JavaScript

Dynamic Behavior in a Web Page

Dr. Jenila Livingston L.M.
VIT Chennai

Presentation Overview

1. Introduction
2. Using JavaScript
   - The First Script
   - Internal Script
     - Event Handler
   - External Script
3. Other Elements
   - noscript
   - Deferred Script
   - Dynamically Loaded Scripts
4. Three methods or Popup boxes
5. JAVASCRIPT INPUT-OUTPUT

1. JavaScript - Introduction

- JavaScript is a front-end scripting language developed by Netscape for dynamic content
  - Lightweight, but with limited capabilities
  - Can be used as object-oriented language
- Client-side technology
  - Embedded in your HTML page
  - Interpreted by the Web browser
- Simple and flexible
- Can read and write HTML elements and Powerful to manipulate the DOM

JavaScript Advantages

- JavaScript allows interactivity such as:
  - Implementing form validation
  - React to user actions, e.g. handle events
    - Changing an image on moving mouse over it
    - Sections of a page appearing and disappearing
  - Can handle exceptions
  - Content loading and changing dynamically
  - Performing complex calculations
  - Can access / modify browser cookies
  - Custom HTML controls, e.g. scrollable table
  - Implementing AJAX functionality (asynchronous server calls)

JavaScript Vs Java

JavaScript	Java
Interpreted by the client-side computer	Compiled on the server before executed on the client machine
Dynamic binding, object references are checked at runtime	Static binding, object references must exist at compile time
No need to declare data types	Data types must be declared
Code is embedded in HTML	Code is not integrated in HTML
Limited by the browser functionality	Java applications are standalone
Can access browser objects	Java has no access to browser objects

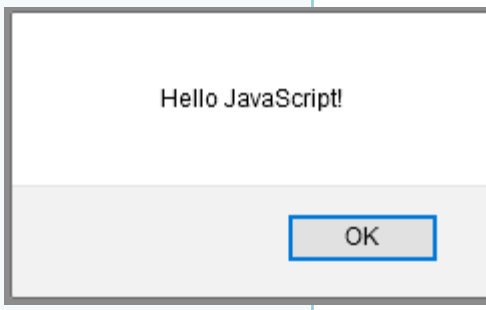
Using JavaScript
2.1 The First Script

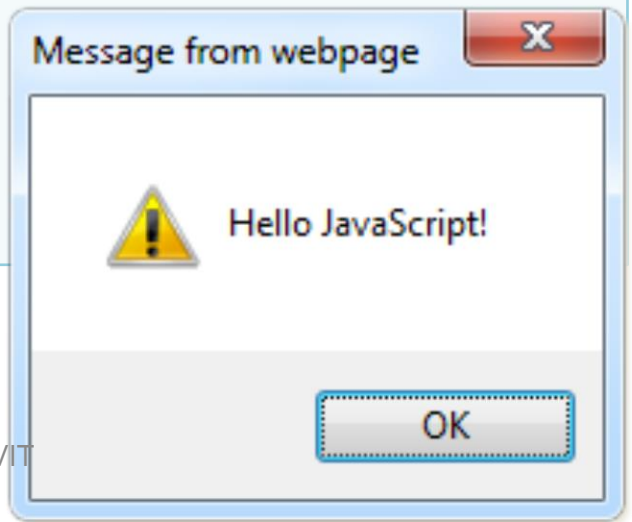
first-script.html

```
<html>

<head>
  <script type="text/javascript">
     alert('Hello JavaScript!');
  </script>
</head>

</html>
```





2.2 Internal Script

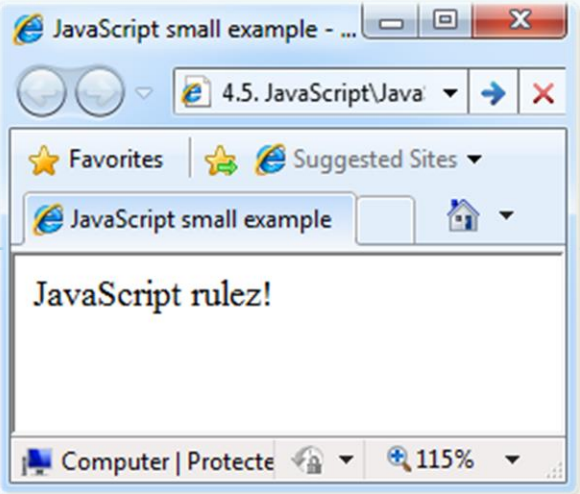
internal-script.html

- `<script>` tag in the head
- `<script>` tag in the body – not recommended

```
<html>

<head>
  <script type="text/javascript">
     document.write('JavaScript rulez!');
  </script>
</head>

</html>
```



Internal Script

Embedding HTML tags with document.write

internal-script.html

```
<html>

<head>
<script>
document.write("<h1 style=color:blue;text-
align:center;>Hello World</h1>");
</script>
</head>

</html>
```

JavaScript – When is Executed?

- JavaScript code is executed during the **page loading** or when the browser fires an **event**
  - All statements are executed at page loading
  - Some statements just define functions that can be called later
- Function calls or code can be attached as "event handlers" via tag attributes
  - Executed when the event is fired by the browser

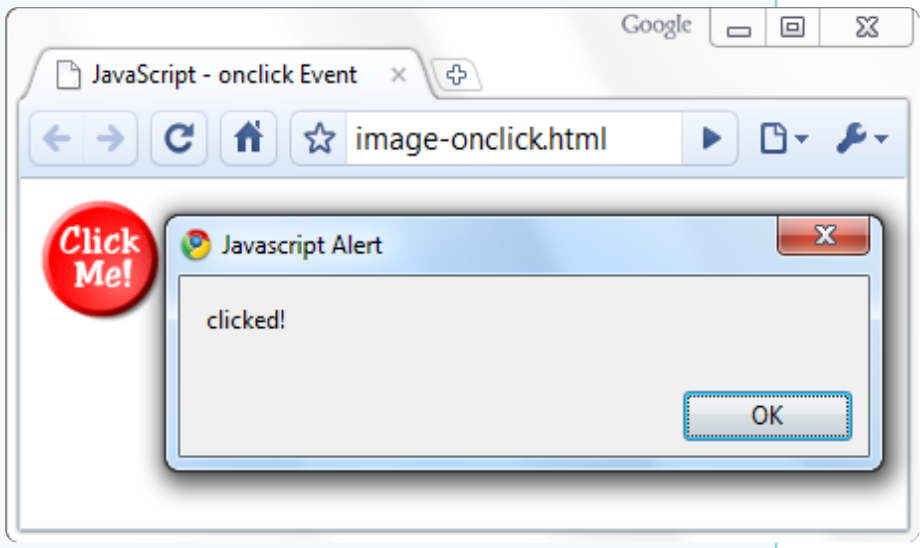
```
<img src="logo.gif" onclick="alert('clicked!')" />
```

Calling a JavaScript Function from Event Handler – Example

image-onclick.html

```
<html>
<head>
<script type="text/javascript">
  function test() {
      alert('clicked!');
  }
</script>
</head>

<body>
  <img src="logo.gif"
    onclick="test()" />
</body>
</html>
```



HTML Event Attributes

Event	Description
onchange	An HTML Element has been changed
onclick	The user clicks an HTML Element
onmouseover	The user moves the mouse over the HTML Element
onmouseout	The user moves the mouse away from the HTML Element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

2.3 External Script

- The JavaScript code can be placed in:
  - External files, linked via `<script>` tag the head
    - Files usually have `.js` extension

```
<script src="scripts.js" type="text/javscript">
<!- code placed here will not be executed! -->
</script>
```

    - Highly recommended
    - The `.js` files get cached by the browser

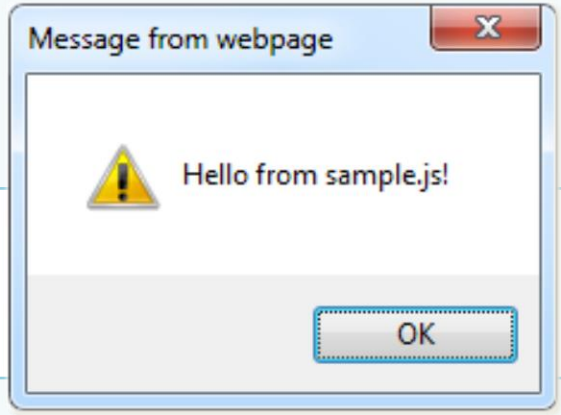
External Script – Function Call

- Using external script files:

external-JavaScript.html

```
<html>
<head>
  <script src="sample.js" type="text/javascript">
  </script>
</head>
<body>
  <button onclick="sample()" value="Call external
JavaScript function" />
</body>
</html>
```

The <script> tag is always empty.



- External JavaScript file:

```
function sample() {
  alert('Hello from sample.js!')
}
```

sample.js

Area of Rectangle– Internal JavaScript

arearect.html

```
<html>

<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript">
    function area() {
      length = parseInt(document.F1.t1.value);
      breadth = parseInt(document.F1.t2.value);
      area = length * breadth;
      alert(area);
    }
  </script>
</head>
```

Area of Rectangle– Internal JavaScript

Arearect.html(contd..)

```
<body>
  <form name=“F1">
    <input type="text" name="t1" /> <br/>
    <input type="text" name="t2" /> <br/>
    <input type="button" value="Process"
      onclick=“area()" />
  </form>
</body>

</html>
```

Area of Rectangle– External JavaScript1

Arearect.html

```
<html>
<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript" src="rect.js">
  </script>
</head>
<body>
  <form name="F1">
    <input type="text" name="t1" /> <br/>
    <input type="text" name="t2" /> <br/>
    <input type="button" value="AreaCalc"
      onclick="area()" />
  </form>
</body>
</html>
```

Area of Rectangle– External JavaScript1

rect.js

Using name.value

```
function area() {
      length = parseInt(document.F1.t1.value);
      breadth = parseInt(document.F1.t2.value);
      area = length * breadth;
      alert(area);
    }
```

Area of Rectangle– External JavaScript1

Arearect.html

```
<html>
<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript" src="rect.js">
  </script>
</head>
<body>
  <form name="F1">
    <input type="text" id="t1" /> <br/>
    <input type="text" id="t2" /> <br/>
    <input type="button" value="AreaCalc"
      onclick="area()" />
  </form>
</body>
</html>
```

Area of Rectangle– External JavaScript2

getElementById

rect.js

```
function area() {
   length = parseInt(document.getElementById("t1"));
   breadth = parseInt(document.getElementById("t2"));
    area = length * breadth;
    alert(area);
 }
```

Advantage of using external script

- Once an external script is downloaded it is kept in memory.
- The next time the page loads it refers to it.
- There is no need to re-download the script every time the page is loaded.
- For large scripts it suggested to make it external.

3. Other Elements
3.1 <noscript> Element

- Sometimes JavaScript can be disabled in browsers.
- In such case the alternate content can be specified in the <noscript> block.

<noscript> Element

```
<html>
<body>
<script type="text/javascript">
alert("Hello");
</script>
<noscript>JavaScript is disabled</noscript>
</body>
</html>
```

3.2 Deferred Script

- The defer attribute of the script tag delays the execution of the script after the DOM has been loaded.
- The DOM-Document Object Model is the object representation of all tags and details of the layout page.
- Deferred script works only for external script.

Deferred script Demo

```
<html>
<head>
<script type="text/javascript" src="script.js" defer="defer">
</script>
</head>
<body>
<h1>HELLO WORLD</h1>
</body>
</html>
```

```
alert("welcome");
```

3.3 Dynamically Loaded Scripts

- To load and run an external JavaScript we use src attribute in script tag.
- Consider a situation where you need to choose between two JavaScript files at the time of loading.
- So the script file has to be loaded dynamically.

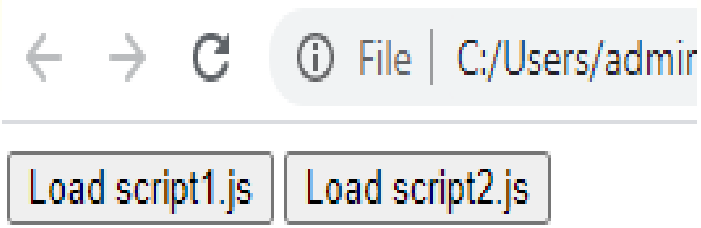
Dynamically Loaded Scripts

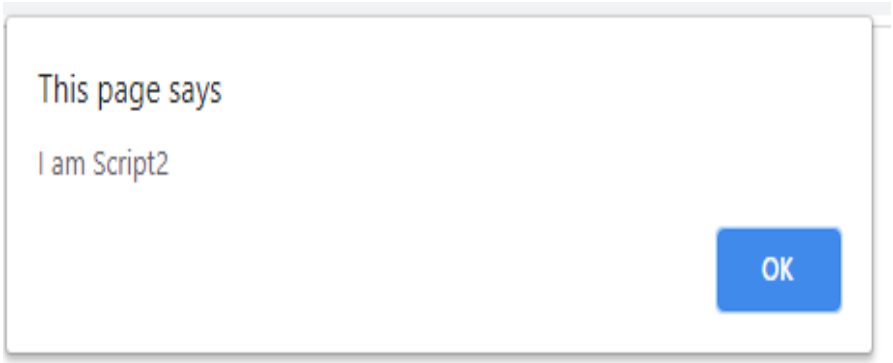
```
<script type="text/javascript">
function dynLoad(file) {
var selm = document.createElement("script");
selm.type = "text/javascript";
selm.src = file;
document.body.appendChild(selm);
}
</script>
<button onclick="dynLoad('script1.js');">Load script1.js</button>
<button onclick="dynLoad('script2.js');">Load script2.js</button>
```

```
alert("I am Script1");
```

```
alert("I am Script2");
```

Dynamically Loaded Scripts





Semicolon

- Semicolon specifies the end of a statement.
- Semicolon can be omitted if a line break is used.

4. Standard Popup Boxes

- Alert box with text and [OK] button
  - Just a message shown in a dialog box:

```
alert("Some text here");
```

- Confirmation box
  - Contains text, [OK] button and [Cancel] button:

```
confirm("Are you sure?");
```

- Prompt box
  - Contains text, input field with default value:

```
prompt ("enter amount", 10);
```

Using the alert() Method

```
<head>
<script language="JavaScript">
        alert("An alert triggered by JavaScript");
</script>
</head>
```

- It is the easiest methods to use amongst alert(), prompt() and confirm().
- You can use it to **display textual information** to the user (simple and concise).
- The user can simply click "OK" to close it.

Using the confirm() Method

```
<head>
<script language="JavaScript">
        confirm("Are you happy with the class?");
</script>
</head>
```

- This box is used to give the user a choice either **OK or Cancel**.
- It is very similar to the "alert()" method.
- You can also put your message in the method.

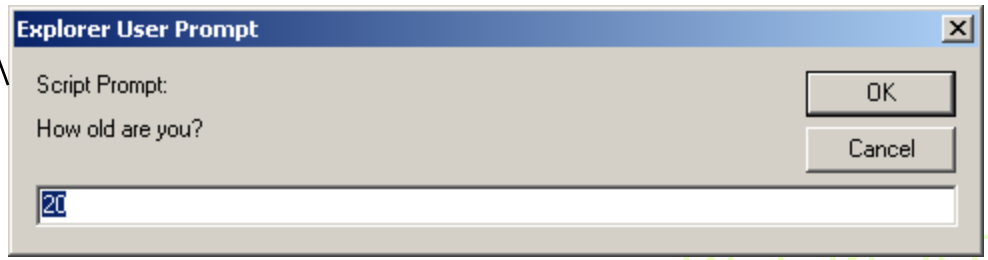
Using the prompt() Method

```
<head>
<script language="JavaScript">
        prompt("What is your student id number?");
        prompt("What is your name?","No name");
</script>
</head>
```

- This is the only one that allows the user to type in his own response to the specific question.
- You can give a default value to avoid displaying "undefined".

Three methods

```
<script language="JavaScript">
alert("This is an Alert method");
confirm("Are you OK?");
prompt("What is your name?");
prompt("How old are you?","20");
</script>
```



5. JAVASCRIPT INPUT-OUTPUT

Variable Declaration

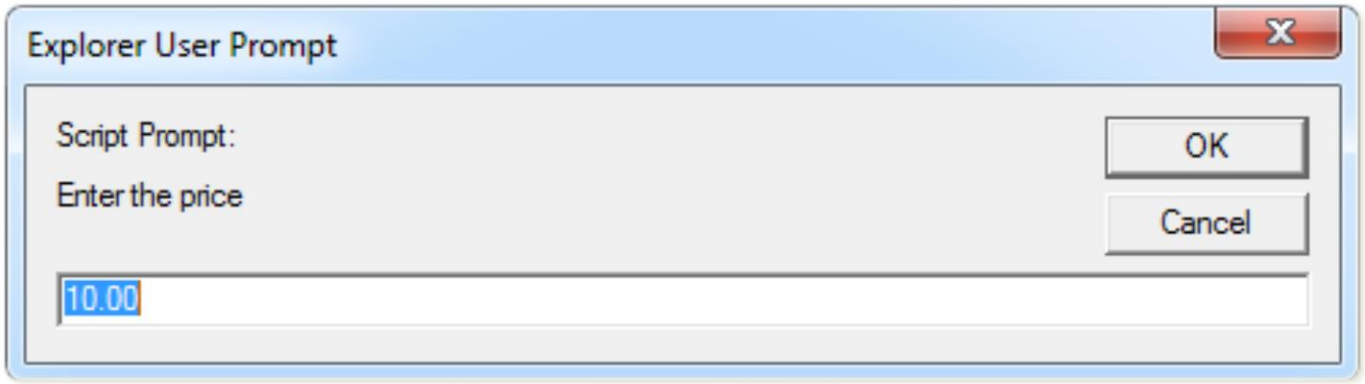
```
<head>
<script language="JavaScript">
        var id;
        id = prompt("What is your student id number?");
        alert(id);
        name = prompt("What is your name?","No name");
        alert(name);
</script>
</head>
```

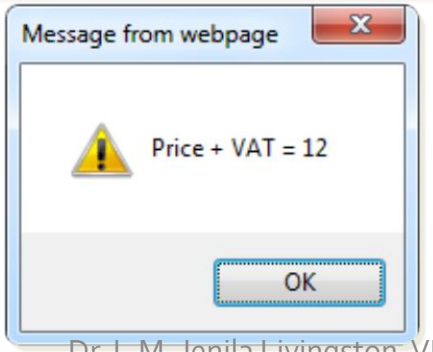
Variable declaration

Input-Output using Prompt & Alert

prompt.html

```
price = prompt("Enter the price", "10.00");
alert('Price + VAT = ' + price * 1.2);
```





Input-Output using Text boxes - Sum of Numbers

sum-of-numbers.html

```
<html>

<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript">
    function calcSum() {
      value1 =
        parseInt(document.mainForm.textBox1.value);
      value2 =
        parseInt(document.mainForm.textBox2.value);
      sum = value1 + value2;
      document.mainForm.textBoxSum.value = sum;
    }
  </script>
</head>
```

Note:
parseInt: Convert string to integer
parseFloat: Convert string to float

Sum of Numbers – Contd..

sum-of-numbers.html (cont.)

```
<body>
  <form name="mainForm">
    <input type="text" name="textBox1" /> <br/>
    <input type="text" name="textBox2" /> <br/>
    <input type="button" value="Process"
      onclick="javascript:calcSum()" />
    <input type="text" name="textBoxSum"
      readonly="readonly"/>
  </form>
</body>

</html>
```

Input-Output using Text boxes - Sum of Numbers - getElementById

sum-of-numbers.html

```
<html>

<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript">
    function calcSum() {
     value1 = document.getElementById(“t1”);
     value2 = document.getElementById(“t2”);
      sum = value1 + value2;
      document.getElementById(“t3”).value = sum;
    }
  </script>
</head>
```

Note:
parseInt: Convert string to integer
parseFloat: Convert string to float

Sum of Numbers – Contd..

sum-of-numbers.html (cont.)

```
<body>
  <form name="mainForm">
    <input type="text" name="text1" id="t1" />
<br/>
    <input type="text" name="text2" id="t2" />
<br/>
    <input type="button" value="Process"
      onclick="calcSum()" />
    <input type="text" name="textBoxSum" id="t3"
      readonly="readonly"/>
  </form>
</body>

</html>
```

JavaScript OUTPUT

1. Writing into the HTML output using document.write()/ document.writeln.
2. Writing into an HTML element, using innerHTML.
3. Writing into text box
4. Writing into an alert box, using window.alert().

Document.write

```
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
document.write(5 + 6);
</script>
</body>
</html>
```

The **writeln**() method is identical to the **document**. **write()** method, with the addition of writing a newline character after each statement.

innerHTML

- JavaScript can use the **document.getElementById(id)** method.

```
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
</body>
</html>
```

The **id** attribute defines the HTML element.

Writing into text box

```
value1 = document.getElementById("t1");
value2 = document.getElementById("t2");
 sum = value1 + value2;
 document.getElementById("t3").value = sum;
  //document.mainForm.textBoxSum.value = sum;
```

inner.htmlOutput using innerHTML

```
<html> <head>
  <title>JavaScript Demo</title>
  <script type="text/javascript">
    function calc() {
    v=document.getElementById(“t1”);
    document.getElementById(‘display’).innerHTML=v;}
</head>
<body>
  <form name="mainForm">
    <input type="text" name=“text1” id=“t1” />
<input type="button" value="Process"
      onClick="calc()" />
</form>
  <p id='display'>
</body> </html>
```

Windows.alert

-

```
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
window.alert(5 + 6);
</script>
</body>
</html>
```

Exercise

BMI = KG / (H/100 * H/100)

Height in CM 170

Weight in KG 70

Submit

Under Weight = Less than 18.6
Normal Range = 18.6 and 24.9
Overweight = Greater than 24.9

Thank You