Objectives

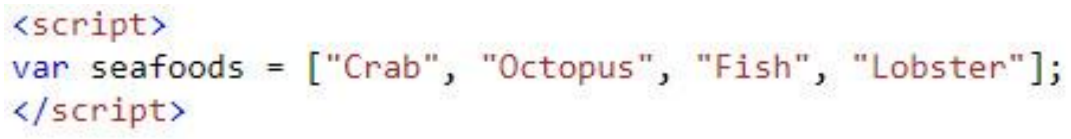
- In this session, you will learn to:
  - JavaScript Arrays
  - HTML DOM Model
  - List of events in JavaScript
  - Validation form using JavaScript

JavaScript Arrays

- JavaScript arrays are used to store multiple values in a single variable.
- An array is a special variable, which can hold more than one value at a time.
- An array can hold many values under a single name, and you can access the values by referring to an index number.

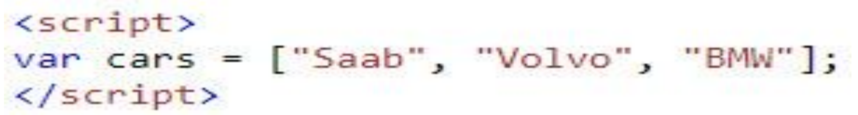
Creating an Array

- Using an array literal



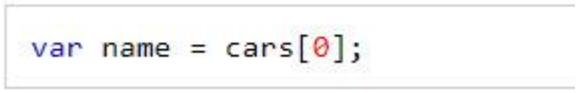
```
<script>
var seafoods = ["Crab", "Octopus", "Fish", "Lobster"];
</script>
```

- Using the JavaScript Keyword new



```
<script>
var cars = ["Saab", "Volvo", "BMW"];
</script>
```

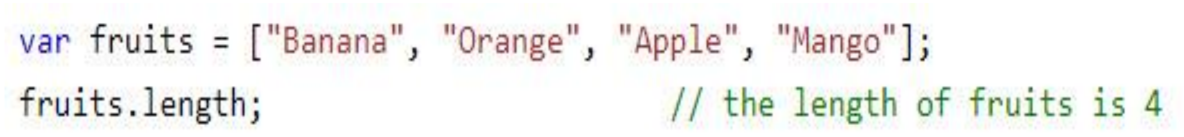
- You refer to an array element by referring to the **index number**.



```
var name = cars[0];
```

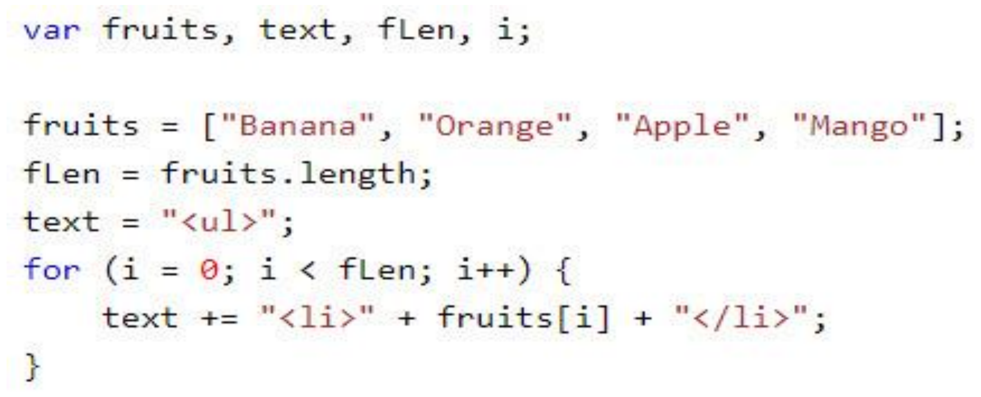
JavaScript Arrays

- The **length** property
  - Returns the length of an array (the number of array elements).



```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;                          // the length of fruits is 4
```

- Looping Array Elements
  - Using a for loop



```
var fruits, text, fLen, i;

fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
    text += "<li>" + fruits[i] + "</li>";
}
```

HTML DOM

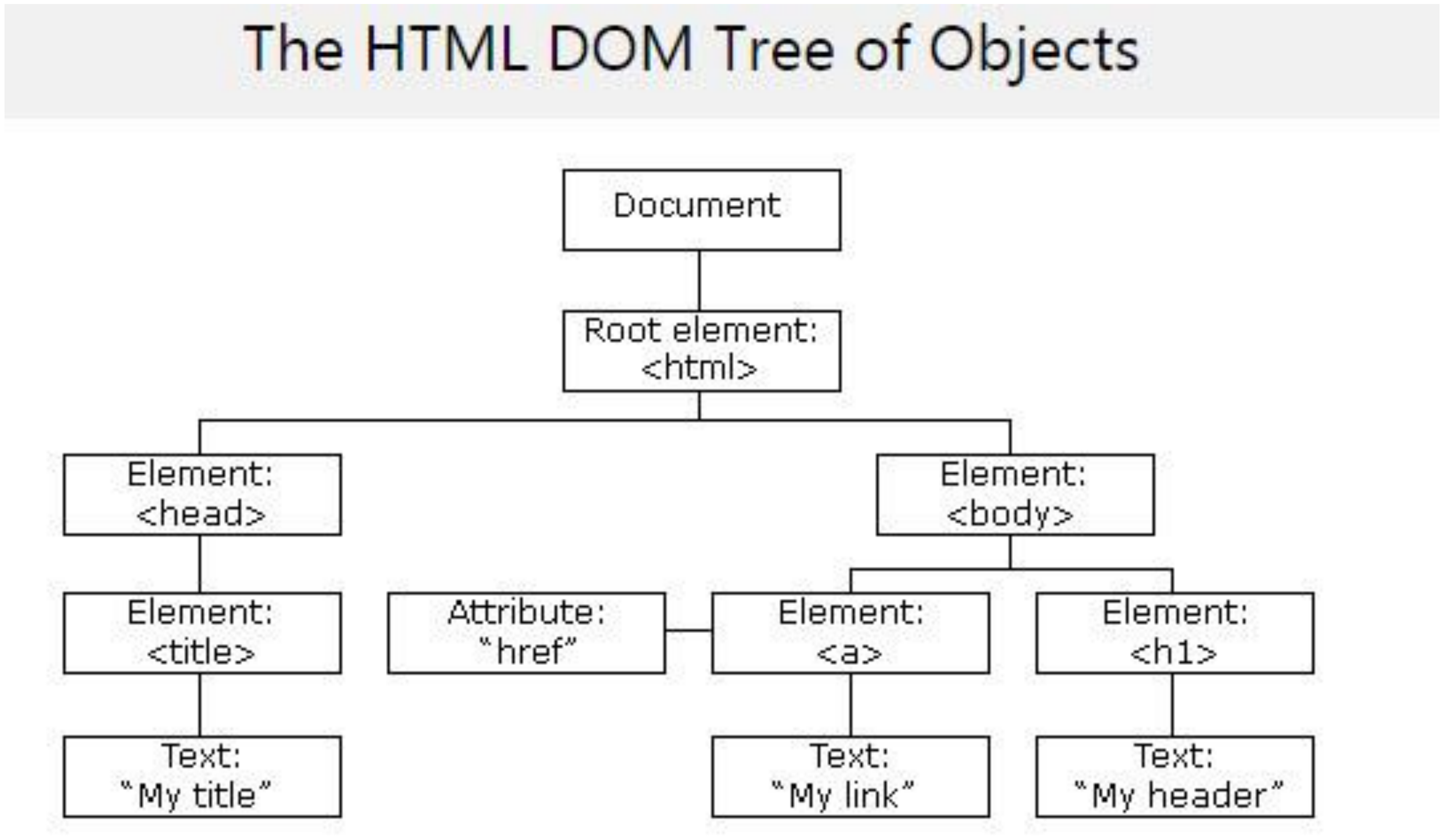
- The HTML DOM (Document Object Model)
  - With the HTML DOM, JavaScript can access and change all the elements of an HTML document.
  - When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
  - The DOM is a W3C (World Wide Web Consortium) standard.
  - The DOM 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."*

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.**

HTML DOM Model

- The **HTML DOM** model is constructed as a tree of **Objects**:

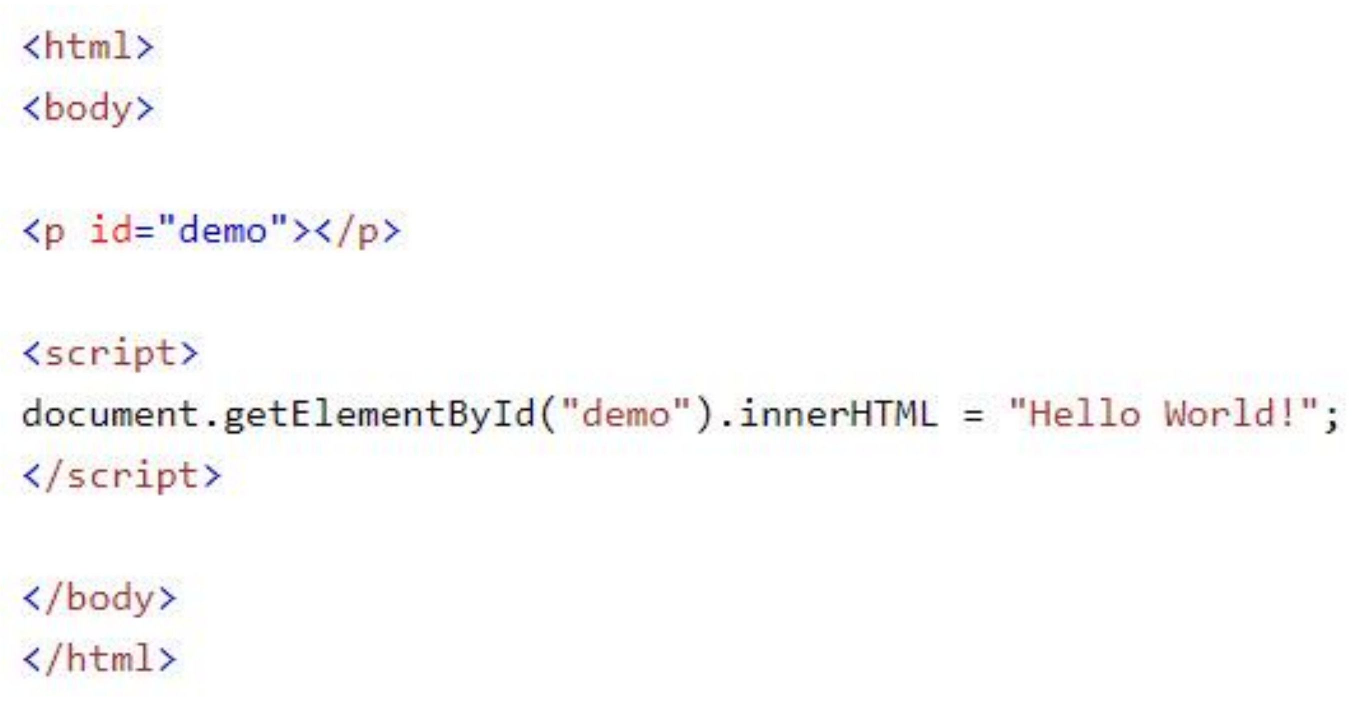


HTML DOM Methods

- 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).
  - getElementById() Method
    - Access an HTML element is to use the id of the element
  - innerHTML Property
    - Useful for getting or replacing the content of HTML elements.

Demo

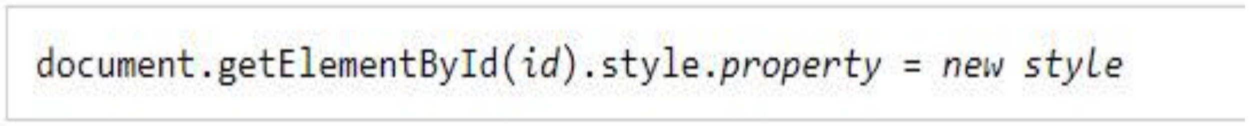
Example:



```
<html>
<body>

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

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

</body>
</html>
```

JavaScript HTML DOM - Changing CSS

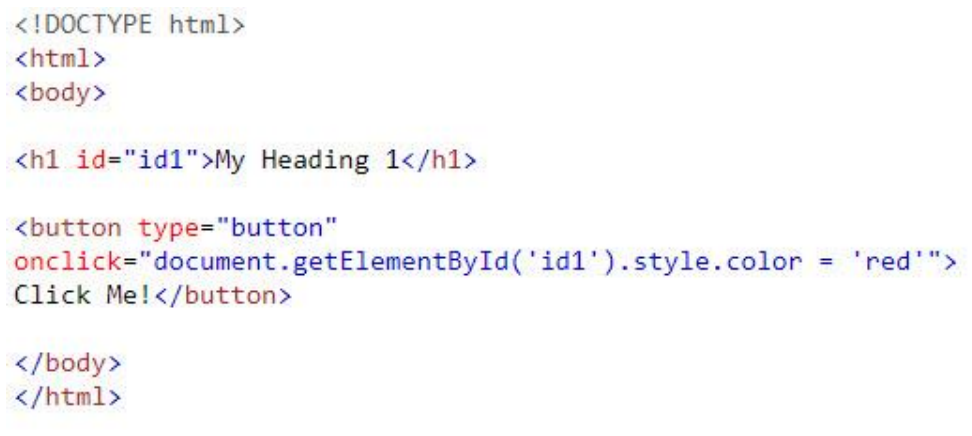
- The HTML DOM allows you to execute code when an event occurs.
- Events are generated by the browser when "things happen" to HTML elements:
  - An element is clicked on
  - The page has loaded
  - Input fields are changed
- To change the style of an HTML element, use this syntax:



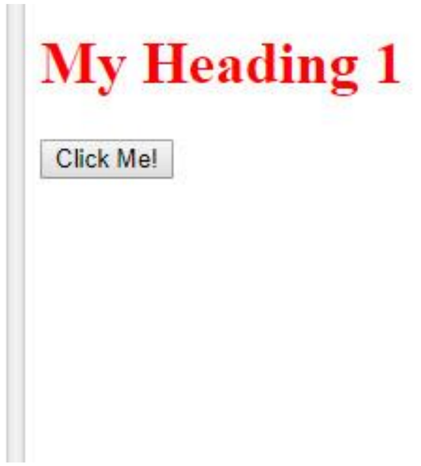
```
document.getElementById(id).style.property = new style
```

Demo

Example:

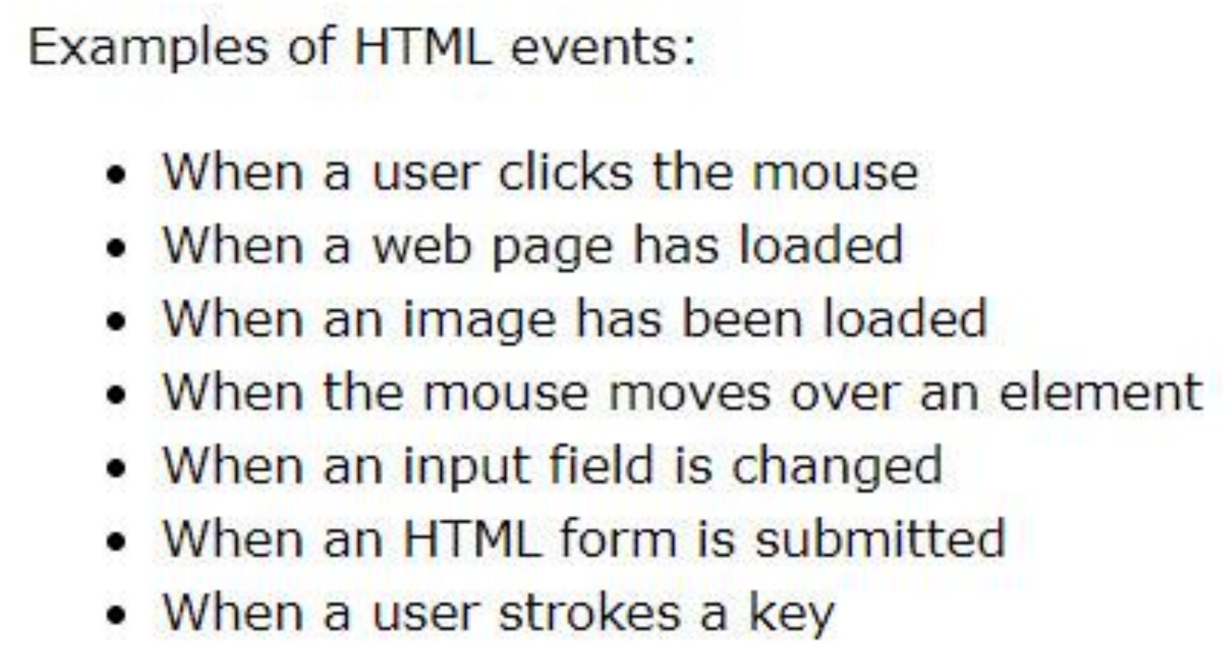


```
<!DOCTYPE html>
<html>
<body>

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

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

</body>
</html>
```



Reacting to Events

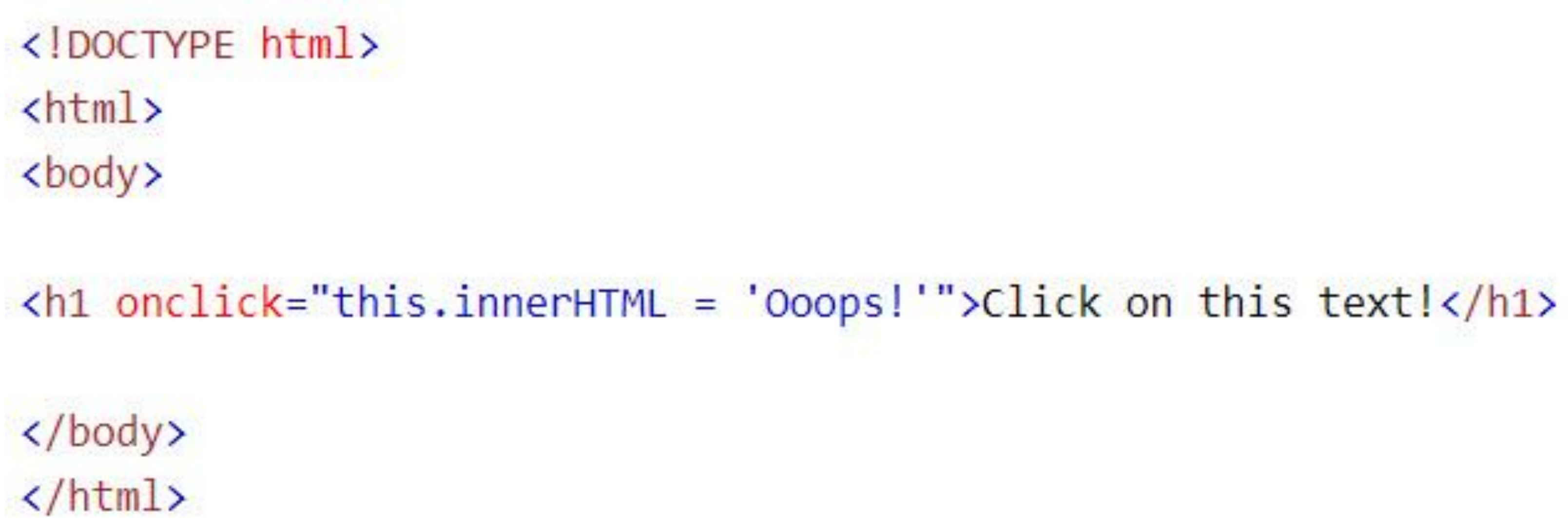
- A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.
- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:
  - Example: onclick=*JavaScript*



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

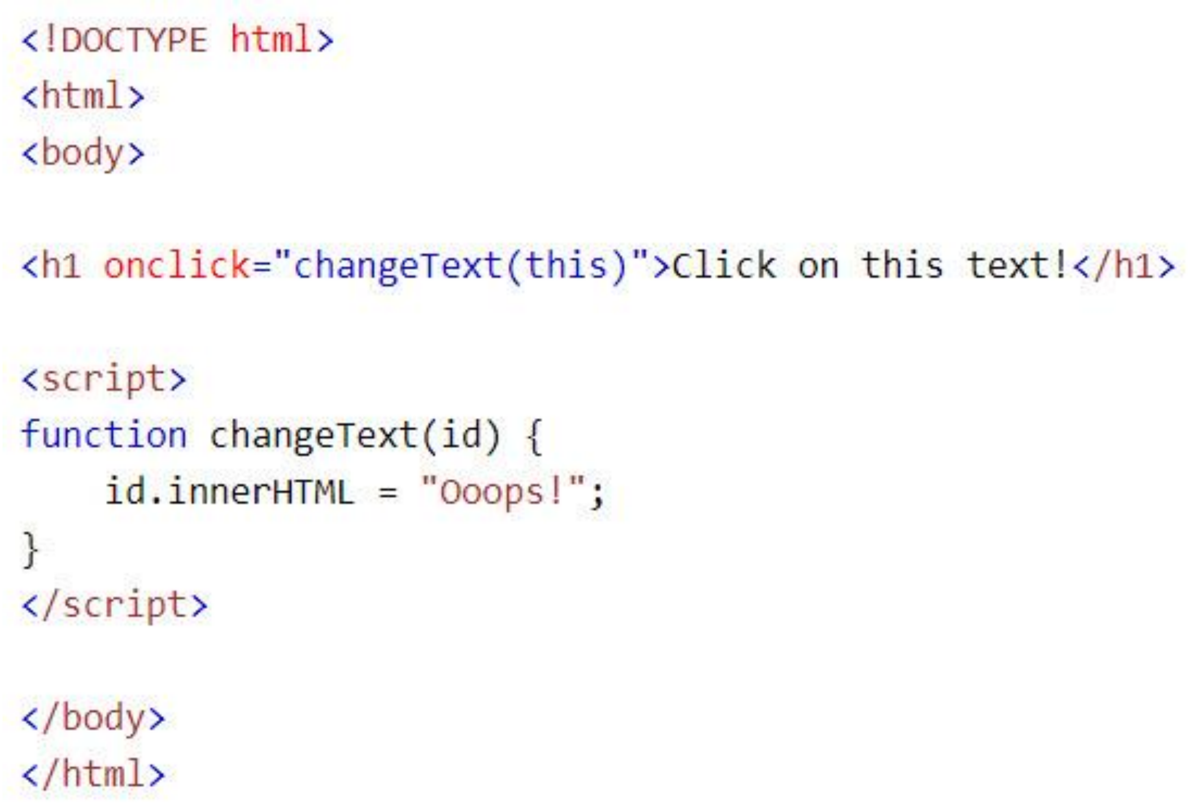
Demo



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

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

</body>
</html>
```

JS Events

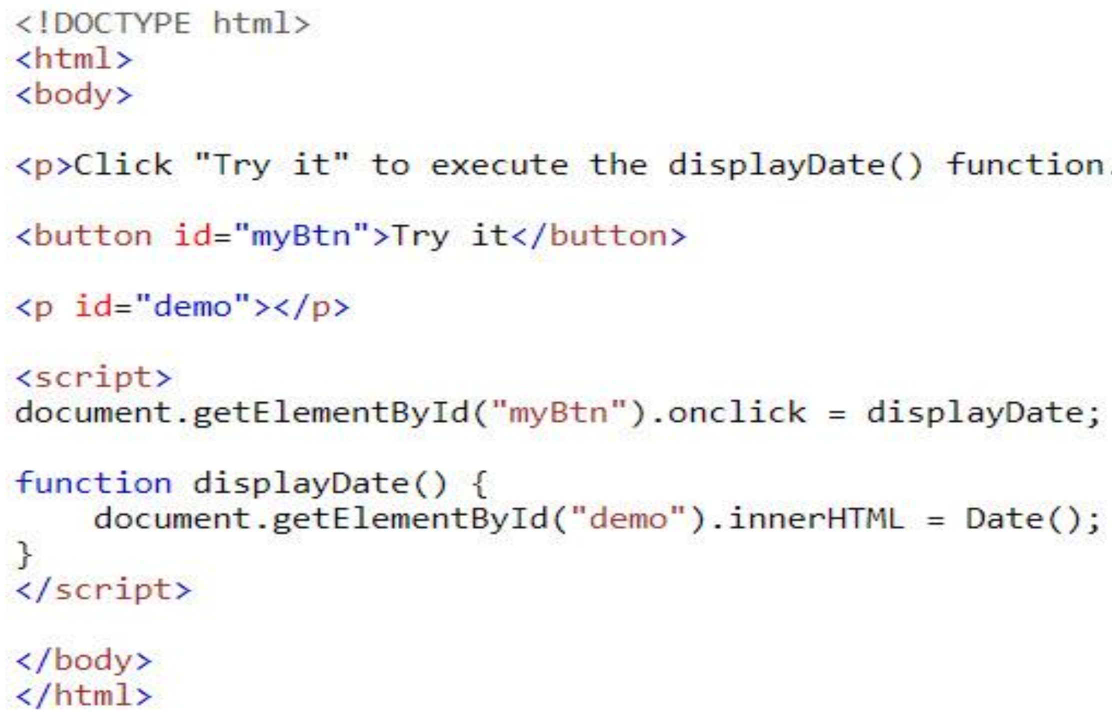
- We can associate an event with an event handler



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

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

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

</body>
</html>
```

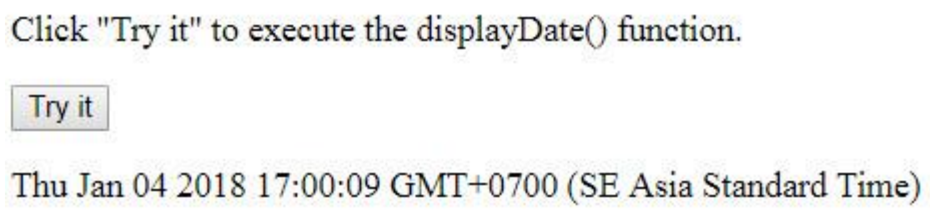
Assign Events Using the HTML DOM

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



```
<!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>
```

Output:



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

Try it

Thu Jan 04 2018 17:00:09 GMT+0700 (SE Asia Standard Time)
```

List of events

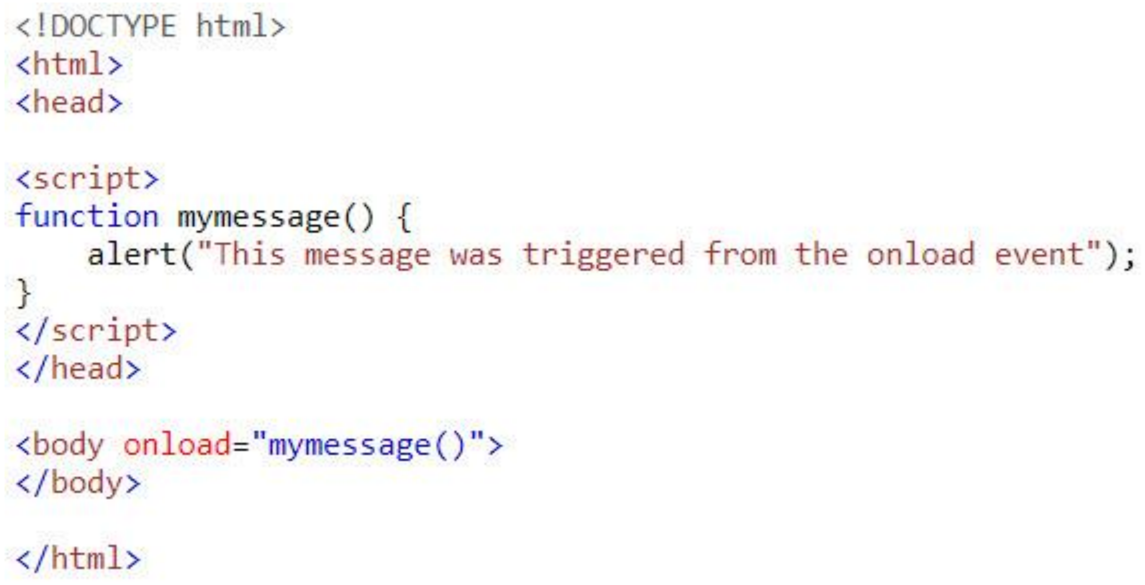
- The onload and onunload events are triggered when the user enters or leaves the page.
- The onchange event is often used in combination with validation of input fields.
- The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element.
- The onfocus event: is triggered when user put focus into a control.
- The onblur event: is triggered when user leave from a control.

List of events

- The onresize event: is triggered when user resize a web page.
- The onsubmit event: is triggered when user submit form.
- 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.

JS Events

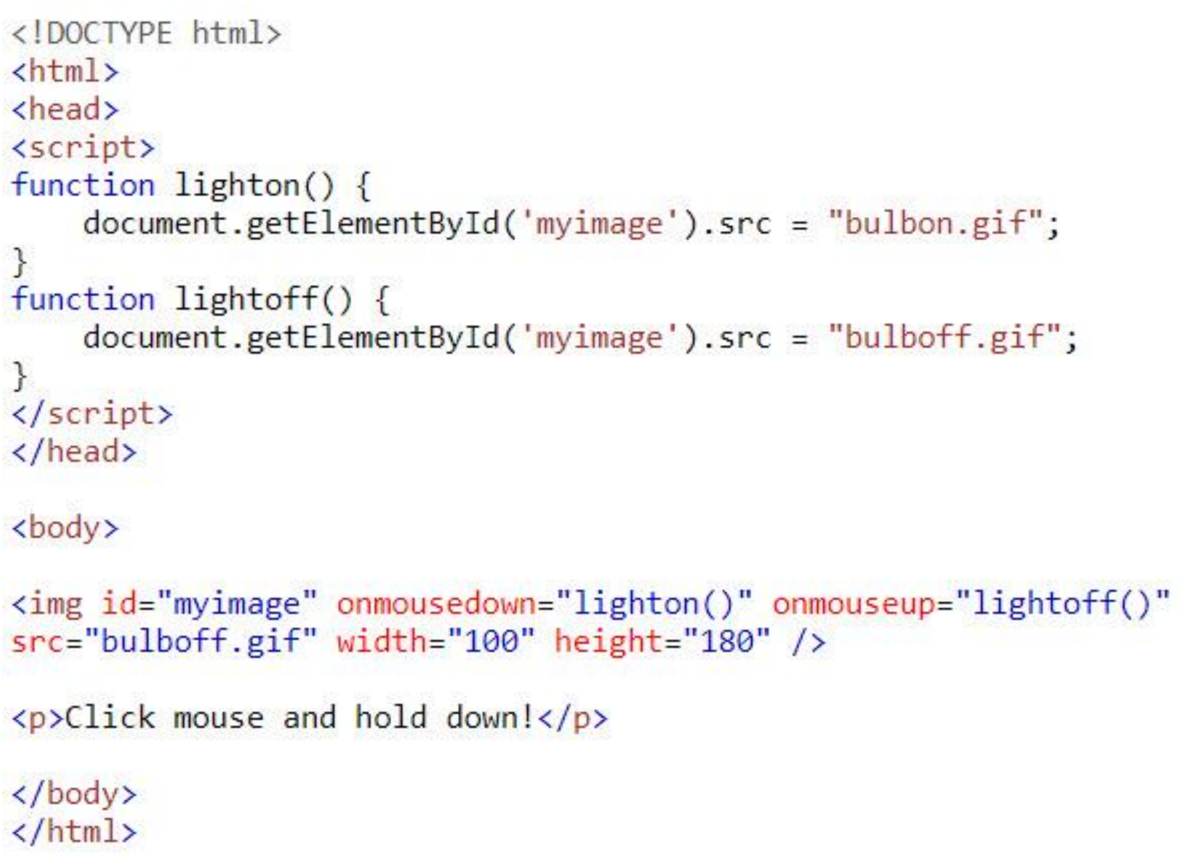
- Example of onload event:



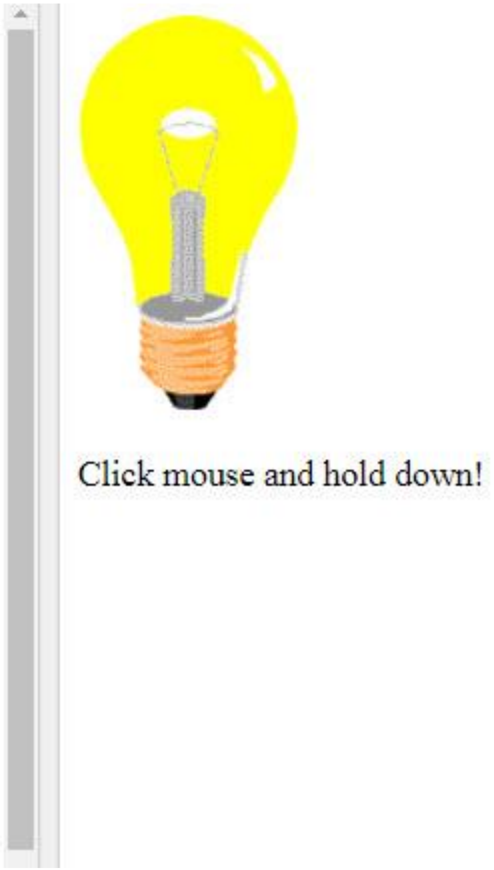
```
<!DOCTYPE html>
<html>
<head>

<script>
function mymessage() {
    alert("This message was triggered from the onload event");
}
</script>
</head>

<body onload="mymessage()">
</body>

</html>
```

JS Events

- Example of mouse events:



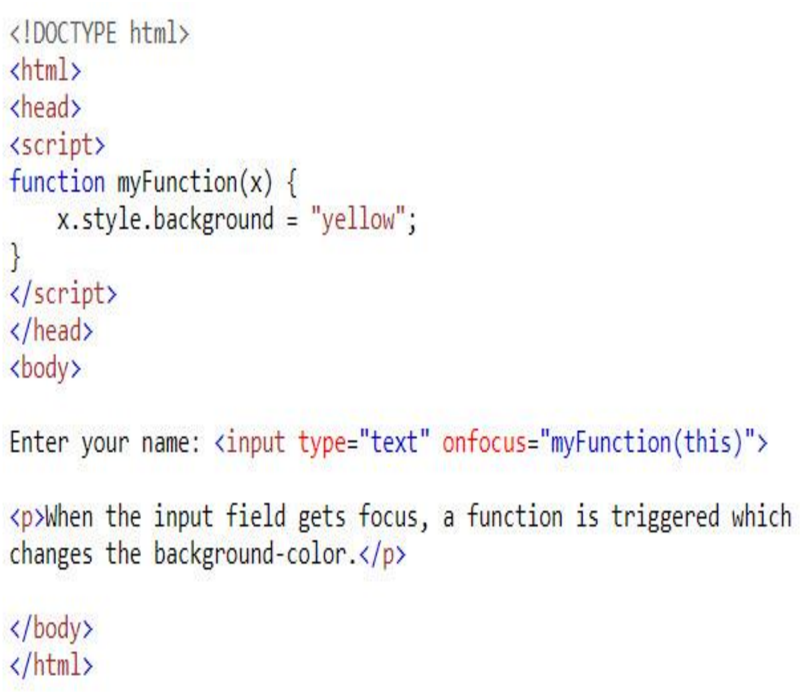
```
<!DOCTYPE html>
<html>
<head>
<script>
function lighton() {
    document.getElementById('myimage').src = "bulbon.gif";
}
function lightoff() {
    document.getElementById('myimage').src = "bulboff.gif";
}
</script>
</head>

<body>

<img id="myimage" onmousedown="lighton()" onmouseup="lightoff()"
src="bulboff.gif" width="100" height="180" />

<p>Click mouse and hold down!</p>

</body>
</html>
```



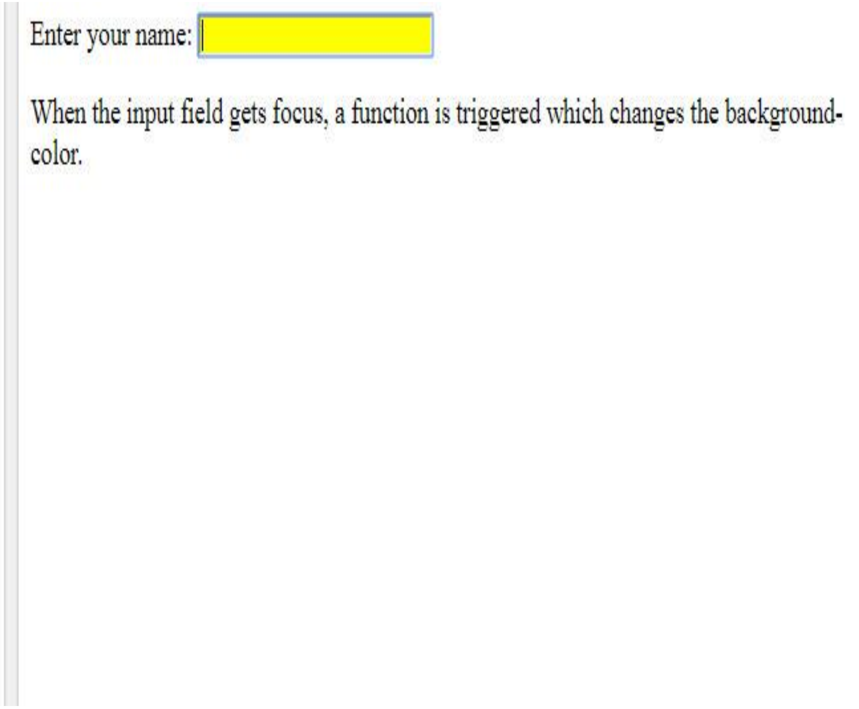
Click mouse and hold down!

JS Events

- Example of onfocus events:



```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction(x) {
    x.style.background = "yellow";
}
</script>
</head>
<body>

Enter your name: <input type="text" onfocus="myFunction(this)">

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

</body>
</html>
```

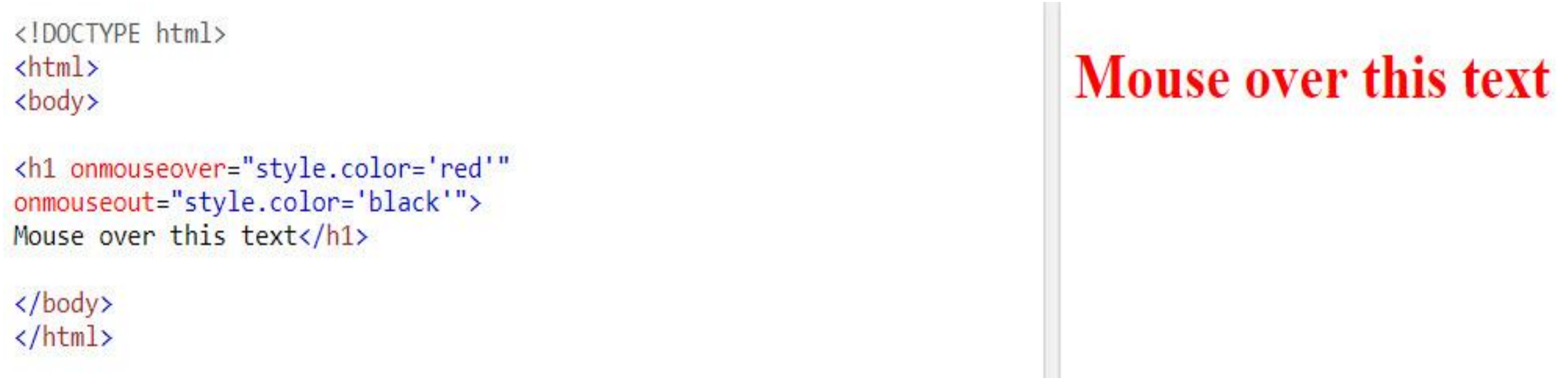


Enter your name:

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

JS Events

- Example of onmouseover/onmouseout events:

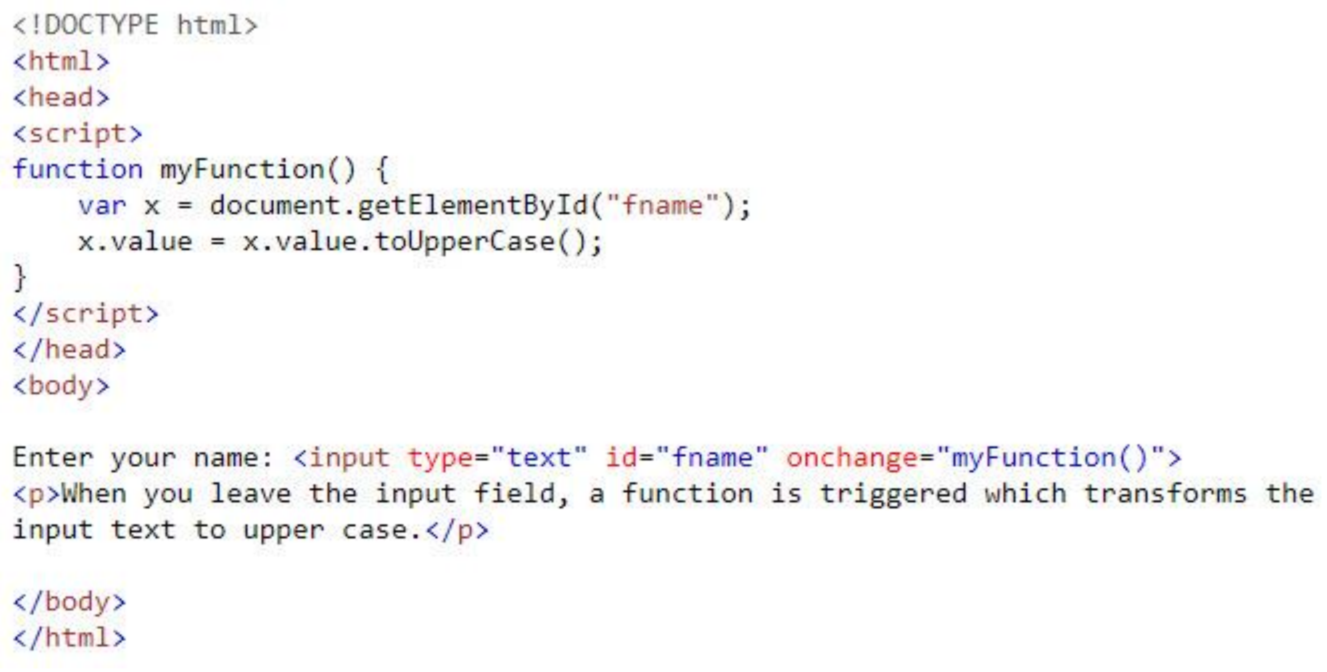


```
<!DOCTYPE html>
<html>
<body>

<h1 onmouseover="style.color='red'"
onmouseout="style.color='black'">
Mouse over this text</h1>

</body>
</html>
```

JS Events

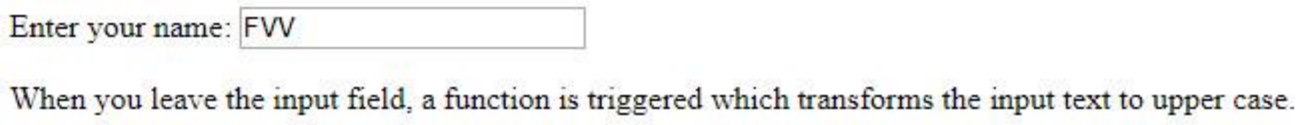
- Example of onchange event:

Code:



```
<!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>
```

Output:



```
Enter your name: FVV

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

Data Validation

- Data validation is the process of ensuring that user input is clean, correct, and useful.
- Typical validation tasks are:
  - has the user filled in all required fields?
  - has the user entered a valid date?
  - has the user entered text in a numeric field?
- Most often, the purpose of data validation is to ensure correct user input.
- Validation can be defined by many different methods, and deployed in many different ways.
  - **Server side validation** is performed by a web server, after input has been sent to the server.
  - **Client side validation** is performed by a web browser, before input is sent to a web server.

Demo

- Demo validate HTML form using JavaScript

Summary

- In this session, you learned that:
  - JavaScript Arrays
  - HTML DOM Model
  - List of events in JavaScript
  - Validation form using JavaScript