px solid; padding: 20px;
    }
  </style>
</head>
<body>
  <h2>JavaScript addEventListener()</h2>
  <div id="myDiv1">
    <h2>Bubbling:</h2>
    <p id="myP1">Click me!</p>
  </div><br>
  <script>
    document.getElementById("myP1").addEventListener("click", function () {
      alert("You clicked the white element!");
    }, false);
    document.getElementById("myDiv1").addEventListener("click", function () {
      alert("You clicked the orange element!");
    }, false);
  </script>
</body>
</html>
```

Event Bubbling

```
document.getElementById("myP").addEventListener("click", myFunction, true);
document.getElementById("myDiv").addEventListener("click", myFunction, true);
```

```
<html>
```

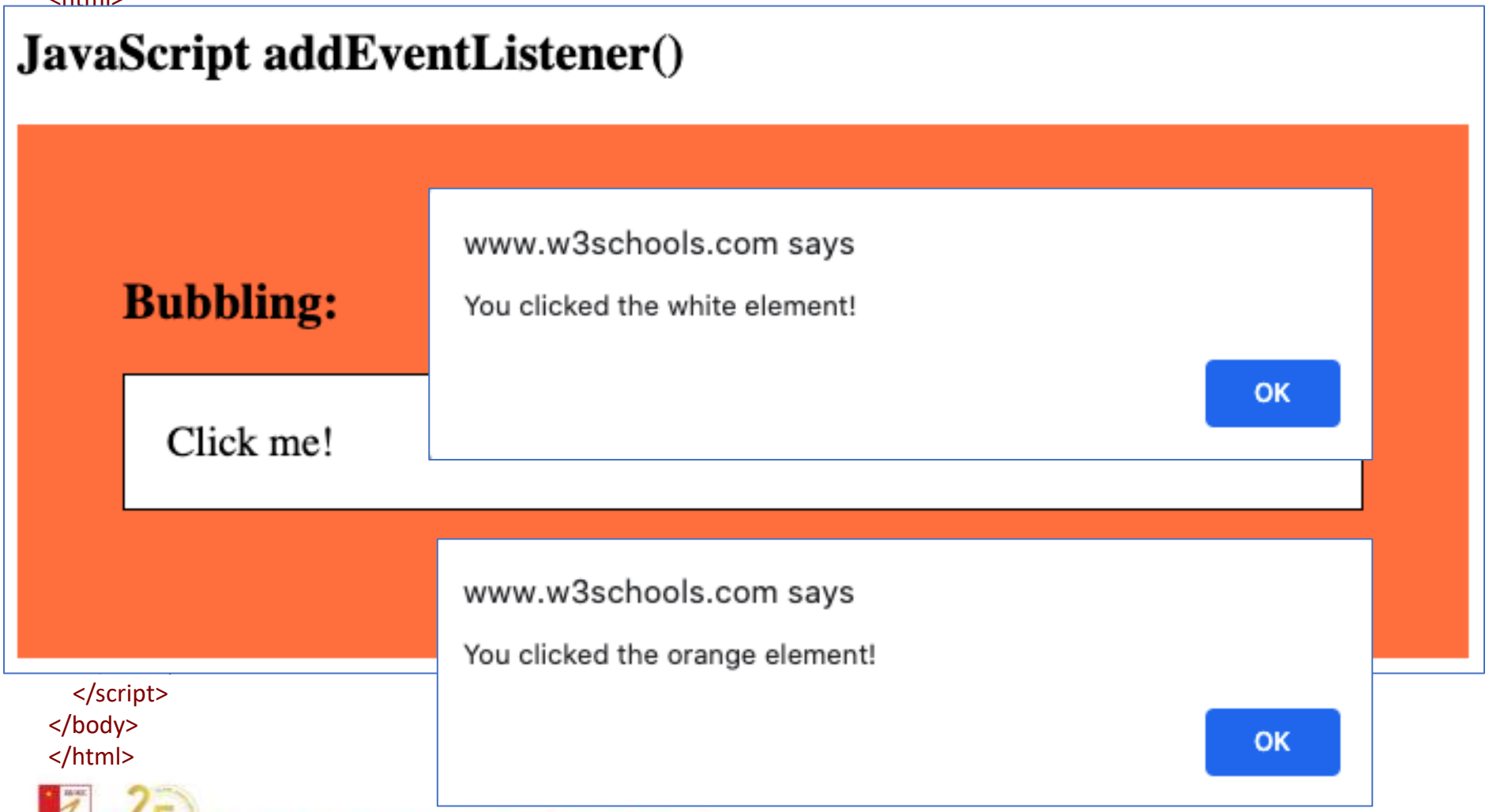



```
  </script>
</body>
</html>
```

Event Bubbling

```
document.getElementById("myP").addEventListener("click", myFunction, true);
document.getElementById("myDiv").addEventListener("click", myFunction, true);
```

```
<html>
```




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

The removeEventListener() method

- The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

```
element.removeEventListener("mousemove", myFunction);
```

```
<h2>JavaScript removeEventListener()</h2>
<div id="myDIV">

  <p>This div element has an onmousemove event handler that displays a random number every time you move your mouse inside this
orange field.</p>
  <p>Click the button to remove the div's event handler.</p>
  <button onclick="removeHandler()" id="myBtn">Remove</button>
</div>
<p id="demo"></p>
<script>
  document.getElementById("myDIV").addEventListener("mousemove", myFunction);
  function myFunction() {
    document.getElementById("demo").innerHTML = Math.random();
  }
  function removeHandler() {
    document.getElementById("myDIV").removeEventListener("mousemove", myFunction);
  }
</script>
```

The removeEventListener() method

- The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

```
element.removeEventListener("mousemove", myFunction);
```

```
<h2>JavaScript re
<div id="myDIV">
  <p>This div ele
orange field.</p>
  <p>Click the bu
  <button onclic
</div>
<p id="demo">
<script>
  document.getE
  function myFu
    document.g
  }
  function remo
    document.g
  }
</script>
```

JavaScript removeEventListener()

This div element has an onmousemove event handler that displays a random number every time you move your mouse inside this orange field.

Click the button to remove the div's event handler.

Remove

0.2867152859163884

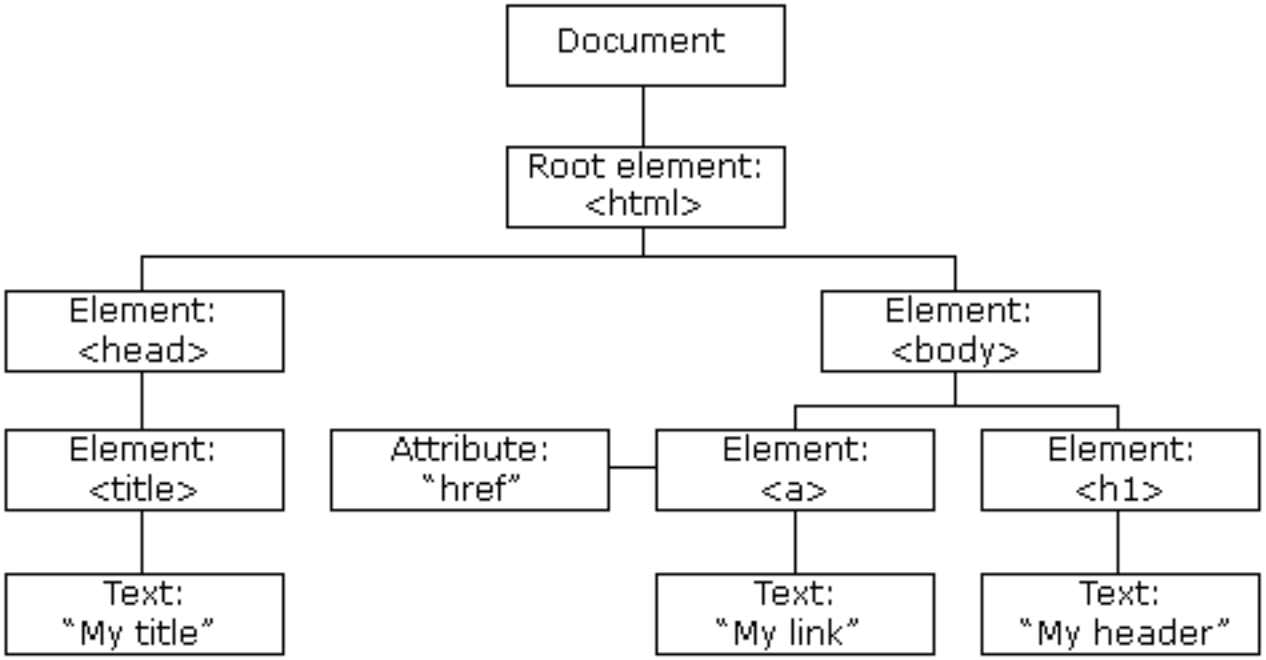
Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. **DOM Navigation**
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

DOM Navigation

- With the HTML DOM, you can navigate the node tree using node relationships.
- According to the W3C HTML DOM standard, everything in an HTML document is a node:
  - The entire document is a document node
  - Every HTML element is an element node
  - The text inside HTML elements are text nodes
  - Every HTML attribute is an attribute node (deprecated)
  - All comments are comment nodes
- With the HTML DOM, all nodes in the node tree can be accessed by JavaScript.
- New nodes can be created, and all nodes can be modified or deleted.

DOM Navigation



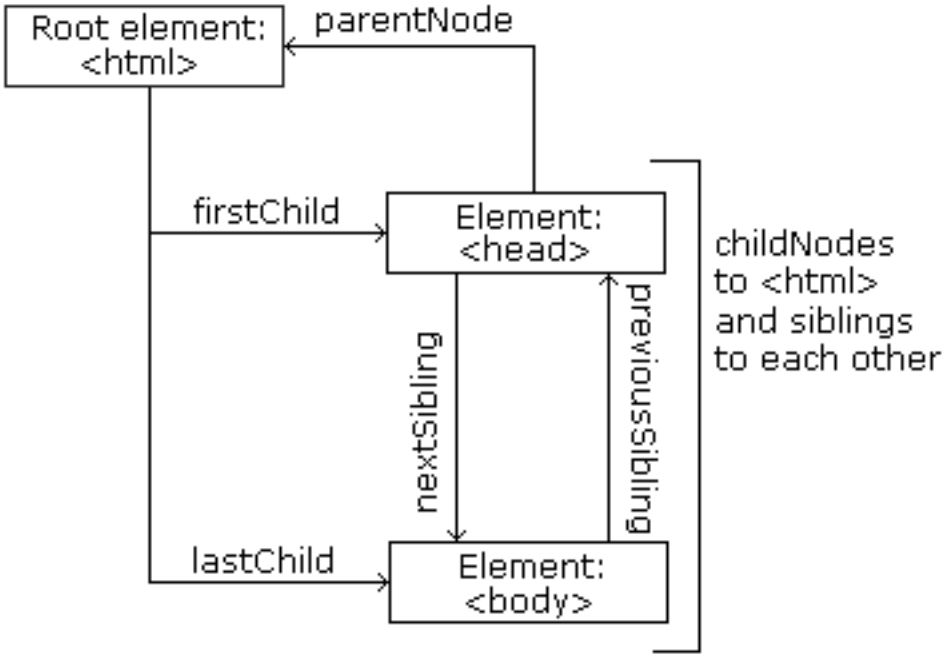
Node Relationships

- The nodes in the node tree have a hierarchical relationship to each other.
- The terms parent, child, and sibling are used to describe the relationships.
- In a node tree, the top node is called the root (or root node)
- Every node has exactly one parent, except the root (which has no parent)
- A node can have a number of children
- Siblings (brothers or sisters) are nodes with the same parent

Node Relationships

- From the HTML above you can read:
  - <html> is the root node
  - <html> has no parents
  - <html> is the parent of <head> and <body>
  - <head> is the first child of <html>
  - <body> is the last child

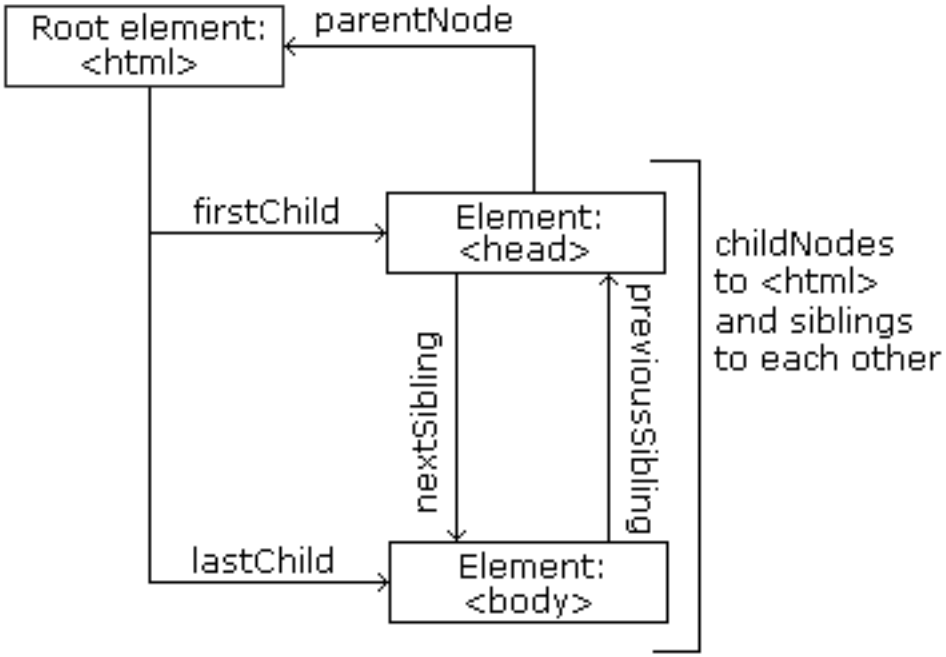
```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```



Node Relationships

- And
  - <head> has one child: <title>
  - <title> has one child (a text node): "DOM Tutorial"
  - <body> has two children: <h1> and <p>
  - <h1> has one child: "DOM Lesson one"
  - <p> has one child: "Hello world!"
  - <h1> and <p> are siblings

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```



Navigating Between Nodes

- You can use the following node properties to navigate between nodes with JavaScript:
  - parentNode
  - childNodes[*nodenumber*]
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling

Child Nodes and Node Values

- A common error in DOM processing is to expect an element node to contain text.
- Example

```
<title id="demo">DOM Tutorial</title>
```

- The element node <title> (in the example above) does **not** contain text.
- It contains a **text node** with the value "DOM Tutorial".
- The value of the text node can be accessed by the node's innerHTML property:

```
var myTitle = document.getElementById("demo").innerHTML;
```

Child Nodes and Node Values

- Accessing the innerHTML property is the same as accessing the nodeValue of the first child:

```
var myTitle = document.getElementById("demo").firstChild.nodeValue;
```

- Accessing the first child can also be done like this:

```
var myTitle = document.getElementById("demo").childNodes[0].nodeValue;
```

Example

- The following example retrieves the text of an <h1> element and copies it into a <p> element:

```
<html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02">Hello!</p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").childNodes[0].nodeValue;
</script>

</body>
</html>
```

My First Page

My First Page

DOM Root Nodes

- There are two special properties that allow access to the full document:
  - document.body - The body of the document
  - document.documentElement - The full document

```
<html>
<body>

<p>Hello World!</p>
<div>
<p>The DOM is very useful!</p>
<p>This example demonstrates the <b>document.body</b> property.</p>
</div>

<script>
alert(document.body.innerHTML);
</script>

</body>
</html>
```

DOM Root Nodes

- There are two special properties that allow access to the full document:
  - document.body - The body of the document
  - do

```
<html>
<body>

<p>Hello
<div>
<p>The D
<p>This                                                    rty.</p>
</div>

<script>
alert(do
</script

</body>
</html>
```

```
<p>Hello World!</p>

<div>
<p>The DOM is very useful!</p>
<p>This example demonstrates the <b>document.body</b>
property.</p>
</div>

<script>
```

OK

The nodeName Property

- The nodeName property specifies the name of a node.
  - nodeName is read-only
  - nodeName of an element node is the same as the tag name
  - nodeName of an attribute node is the attribute name
  - nodeName of a text node is always #text
  - nodeName of the document node is always #document

```
<html>
<body>
  <h1 id="id01">My First Page</h1>
  <p id="id02"></p>
  <script>
    document.getElementById("id02").innerHTML =
                                    document.getElementById("id01").nodeName;
  </script>
</body>
</html>
```

My First Page

H1

The nodeValue Property

- The nodeValue property specifies the value of a node.
  - nodeValue for element nodes is null
  - nodeValue for text nodes is the text itself
  - nodeValue for attribute nodes is the attribute value

The nodeType Property

- The nodeType property is read only. It returns the type of a node.
- The most important nodeType properties are:

Node	Type	Example
ELEMENT_NODE	1	<h1 class="heading">W3Schools</h1>
ATTRIBUTE_NODE	2	class = "heading" (deprecated)
TEXT_NODE	3	W3Schools
COMMENT_NODE	8	<!-- This is a comment -->
DOCUMENT_NODE	9	The HTML document itself (the parent of <html>)
DOCUMENT_TYPE_NODE	10	<!Doctype html>

The nodeType Property

- The nodeType property is read only. It returns the type of a node.

```
<html>
<body>
  <h1 id="id01">My First Page</h1>
  <p id="id02"></p>

  <script>
    document.getElementById("id02").innerHTML =
                    document.getElementById("id01").nodeType;
  </script>
</body>
</html>
```

My First Page

1

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. **DOM Elements (Nodes)**
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

JavaScript HTML DOM Elements (Nodes)

- Creating New HTML Elements (Nodes)
- Creating new HTML Elements - insertBefore()
- Removing Existing HTML Elements
- Removing a Child Node
- Replacing HTML Elements

Creating New HTML Elements (Nodes)

- To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
<html>
<body>
  <div id="div1">
    <p id="p1">This is a paragraph.</p>
    <p id="p2">This is another paragraph.</p>
  </div>
  <script>
    var para = document.createElement("p");
    var node = document.createTextNode("This is new.");
    para.appendChild(node);
    var element = document.getElementById("div1");
    element.appendChild(para);
  </script>
</body>
</html>
```

This is a paragraph.

This is another paragraph.

This is new.

Example Explained

- This code creates a new <p> element:

```
var para = document.createElement("p");
```

- To add text to the <p> element, you must create a text node first. This code creates a text node:

```
var node = document.createTextNode("This is a new paragraph.");
```

- Then you must append the text node to the <p> element:

```
para.appendChild(node);
```

- Finally you must append the new element to an existing element.
- This code finds an existing element:

```
var element = document.getElementById("div1");
```

- This code appends the new element to the existing element:

```
element.appendChild(para);
```

Creating new HTML Elements - insertBefore()

- The appendChild() method in the previous example, appended the new element as the last child of the parent.
- If you don't want that you can use the insertBefore() method:

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
var child = document.getElementById("p1");
element.insertBefore(para, child);
</script>
```

This is new.

This is a paragraph.

This is another paragraph.

Removing Existing HTML Elements

- To remove an HTML element, use the remove() method:

```
<div>
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var elmnt = document.getElementById("p1");
elmnt.remove();
</script>
```

This is a paragraph.

This is another paragraph.

Remove Element

Removing a Child Node

- For browsers that does not support the remove() method, you have to find the parent node to remove an element:

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

This is another paragraph.

Replacing HTML Elements

- To replace an element to the HTML DOM, use the replaceChild() method:

This is new.

This is another paragraph.

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.replaceChild(para, child);
</script>
```

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. **DOM Collections**
13. DOM Node Lists
14. Browser Object Model (BOM)
15. Jquery HTML DOM

JavaScript HTML DOM Collections

- The getElementsByTagName() method returns an HTMLCollection object.
- An HTMLCollection object is an array-like list (collection) of HTML elements.
- The following code selects all <p> elements in a document:

```
var x = document.getElementsByTagName("p");
```

- The elements in the collection can be accessed by an index number.
- To access the second <p> element you can write:

```
y = x[1];
```

Example

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript HTML DOM</h2>
  <p>Hello World!</p>
  <p>Hello Norway!</p>
  <p id="demo"></p>

  <script>
    var myCollection = document.getElementsByTagName("p");
    document.getElementById("demo").innerHTML =
      "The innerHTML of the second paragraph is: " +
      myCollection[1].innerHTML;
  </script>
</body>
</html>
```

JavaScript HTML DOM

Hello World!

Hello Norway!

The innerHTML of the second paragraph is: Hello Norway!

HTMLCollection Length

- The length property defines the number of elements in an HTMLCollection:

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript HTML DOM</h2>
  <p>Hello World!</p>
  <p>Hello Norway!</p>
  <p id="demo"></p>

  <script>
    var myCollection = document.getElementsByTagName("p");
    document.getElementById("demo").innerHTML =
      "This document contains " + myCollection.length + " paragraphs.";
  </script>
</body>
</html>
```

JavaScript HTML DOM

Hello World!

Hello Norway!

This document contains 3 paragraphs.

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. **DOM Node Lists**
14. Browser Object Model (BOM)
15. Jquery HTML DOM

DOM Node Lists

- A NodeList object is a list (collection) of nodes extracted from a document.
- A NodeList object is almost the same as an HTMLCollection object.
- Some (older) browsers return a NodeList object instead of an HTMLCollection for methods like getElementsByClassName().
- All browsers return a NodeList object for the property childNodes.
- Most browsers return a NodeList object for the method querySelectorAll().
- The following code selects all <p> nodes in a document:

```
var myNodeList = document.querySelectorAll("p");
```

Example

```
<html>
<body>
  <h2>JavaScript HTML DOM!</h2>
  <p>Hello World!</p>
  <p>Hello Norway!</p>
  <p id="demo"></p>

  <script>
    var myNodelist = document.querySelectorAll("p");
    document.getElementById("demo").innerHTML =
      "The innerHTML of the second paragraph is: " +
      myNodelist[1].innerHTML;
  </script>
</body>
</html>
```

JavaScript HTML DOM!

Hello World!

Hello Norway!

The innerHTML of the second paragraph is: Hello Norway!

HTML DOM Node List Length

- The length property defines the number of nodes in a node list:

```
<html>
<body>
  <h2>JavaScript HTML DOM!</h2>
  <p>Hellow World!</p>
  <p>Hellow Norway!</p>
  <p id="demo"></p>

  <script>
    var myNodelist = document.querySelectorAll("p");
    document.getElementById("demo").innerHTML =
      "This document contains " + myNodelist.length + " paragraphs.";
  </script>
</body>
</html>
```

JavaScript HTML DOM!

Hellow World!

Hellow Norway!

This document contains 3 paragraphs.

HTML DOM Node List Length

❖ Change the color of all <p> elements in a node list:

```
<html>
<body>
  <h2>JavaScript HTML DOM!</h2>
  <p>Hello World!</p>
  <p>Hello Norway!</p>
  <p>Click the button to change the color of all p elements.</p>
  <button onclick="myFunction()">Try it</button>
  <script>
    function myFunction() {
      var myNodelist = document.querySelectorAll("p");
      var i;
      for (i = 0; i < myNodelist.length; i++) {
        myNodelist[i].style.color = "red";
      }
    }
  </script>
</body>
</html>
```

HTML DOM Node List Length

❖ Change the color of all <p> elements in a node list:

```
<html>
<body>
  <h2>JavaScript HTML DOM!</h2>
  <p>Hello World!</p>
  <p>Hello Norway!</p>
  <p>Click the button to change the color c
  <button onclick="myFunction()">Try it</l
  <script>
    function myFunction() {
      var myNodelist = document.querySe
      var i;
      for (i = 0; i < myNodelist.length; i++) {
        myNodelist[i].style.color = "red";
      }
    }
  </script>
</body>
</html>
```

JavaScript HTML DOM!

Hello World!

Hello Norway!

Click the button to change the color of all p elements.

Try it

JavaScript HTML DOM!

Hello World!

Hello Norway!

Click the button to change the color of all p elements.

Try it

The Difference Between an HTMLCollection and a NodeList

- An HTMLCollection is a collection of HTML elements.
- A NodeList is a collection of document nodes.
- A NodeList and an HTML collection is very much the same thing.
- Both an HTMLCollection object and a NodeList object is an array-like list (collection) of objects.
- Both have a length property defining the number of items in the list (collection).
- Both provide an index (0, 1, 2, 3, 4, ...) to access each item like an array.
- HTMLCollection items can be accessed by their name, id, or index number.
- NodeList items can only be accessed by their index number.
- Only the NodeList object can contain attribute nodes and text nodes.

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. **Browser Object Model (BOM)**
15. Jquery HTML DOM

The Browser Object Model (BOM)

- There are no official standards for the **B**rowser **O**bject **M**odel (BOM).
- Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

The Window Object

- The window object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:

```
window.document.getElementById("header");
```

- is the same as:

```
document.getElementById("header");
```

Window Size

- Two properties can be used to determine the size of the browser window.
- Both properties return the sizes in pixels:
  - window.innerHeight - the inner height of the browser window (in pixels)
  - window.innerWidth - the inner width of the browser window (in pixels)
- The browser window (the browser viewport) is NOT including toolbars and scrollbars.
- For Internet Explorer 8, 7, 6, 5:
  - document.documentElement.clientHeight
  - document.documentElement.clientWidth

  or
  - document.body.clientHeight
  - document.body.clientWidth

Example

- A practical JavaScript solution (covering all browsers):

Browser inner window width: 705, height: 747.

```
<html>
<body>
  <p id="demo"></p>
  <script>
    var w = window.innerWidth
      || document.documentElement.clientWidth
      || document.body.clientWidth
    var h = window.innerHeight
      || document.documentElement.clientHeight
      || document.body.clientHeight;
    var x = document.getElementById("demo");
    x.innerHTML = "Browser inner window width: " + w + ", height: " + h + ".";
  </script>
</body>
</html>
```

Other Window Methods

- Some other methods:
  - window.open() - open a new window
  - window.close() - close the current window
  - window.moveTo() - move the current window
  - window.resizeTo() - resize the current window

Window Screen

- The window.screen object contains information about the user's screen.
- The window.screen object can be written without the window prefix.
- Properties:
  - screen.width
  - screen.height
  - screen.availWidth
  - screen.availHeight
  - screen.colorDepth
  - screen.pixelDepth

Window Location

- The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.
- The window.location object can be written without the window prefix.
- Some examples:
  - window.location.href returns the href (URL) of the current page
  - window.location.hostname returns the domain name of the web host
  - window.location.pathname returns the path and filename of the current page
  - window.location.protocol returns the web protocol used (http: or https:)
  - window.location.assign() loads a new document

Window History

- The window.history object contains the browsers history.
- The window.history object can be written without the window prefix.
- To protect the privacy of the users, there are limitations to how JavaScript can access this object.
- Some methods:
  - history.back() - same as clicking back in the browser
  - history.forward() - same as clicking forward in the browser

Window History

```
<html>
<head>
<script>
function goBack() {
  window.history.back()
}
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

Back

Window Navigator

- The window.navigator object contains information about the visitor's browser.
- The window.navigator object can be written without the window prefix.
- Some examples:
  - navigator.appName: returns the application name of the browser
  - navigator.appCodeName: returns the application code name of the browser
  - navigator.platform: returns the product name of the browser engine:

Popup Boxes – Alert box

- JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.
- An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.
  - Syntax: window.alert("*sometext*");
  - Example

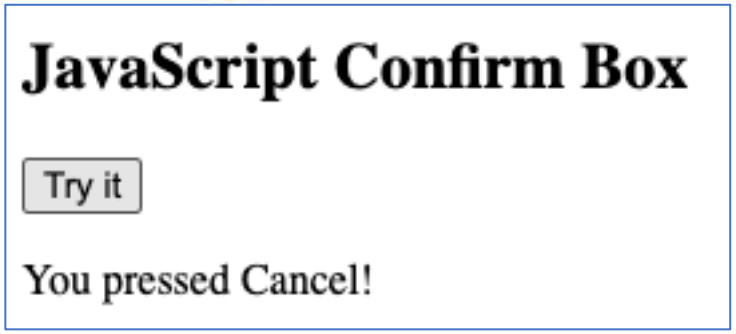
Popup Boxes – Confirm box

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.
- Syntax

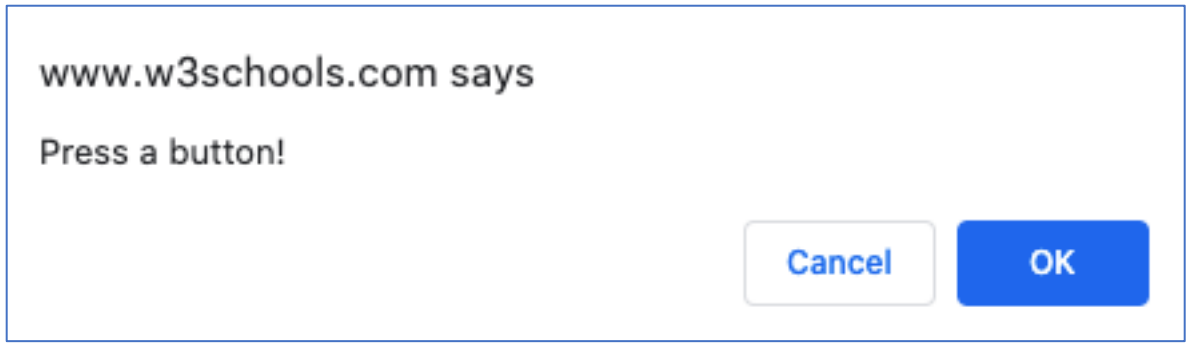
```
window.confirm("sometext");
```

- The window.confirm() method can be written without the window prefix.

Confirm box




```
<html>
<body>
  <h2>JavaScript Confirm Box</h2>
  <button onclick="myFunction()">Try it</button>
  <p id="demo"></p>
  <script>
    function myFunction() {
      var txt;
      if (confirm("Press a button!")) {
        txt = "You pressed OK!";
      } else {
        txt = "You pressed Cancel!";
      }
      document.getElementById("demo").innerHTML = txt;
    }
  </script>
</body>
</html>
```



Popup Boxes – Prompt box

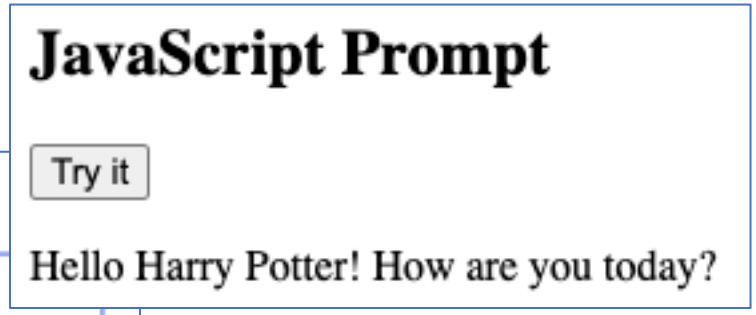
- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- Syntax

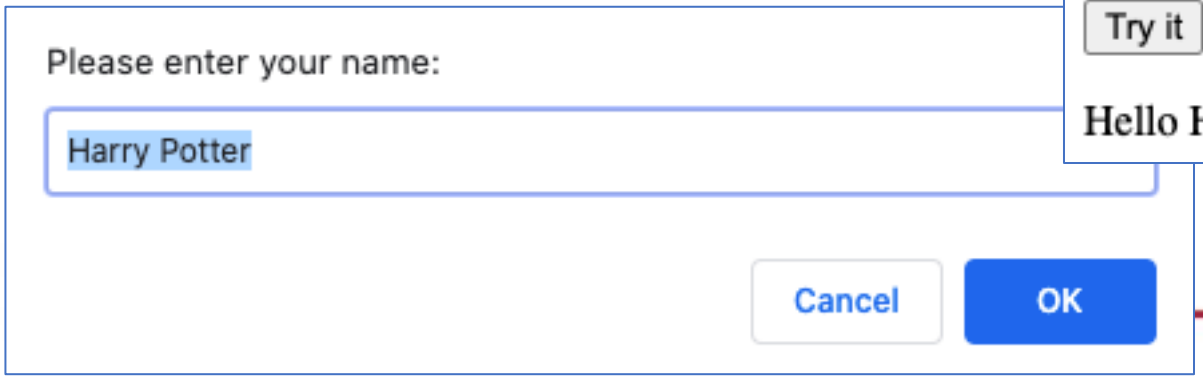
```
window.prompt("sometext","defaultText");
```

- The window.prompt() method can be written without the window prefix.

Popup Boxes – Prompt box

```
<html>
<body>
  <h2>JavaScript Prompt</h2>
  <button onclick="myFunction()">Try it</button>
  <p id="demo"></p>
  <script>
    function myFunction() {
      var txt;
      var person = prompt("Please enter your name:", "Harry Potter");
      if (person == null || person == "") {
        txt = "User cancelled the prompt.";
      } else {
        txt = "Hello " + person + "! How are you today?";
      }
      document.getElementById("demo").innerHTML = txt;
    }
  </script>
</body>
</html>
```





Timing Events

- The window object allows execution of code at specified time intervals.
- These time intervals are called timing events.
- The two key methods to use with JavaScript are:
  - window.setTimeout(*function, milliseconds*)
    Executes a function, after waiting a specified number of milliseconds.
  - window.setInterval(*function, milliseconds*)
    Same as setTimeout(), but repeats the execution of the function continuously.
- Stop the execution:

```
myVar = setTimeout(function, milliseconds);
window.clearTimeout(myVar);

myVar = setInterval(function, milliseconds);
clearInterval(myVar);
```

Timing Events

- This example executes a function called "myTimer" once every second (like a digital watch).

```
<html>
<body>
  <p>A script on this page starts this clock:</p>
  <p id="demo"></p>
  <button onclick="clearInterval(myVar)">Stop time</button>
  <script>
    var myVar = setInterval(myTimer, 1000);
    function myTimer() {
      var d = new Date();
      document.getElementById("demo").innerHTML = d.toLocaleTimeString();
    }
  </script>
</body>
</html>
```

A script on this page starts this clock:

1:20:50 PM

Stop time

Document Object Model (DOM)

1. Introduction
2. DOM Methods
3. DOM Document
4. DOM Elements
5. DOM - Changing HTML
6. DOM - Changing CSS
7. DOM Animation
8. DOM Events
9. DOM EventListener
10. DOM Navigation
11. DOM Elements (Nodes)
12. DOM Collections
13. DOM Node Lists
14. Browser Object Model (BOM)
15. **Jquery HTML DOM**

jQuery HTML DOM

- https://jquery.com/
- jQuery was created in 2006 by John Resig. It was designed to handle Browser Incompatibilities and to simplify HTML DOM Manipulation, Event Handling, Animations, and Ajax.
- For more than 10 years, jQuery has been the most popular JavaScript library in the world.

jQuery HTML DOM – Removing Element

```
<html>
<head>
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.
js"></script>
</head>
<body>
   <h2>Remove an HTML Element</h2>
   <h2 id="id01">Hello World!</h2>
   <h2 id="id02">Hello Sweden!</h2>
   <script>
      $(document).ready(function () {
         $("#id02").remove();
      });
   </script>
</body>
</html>
```

Remove an HTML Element

Hello World!

jQuery HTML DOM – Get Parent Element

```
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
</head>
<body>
  <h1>Getting Parent HTML Element</h1>
  <h2 id="01">Hello World!</h2>
  <h2 id="02">Hello Sweden!</h2>
  <h2 id="03">Hello Japan!</h2>
  <p id="demo"></p>
  <script>
    $(document).ready(function () {
      var myParent = $("#02").parent();
      $("#demo").text(myParent.prop("nodeName"));
    });
  </script>
</body>
</html>
```

jQuery HTML DOM – Get Parent Element

```
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
</head>
<body>
  <h1>Getting Paren
  <h2 id="01">Hello
  <h2 id="02">Hello
  <h2 id="03">Hello
  <p id="demo"></p
  <script>
    $(document).rea
      var myParent =
      $("#demo").te
    });
  </script>
</body>
</html>
```

Getting Parent HTML Element

Hello World!

Hello Sweden!

Hello Japan!

BODY

EXERCISE

Exercise 1

- Extract the contents of paragraph tags
- Highlight all of the words over 8 characters long in the paragraph text (with a yellow background for example)

Heading

Hey, you're not permitted in there. It's restricted. You'll be deactivated for sure.. Don't call me a mindless philosopher, you overweight glob of grease! Now come out before somebody sees you. Secret mission? What plans? What are you talking about? I'm not getting in there! I'm going to regret this. There goes another one. Hold your fire. There are no life forms. It must have been short-circuited. That's funny, the damage doesn't look as bad from out here. Are you sure this things safe? Close up formation. You'd better let her loose. Almost there! I can't hold them! It's away! It's a hit! Negative. Negative! It didn't go in. It just impacted on the surface. Red Leader, we're right above you. Turn to point... oh-five, we'll cover for you. Stay there... I just lost my starboard engine. Get set to make your attack run. The Death Star plans are not in the main computer. Where are those transmissions you intercepted? What have you done with those plans? We intercepted no transmissions. Aaah....This is a consular ship. Were on a diplomatic mission. If this is a consular ship...were is the Ambassador? Commander, tear this ship apart until you've found those plans and bring me the Ambassador. I want her alive! There she is! Set for stun! She'll be all right. Inform Lord Vader we have a prisoner. What a piece of junk. She'll make point five beyond the speed of light. She may not look like much, but she's got it where it counts, kid. I've added some special modifications myself. We're a little rushed, so if you'll hurry aboard we'll get out of here. Hello, sir.

Exercise 2

- Replace all question marks (?) with thinking faces (🤔) and exclamation marks (!) with astonished faces (😲)

Hey, you're not permitted in there.

It's restricted. You'll be deactivated for sure..

Don't call me a mindless philosopher, you overweight glob of grease🤔 Now come out before somebody sees you.

Secret mission🤔 What plans🤔 What are you talking about🤔 I'm not getting in there🤔 I'm going to regret this.

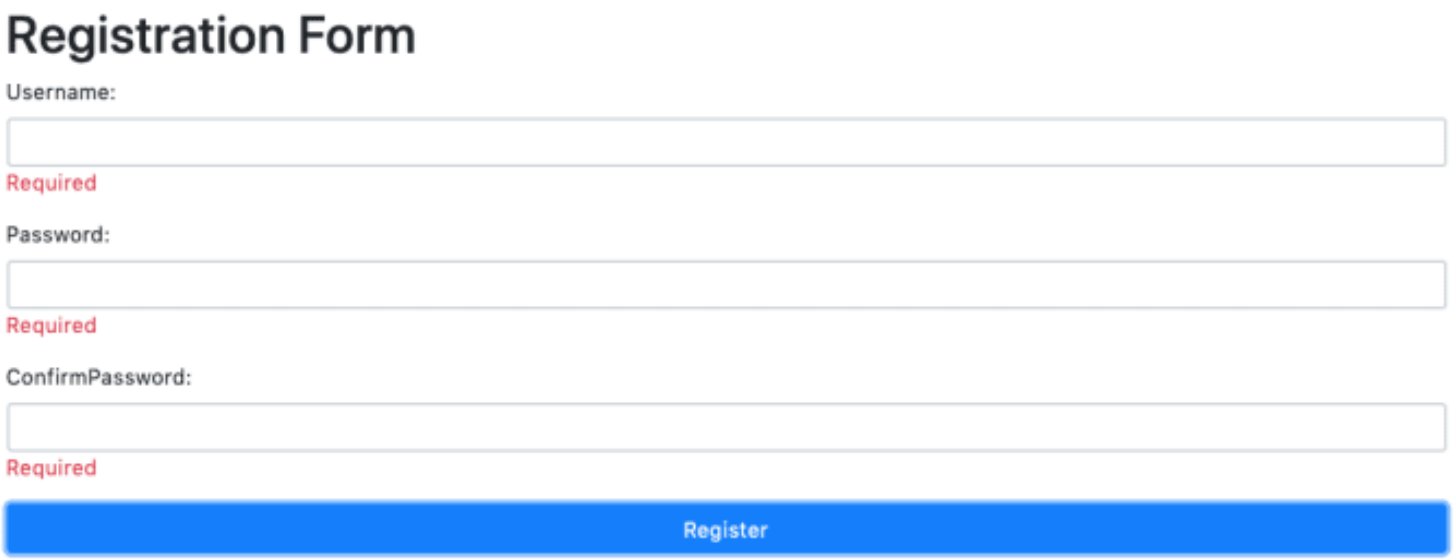
There goes another one.

Hold your fire.

There are no life forms.

Exercise 3

- Add a required validation to each input that shows an error message next to the entry if it does not have any text entered.



Exercise 4

- *Write a function called by clicking a button on a page to alert*
  - The number of links on the page
  - The first and last of these links

Get all links

DOM Reference Portable FF

An embedded page at null.jsbin.com says

2

OK

First:
http://www.b92.net/info/rss/tehnopolis.xml

Last:
http://portableapps.com/apps/internet/firefox_portable

OK

Exercise 5

- Create a webpage with an h1 of "My Book List".
- Add a script tag to the bottom of the page, where all your JS will go. Copy the array of books from belows.
- Iterate through the array of books. For each book, create a p element with the book title and author and append it to the page.
- Use a ul and li to display the books. Add a property to each book with the URL of the book cover, and add an img element for each book on the page. Change the style of the book depending on whether you have read it or not.

```
var books = [
  {
    title: 'The Design of EveryDay Things',
    author: 'Don Norman',
    alreadyRead: false
  },
  {
    title: 'The Most Human Human',
    author: 'Brian Christian',
    alreadyRead: true
  }
];
```

Exercise 5

- Cre
- Ad go.
  Co
- Ite ent
  wi
- Us with
  the on
  the u
  ha

My Book List

The Design of EveryDay Things by Don Norman

The Most Human Human by Brian Christian

- 
  The Design of EveryDay Things by Don Norman
- 
  The Most Human Human by Brian Christian

}

];

Exercise 6

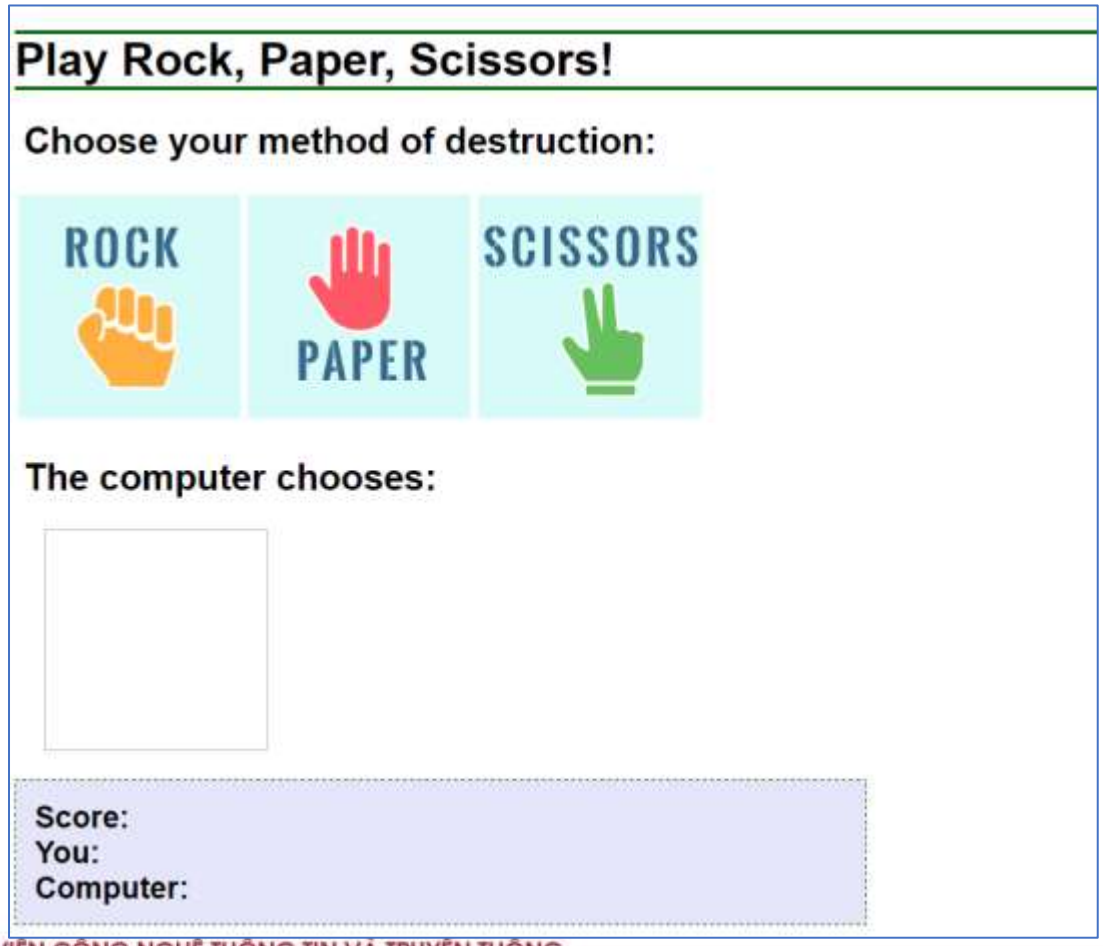
- *Write three buttons:*
  - *Select first*
  - *The second Uncheck All*
  - *A third anti-election*

Select | deselect all | Invert

- [x] Tiexuedanxin
- [x] Fire
- [ ] sea
- [ ] Hao Hange
- [ ] We are not the same
- [ ] Chengdu
- [ ] Banhu yarn
- [ ] How do you want me
- [ ] love life
- [ ] chasing light by

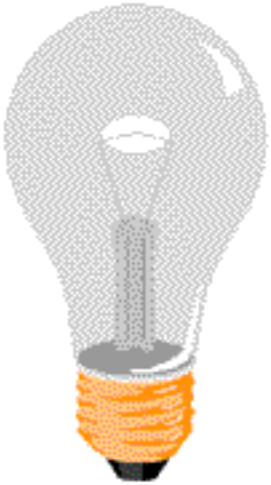
Exercise 7

- Write a web application that plays Rock, Paper, Scissors. In this assignment you will create all of the elements on the main app page



Excercise

- Change an image when a user holds down the mouse button (user must click and hold down to turn on the light)



Click mouse and hold down!

Click mouse and hold down!

https://www.w3schools.com/js/bulboff.gif

https://www.w3schools.com/js/bulbon.gif

Excercise

- Change the background-color of an input field when it gets focus.

Enter your name:

When the input field gets focus, a function is triggered which changes the background-color.

Enter your name:

When the input field gets focus, a function is triggered which changes the background-color.

Excercise

- Change the color of a heading when the cursor moves over it.

Mouse over this text

Mouse over this text

Exercise

JavaScript addEventListener()

This example uses the addEventListener() method to add many events on the same button.

Try it

Moused over!
Clicked!
Moused out!
Moused over!
Moused out!