Introduction to JavaScript: Part 3

Introduction to Internet and Web

부산대학교 정보·의생명공학대학
정보컴퓨터공학부

Contents

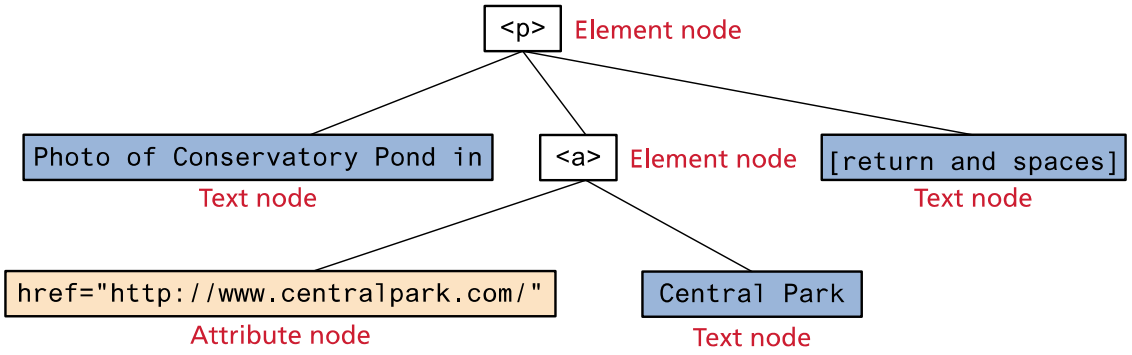
- ❖ JavaScript HTML DOM
- ❖ JavaScript Events
- ❖ Form Validation

JAVASCRIPT HTML DOM

HTML DOM

- When a web page is loaded, the browser creates a Document Object Model of the page
- The HTML DOM model is constructed as a tree of Objects

```
<p>Photo of Conservatory Pond in
   <a href="http://www.centralpark.com/">Central Park</a>
</p>
```



What is the HTML DOM?

❖ **The HTML DOM is a standard object model and programming interface for HTML. It defines:**

- The HTML elements.
- 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.**

What is the HTML DOM?

- **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.**
  - In the example above, 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>
```

Finding HTML elements

❖ By ID

```
const element = document.getElementById("intro");
```

❖ By tag name

```
const element = document.getElementsByTagName("p");
```

❖ By class name

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

❖ By CSS selectors

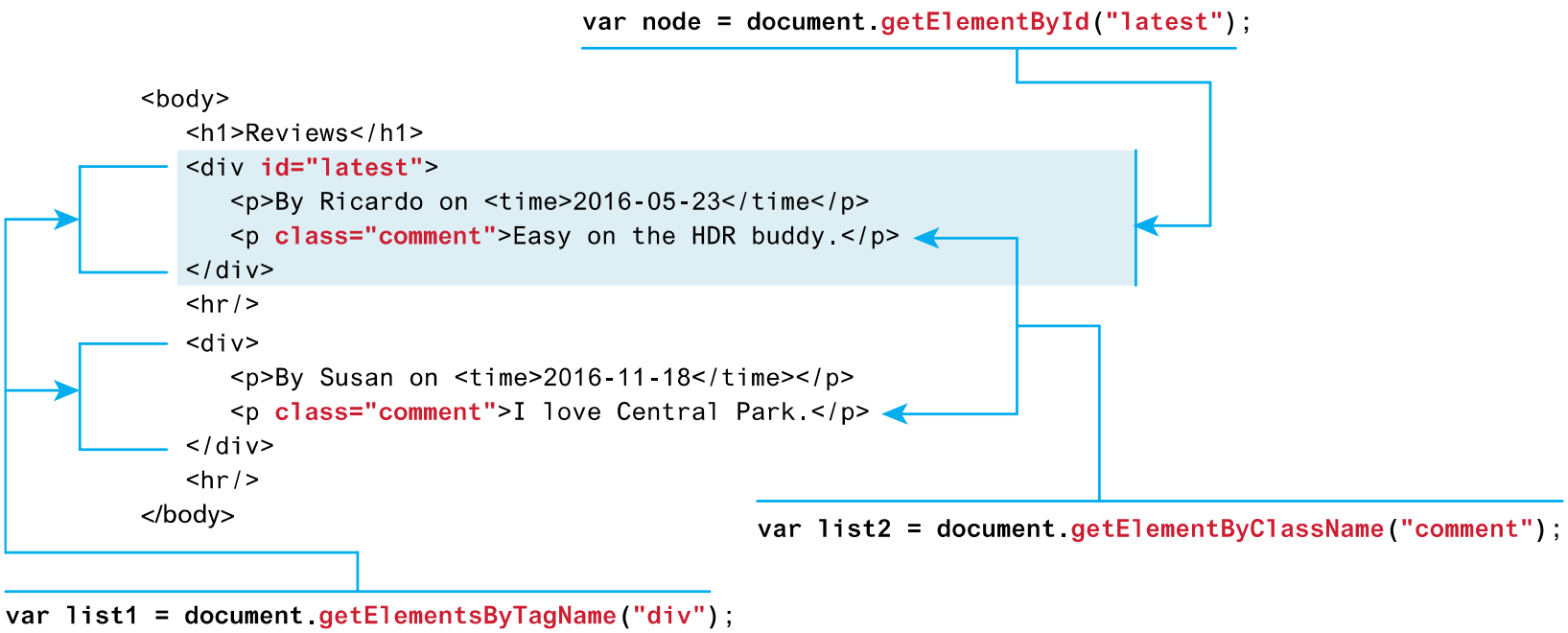
```
const x = document.querySelectorAll("p.intro");
```

❖ Example

```
const x = document.getElementById("main");
const y = x.getElementsByTagName("p");
```

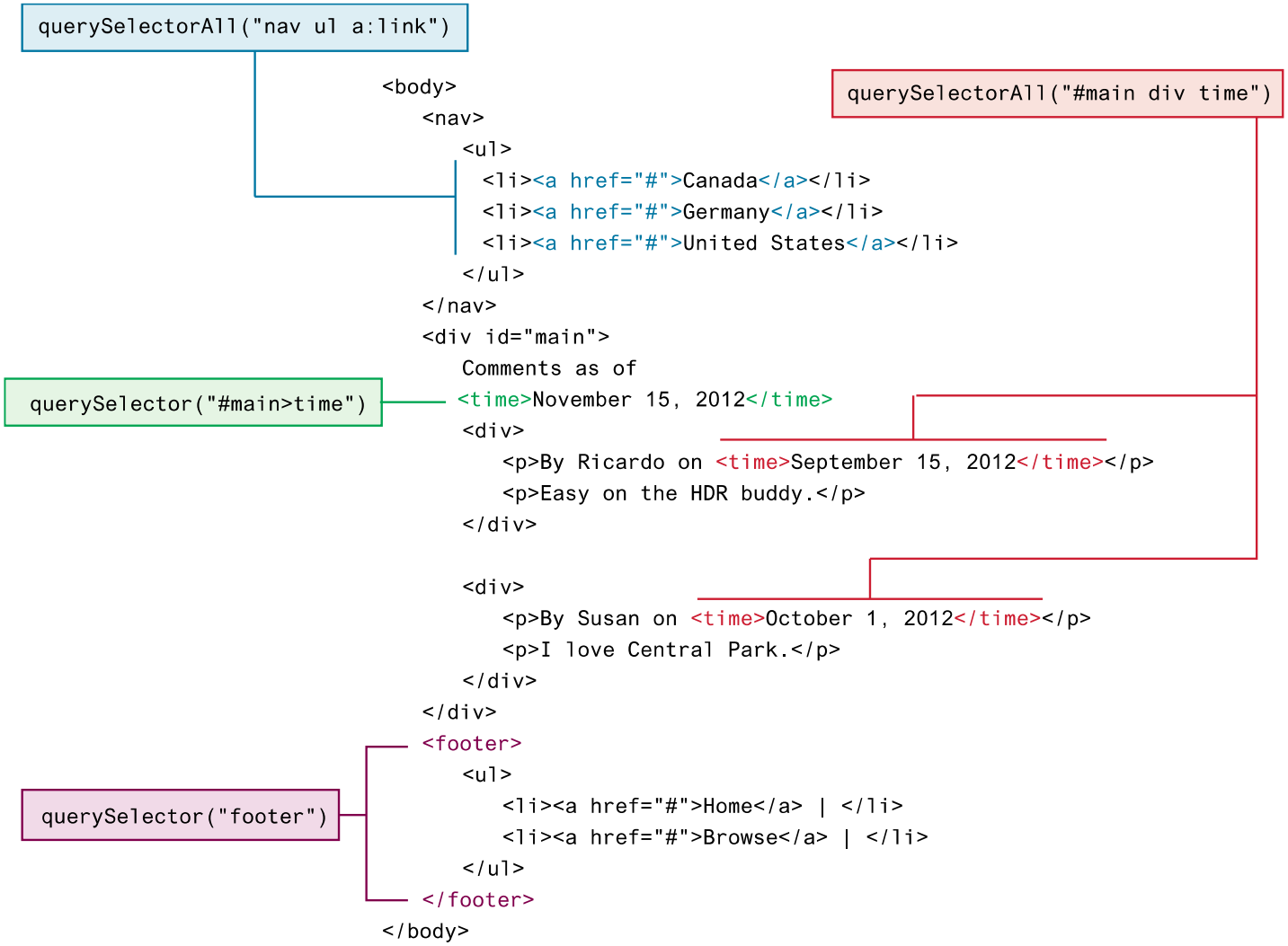
Finding HTML elements

❖ Selection Methods



Finding HTML elements

❖ Query Selector



Element node Object

❖ Essential Element Node Properties

- https://developer.mozilla.org/en-US/docs/Web/API/Element

Property	Description
className	The current value for the class attribute of this HTML element.
id	The current value for the id of this element.
innerHTML	Represents all the things inside of the tags. This can be read or written to and is the primary way which we update particular div's using JS.
style	The style attribute of an element. We can read and modify this property.
tagName	The tag name for the element.

Element node Object

❖ Some Specific HTML DOM Element Properties for Certain Tag Types

Property	Description	Tags
href	The href attribute used in a tags to specify a URL to link to.	a
name	The name property is a bookmark to identify this tag. Unlike id which is available to all tags, name is limited to certain form related tags.	a, input, textarea, form
src	Links to an external URL that should be loaded into the page (as opposed to href which is a link to follow when clicked)	img, input, iframe, script
value	The value is related tot he value attribute of input tags. Often the value of an input field is user defined, and we use value to get that user input.	Input, textarea, submit

DOM navigation

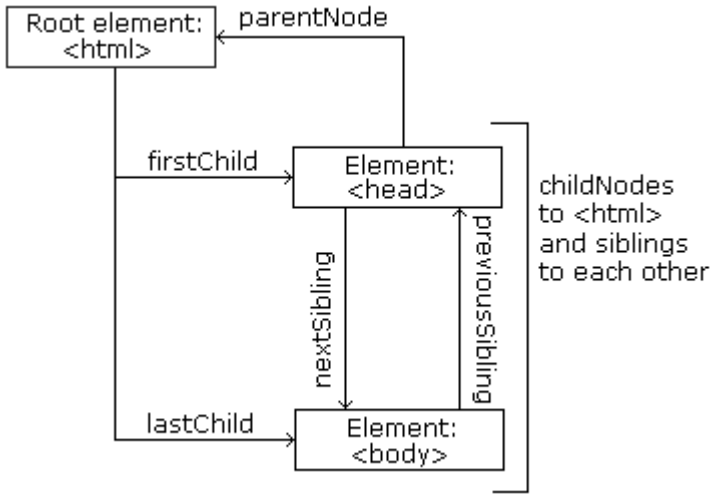
❖ With the HTML DOM, you can navigate the node tree using node relationships.

```
<html>

  <head>
    <title>DOM Tutorial</title>
  </head>

  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>

</html>
```



DOM Nodes

❖ Essential Node Object properties

- ***Element*** inherits properties from its parent interface, ***Node***, and by extension that interface's ***EventTarget***
- ***EventTarget*** ←***Node*** ←***Element***

Property	Description
attributes	Collection of node attributes
childNodes	A NodeList of child nodes for this node
firstChild	First child node of this node.
lastChild	Last child of this node.
nextSibling	Next sibling node for this node.
nodeName	Name of the node
nodeType	Type of the node
nodeValue	Value of the node
parentNode	Parent node for this node.
previousSibling	Previous sibling node for this node.

nodeType

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

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

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

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>

nodeValue and nodeName

❖ 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 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
- nodeName always contains the uppercase tag name of an HTML element.

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

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

DOM navigation example

```
<html>
<body>

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

<script>
// 1()
document.getElementById("id02").innerHTML = document.getElementById("id01").innerHTML;
// (2)
document.getElementById("id02").innerHTML =
          document.getElementById("id01").firstChild.nodeValue;
// (3)
document.getElementById("id02").innerHTML =
          document.getElementById("id01").childNodes[0].nodeValue;
</script>

</body>
</html>
```

Creating New HTML Elements

❖ 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.

1. Create a new text node

```
var text = document.createTextNode("this is dynamic");
```

"this is dynamic"

2. Create a new empty element

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

3. Add the text node to new element

```
p.appendChild(text);
```

 "this is dynamic"

4. Add the element to the

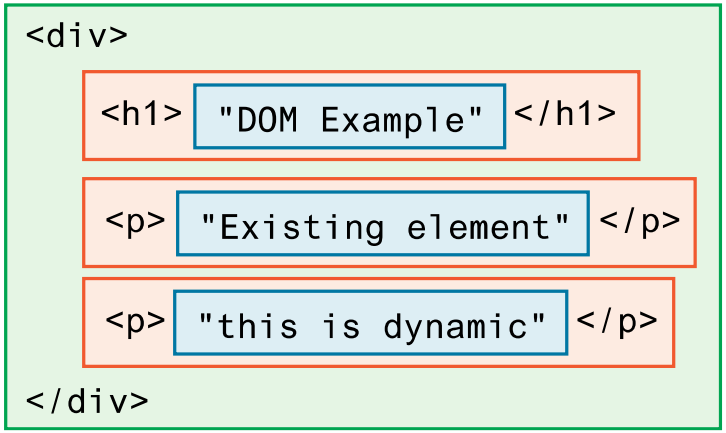
```
var first = document.getElementById("first");
first.appendChild(p);
```

Creating New HTML Elements

4 Add the <p> element to the

```
var first = document.getElementById("first");
first.appendChild(p);
```

```
<div id="first">
    <h1>DOM Example</h1>
    <p>Existing element</p>
    <p>this is dynamic</p>
</div>
```



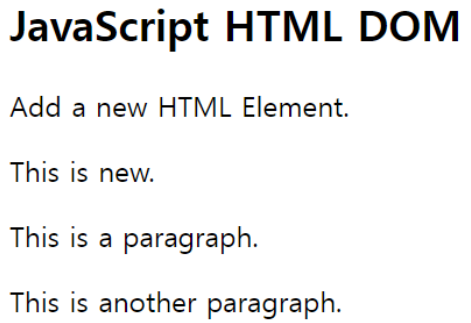
Creating new HTML Elements

- The appendChild() method 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>
const para = document.createElement("p");
const node = document.createTextNode("This is new.");
para.appendChild(node);

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



Removing HTML Elements

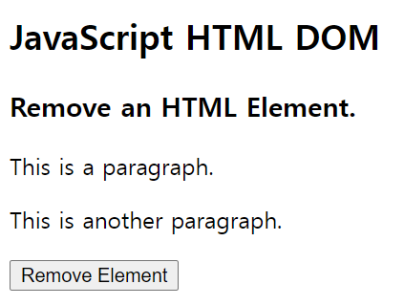
❖ To remove an HTML element, use the remove() method:

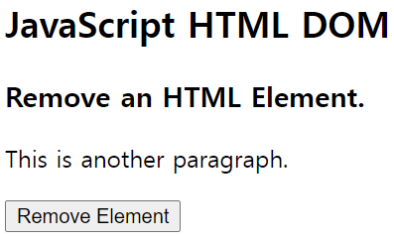
```
<h2>JavaScript HTML DOM</h2>
<h3>Remove an HTML Element.</h3>

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

<button onclick="myFunction()">Remove Element</button>

<script>
function myFunction() {
document.getElementById("p1").remove();
}
</script>
```



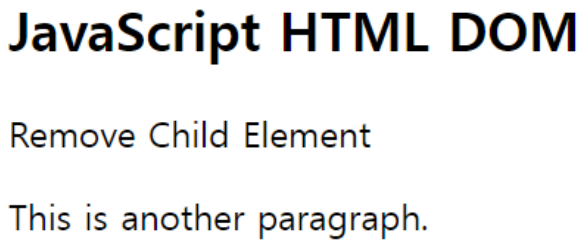


Removing a Child Node

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



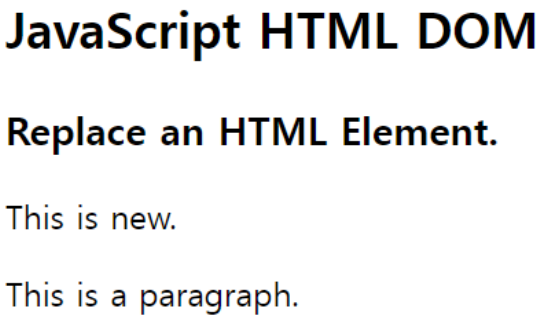
Replacing HTML Elements

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

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

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

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



JAVASCRIPT EVENTS

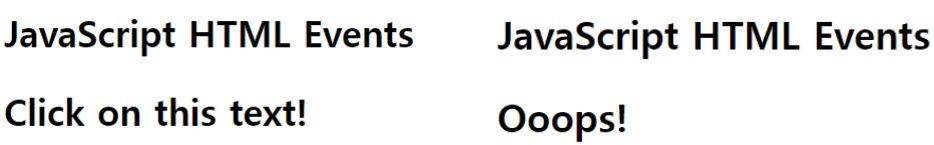
Reacting to Events

- 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*

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

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

</body>
</html>
```



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>
```

Inline Hooks

HTML document using the inline hooks

```
...
 <script type="text/javascript" src="inline.js"></script>
...
<form name='mainForm' onsubmit="validate(this);">
      <input name="name" type="text"
          onchange="check(this);"
          onfocus="highlight(this, true);"
          onblur="highlight(this, false);">
      <input name="email" type="text"
          onchange="check(this);"
          onfocus="highlight(this, true);"
          onblur="highlight(this, false);">
      <input type="submit"
          onclick="function (e) {
                      ...
                  }">
...
```

Notice that you can define an entire event handling function within the markup. This is NOT recommended!

inline.js

```
function validate(node) {
  ...
}
function check(node) {
  ...
}
function highlight(node) {
  ...
}
```

Assign Events Using HTML DOM

❖ The HTML DOM allows you to assign events to HTML elements using JavaScript:

```
<body>

  <h2>JavaScript HTML Events</h2>
  <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 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.
- 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.

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

HTML DOM EventListener

```
<body>

  <h2>JavaScript addEventListener()</h2>

  <p>This example uses the addEventListener() method to execute a function
when a user clicks on a button.</p>

  <button id="myBtn">Try it</button>

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

  function myFunction() {
    alert ("Hello World!");
  }
  </script>

  </body>
```

HTML DOM EventListener

❖ Add Many Event Handlers to the Same Element

```
<body>

  <h2>JavaScript addEventListener()</h2>
  <p>This example uses the addEventListener() method to add two click events
to the same button.</p>

  <button id="myBtn">Try it</button>

  <script>
  var x = document.getElementById("myBtn");
  x.addEventListener("click", myFunction);
  x.addEventListener("click", someOtherFunction);
    function myFunction() {
    alert ("Hello World!");
  }
    function someOtherFunction() {
    alert ("This function was also executed!");
  }
  </script>
</body>
```

HTML DOM EventListener

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

```
<h2>JavaScript addEventListener()</h2>
<p>This example demonstrates how to pass parameter values when using the addEventListener() method.</p>
<p>Click the button to perform a calculation.</p>

<button id="myBtn">Try it</button>
<p id="demo"></p>

<script>
let p1 = 5;
let p2 = 7;
document.getElementById("myBtn").addEventListener("click", function() {
  myFunction(p1, p2);
});

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

Event Types

❖ Mouse Events

Event	Description
onclick	The event occurs when the user clicks on an element
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu
ondblclick	The event occurs when the user double-clicks on an element
onmousedown	The event occurs when the user presses a mouse button over an element
onmouseenter	The event occurs when the pointer is moved onto an element
onmouseleave	The event occurs when the pointer is moved out of an element
onmousemove	The event occurs when the pointer is moving while it is over an element
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children
onmouseup	The event occurs when a user releases a mouse button over an element

Event Types

❖ Keyboard Events, Form Events

Attribute	Value	Description
onkeydown	*script*	Fires when a user is pressing a key
onkeypress	*script*	Fires when a user presses a key
onkeyup	*script*	Fires when a user releases a key

Attribute	Value	Description
onblur	*script*	Fires the moment that the element loses focus
onchange	*script*	Fires the moment when the value of the element is changed
oncontextmenu	*script*	Script to be run when a context menu is triggered
onfocus	*script*	Fires the moment when the element gets focus
oninput	*script*	Script to be run when an element gets user input
oninvalid	*script*	Script to be run when an element is invalid
onreset	*script*	Fires when the Reset button in a form is clicked
onsearch	*script*	Fires when the user writes something in a search field (for <input="search">)
onselect	*script*	Fires after some text has been selected in an element
onsubmit	*script*	Fires when a form is submitted

Event Objects

❖ **When an event is triggered, the browser will construct an event object that contains information about the event.**

- https://developer.mozilla.org/en-US/docs/Web/API/Event

```
<body>
  <div onmousemove="showCoords(event)" onmouseout="clearCoor()"></div>
  <p>Mouse over the rectangle above to get the horizontal and vertical
coordinates of your mouse pointer.</p>
  <p id="demo"></p>

  <script>
  function showCoords(event) {
    var x = event.clientX;
    var y = event.clientY;
    var coor = "X coords: " + x + ", Y coords: " + y;
    document.getElementById("demo").innerHTML = coor;
  }

  function clearCoor() {
    document.getElementById("demo").innerHTML = "";
  }
  </script>
</body>
```

Event Objects

```
<body>

  <p>Press the "O" key on the keyboard in the input field to
alert some text.</p>

  <input type="text" size="40" onkeypress="myFunction(event)">

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

  <script>
  function myFunction(event) {
    var x = event.charCode || event.keyCode;
    if (x == 111 || x == 79) {  // o is 111, O is 79
      alert("You pressed the 'O' key!");
    }
  }
  </script>

  </body>
```

Event Bubbling/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.

Event Bubbling/Capturing

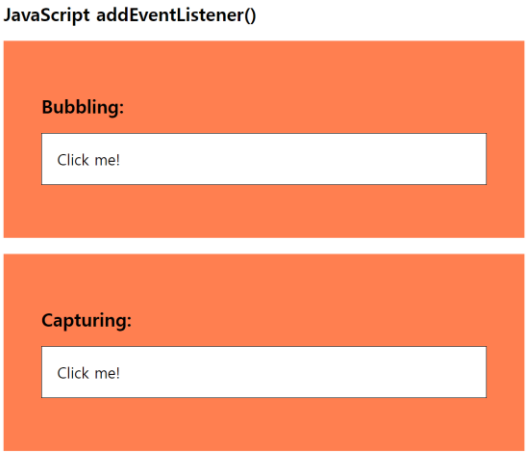
```
<h2>JavaScript addEventListener()</h2>
<div id="myDiv1">
  <h2>Bubbling:</h2>
  <p id="myP1">Click me!</p>
</div><br>
<div id="myDiv2">
  <h2>Capturing:</h2>
  <p id="myP2">Click me!</p>
</div>

<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);

document.getElementById("myP2").addEventListener("click", function() {
  alert("You clicked the white element!");
}, true);

document.getElementById("myDiv2").addEventListener("click", function() {
  alert("You clicked the orange element!");
}, true);
</script>
```



FORM VALIDATION

JavaScript Form Validation

❖ **The function can be called when the form is submitted:**

```
<head>
  <script>
  function validateForm() {
    let x = document.forms["myForm"]["fname"].value;
    if (x == "") {
      alert("Name must be filled out");
      return false;
    }
  }
  </script>
  </head>
  <body>

  <h2>JavaScript Validation</h2>

  <form name="myForm" action="/action_page.php" onsubmit="return validateForm()"
method="post">
    Name: <input type="text" name="fname">
    <input type="submit" value="Submit">
  </form>

  </body>
```

Automatic HTML Form Validation

- HTML form validation can be performed automatically by the browser:
- If a form field (fname) is empty, the required attribute prevents this form from being submitted:

```
<body>

  <h2>JavaScript Validation</h2>

  <form action="/action_page.php" method="post">
    <input type="text" name="fname" required>
    <input type="submit" value="Submit">
  </form>

  <p>If you click submit, without filling out the text field,
  your browser will display an error message.</p>

  </body>
```

요 약

Summary

- JavaScript HTML DOM
- JavaScript Events
- Form Validation