

Web Technologies (ICDT6004) – Lectures 11 and 12

Introduction to JavaScript

Aims of lecture

- At the end of today's lecture students should be able to
 - Write JavaScript code for functions, for, while, do while and for/in loops
 - Use JavaScript to add interactivity to a website

JavaScript

- JavaScript is a client-side scripting language.
- A scripting language is a lightweight programming language.
- A client-side script is a script that runs on the client i.e. on the user's machine, namely the browser.
- Examples of client-side scripting are:
 - Javascript
 - Vbscript

JavaScript (2)

- JavaScript is programming code that can be inserted into HTML pages.
- JavaScript inserted into HTML pages, can be executed by all modern web browsers.
- JavaScript can be used to write to the browser by using – `document.write`

Example 1

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<script>
```

```
document.write("Hello World!")
```

```
</script>
```

```
</body>
```

```
</html>
```

Example 1- explanation

- In the above example, JavaScript writes into the HTML <body> while the page loads.
- document.write is known as a statement
- The purpose of the statement is to tell the browser what to do.
- document object allows access to the whole page
- The write() method is mostly used for testing: If it is used after an HTML document is fully loaded, it will delete all existing HTML.

Where to insert the JavaScript code?

- JavaScripts in HTML must be inserted between `<script>` and `</script>` tags.
- The `<script>` and `</script>` tells where the JavaScript starts and ends.
- Old examples may have `type="text/javascript"` in the `<script>` tag. This is no longer required. JavaScript is the default scripting language in all modern browsers and in HTML5.
- Scripts can be in the `<body>` or in the `<head>` section of HTML, and/or in both.

Where to insert the JavaScript code?(2)

- More often, we want to execute code when an **event** occurs, like when the user clicks a button.
- If we put JavaScript code inside a **function**, we can call that function when an event occurs.
- You can place an unlimited number of scripts in an HTML document.
- It is a common practice to put functions in the `<head>` section, or at the bottom of the page. This way they are all in one place and do not interfere with page content.

Example 2 – Using JavaScript to write into HTML

```
<html>
<body>
<script>
//write to the browser
document.write("Hello!")
document.write("<BR>")
document.write("Welcome To My WebSite")
</script>
</body>
</html>
```

Comments in JavaScript

- Comments will not be executed by JavaScript.
- Comments can be added to explain the JavaScript, or to make the code more readable.
- Single line comments start with `//`.

Variables in JavaScript

- A variable is a "container" for information you want to store.
- A variable's value can change during the script.
- You can refer to a variable by name to see its value or to change its value.
- When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

Variables in JavaScript (2)

- If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.
- Variables in JavaScript are case-sensitive and must begin with a letter or \$ or underscore.

`var name;`

`name = "John";`

- Can be combined into 1 statement:

`var name = "John";`

Data Types

- JavaScript variables can contain data like text values (person="John Doe"). This is known as a string.
- There are many types of JavaScript variables, but for now, just think of numbers and strings.
- When you assign a text value to a variable, put double or single quotes around the value.
- When you assign a numeric value to a variable, do not put quotes around the value. If you put quotes around a numeric value, it will be treated as text.

Example 3

```
<HTML>
```

```
<BODY>
```

```
<script>
```

```
var moduleName = "Web design"
```

```
document.write("<h1>" + moduleName + "</h1>")
```

```
</script>
```

```
<p>This example declares a variable, assigns a value  
to it, and then displays the variable.</p>
```

```
</BODY>
```

```
</HTML>
```

Writing to a paragraph

```
<!DOCTYPE html>
<html>
<body>
<h2>Writing to a div using JavaScript</h2>
<p id="demo"></p>
<script>
//making use of DOM – Document Object Model
document.getElementById("demo").innerHTML = "Writing to a
div";
</script>
</body>
</html>
```

Loops

JavaScript supports different kinds of loops:

- **for** - loops through a block of code a number of times
- **for/in** - loops through the properties of an object
- **while** - loops through a block of code while a specified condition is true
- **do/while** - also loops through a block of code while a specified condition is true

The for loop

```
<!DOCTYPE html>
<html><body>
<h2>JavaScript Loops</h2>
<p id="demo"></p>
<script>
var text = "";
var i;
for (i = 0; i < 5; i++) {
    text = text + "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script></body></html>
```

The for/in loop

```
<!DOCTYPE html><html><body>
<h2>JavaScript Loops</h2>
<p>The for/in statement loops through the properties of an object.</p>
<p id="demo"></p>
<script>
var txt = "";
var person = {fname:"John", lname:"Doe", age:25};
var x;
for (x in person) {
    txt += person[x] + " ";
}
document.getElementById("demo").innerHTML = txt;
</script></body></html>
```

The while loop

```
<!DOCTYPE html>
<html><body><h2>JavaScript while</h2>
<p id="demo"></p>
<script>
var text = "";
var i = 0;
while (i < 10) {
    text += "<br>The number is " + i;
    i++;
}
document.getElementById("demo").innerHTML = text;
</script>
</body></html>
```

The do-while loop

```
<!DOCTYPE html>
<html><body><h2>JavaScript do ... while</h2>
<p id="demo"></p>
<script>
var text = ""
var i = 0;
do {
    text += "<br>The number is " + i;
    i++;
}
while (i < 10);
document.getElementById("demo").innerHTML = text;
</script></body></html>
```

Object- Oriented

- Almost everything in JavaScript can be an Object: Strings, Functions, Arrays, Dates....
- Objects are just data, with properties and methods.
- Properties are values associated with objects.
- Methods are actions that objects can perform.

Functions

- A function contains some code that will be executed by an event or a call to that function.
- A function is a set of statements. You can reuse functions within the same script, or in other documents.
- You define functions at the beginning of a file (in the head section), and call them later in the document.

Event-Handling

- The JavaScript statements, in the above examples, are executed while the page loads.
- More often, we want to execute code when an **event** occurs, like when the user clicks a button.
- If we put JavaScript code inside a **function**, we can call that function when an event occurs.
- Events describe actions that occur as the result of user interaction with a web page or other browser-related activities. E.g. User clicks a hyperlink, or enters data in a form.
- The browser can capture events as they occur and process any scripts associated to the particular event.

Event-Handling (2)

- *Event Handling* is the processing of the code associated with an event. The code being processed is called an *event handler*.
- One or more events are defined for each HTML element.

```
<button type="button"  
onclick="alert('Welcome!')">Click Me!</button>
```


Event-Handling (3)

link	<code><A> ... </code>	<code>click, dblClick,</code> <code>mouseDown, mouseUp,</code> <code>mouseover, mouseOut,</code> <code>keyDown, keyUp,</code> <code>keyPress</code>
image	<code></code>	<code>abort, error, load,</code> <code>keyDown, keyUp,</code> <code>keyPress</code>
form	<code><form>...</form></code>	<code>submit, reset</code>

Changing HTML Content using JavaScript

- **JavaScript Can Change HTML Content**
- One of many JavaScript HTML methods is **getElementById()**.
- The example that follows uses the method to "find" an HTML element (with id="demo") and changes the element content (**innerHTML**) to "Hello JavaScript":
- JavaScript accepts both double and single quotes.

Example 4

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>What Can JavaScript Do?</h2>
```

```
<p id="demo">JavaScript can change HTML content.</p>
```

```
<button type="button"  
onclick='document.getElementById("demo").innerHTML =  
"Hello JavaScript!"'>Click Me!</button>
```

```
</body>
```

```
</html>
```

Functions

- A function is written as a code block (inside curly { } braces), preceded by the **function** keyword:
- ```
function functionname()
{
 some code to be executed
}
```
- The code inside the function will be executed when "someone" calls the function.
- The function can be called directly when an event occurs (like when a user clicks a button), and it can be called from "anywhere" by JavaScript code.
- JavaScript is case sensitive. The function keyword must be written in lowercase letters, and the function must be called with the same capitals as used in the function name.

# Example 5 – Function without argument

```
<html>
<head>
<script>
function myfunction()
{
alert("HELLO")
}
</script>
</head>
<body><button type="button" onclick= " myfunction()">Click Me!</button>
<p>By clicking on the button, a function will be called. The function will alert
a message.</p>
</body>
</html>
```

## Example 6 – Function with argument

```
<html>
<head><script>
function myfunction(txt)
{
alert(txt)
}
</script></head>
<body><button type="button" onclick= " myfunction('Hello from
JavaScript functions')">Click Me!</button>
<p>By clicking on the button, a function will be called. The
function will alert amessage.</p>
</body></html>
```

# Functions with multiple arguments

- `<button onclick="myFunction('Harry Potter','Wizard')">Try it</button>`

```
<script>
```

```
function myFunction(name,job)
```

```
{
```

```
 alert("Welcome " + name + ", the " + job);
```

```
}
```

```
</script>
```

# Example 7 – Function returning a value

```
<html>
<head><script>
function myFunction(){
return ("Hello, have a nice day!")
}
</script></head>
<body>
<p id="demo"></p>
<script>
txt = myFunction();
document.getElementById("demo").innerHTML = txt;
</script>
<p>The script in the body section calls afunction.</p>
<p>The function returns a text.</p></body></html>
```



# External Javascript Files

- Scripts can also be placed in external files. External files often contain code to be used by several different web pages.
- External JavaScript files have the file extension .js.
- To use an external script, point to the .js file in the "src" attribute of the `<script>` tag:
- `<!DOCTYPE html>`

```
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Note: We cannot write code inside the script tag here - we need to open another `<script>` tag



## Example 8 – Displaying a prompt

```
<html>
<head>
</head>
<body>
<p id="demo"></p>
<script >
var name = prompt("Please enter your name","")
if (name != null && name != "")
{
document.getElementById("demo").innerHTML =
name;
}
</script>
</body>
</html>
```

# Classwork 1

- Write the javascript code to prompt a user to enter a first number, prompts the user to enter a second number and displays the sum on the page.
- Note: You will have to use a function `sum(x,y)` which returns a value.

# JavaScript can change HTML attributes

```
<!DOCTYPE html>
```

```
<html><body>
```

```
<h2>What Can JavaScript Do?</h2>
```

```
<p>JavaScript can change HTML attributes.</p>
```

```
<p>In this case JavaScript changes the src (source) attribute of an
image.</p>
```

```
<button
onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Tur
n on the light</button>
```

```

```

```
<button
onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Tur
n off the light</button></body></html>
```

# JavaScript Can Change HTML Styles (CSS)

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>What Can JavaScript Do?</h2>
```

```
<p id="demo">JavaScript can change the style of an HTML
element.</p>
```

```
<button type="button"
onclick="document.getElementById('demo').style.fontSize='35
px'">Click Me!</button>
```

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

# JavaScript Can Change HTML Styles (CSS)(2)

```
<!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>
```

# JavaScript Can Hide HTML Elements

```
<!DOCTYPE html>
```

```
<html><body>
```

```
<h2>What Can JavaScript Do?</h2>
```

```
<p id="demo">JavaScript can hide HTML
elements.</p>
```

```
<button type="button"
onclick="document.getElementById('demo').style.display='none'">Click Me!</button>
```

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

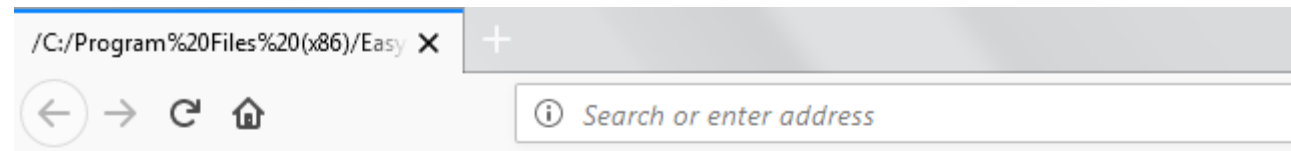
# JavaScript Can Show HTML Elements

```
<!DOCTYPE html>
<html>
<body>
<h2>What Can JavaScript Do?</h2>
<p>JavaScript can show hidden HTML elements.</p>
<p id="demo" style="display:none">Hello JavaScript!</p>
<button type="button"
onclick="document.getElementById('demo').style.display='block'">Click Me!</button>
</body></html>
```



# Classwork2

- Write the code for the page cwk2.html given that clicking on Show Text makes the text 'Hello!' appear in the div, demo, and clicking on Hide Text makes the text disappear.



## What Can JavaScript Do?


JavaScript can show hidden HTML elements.

Show Text

Hide Text

# Arrays

```
<!DOCTYPE html>
<html><body>
<h2>JavaScript Arrays</h2>
<p id="demo"></p>
<script>
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;
</script>
</body></html>
```



# Difference between arrays and objects

- In JavaScript, **arrays** use **numbered indexes**.
- In JavaScript, **objects** use **named indexes**.
- Arrays are a special kind of objects, with numbered indexes.  
Preetisha Gungadoo
- JavaScript does not support associative arrays.
- You should use **objects** when you want the element names to be **strings (text)**.
- You should use **arrays** when you want the element names to be **numbers**.

# How to capture/set the value of a textfield

- `document.formName.textfieldname.value`

- reads value in textfield



- `document.formName.textfieldname.value = "something"`

- writing to the textfield

- Classwork

```
<form name="frm_login">
```

```
Username:<input type="text" name="txt_name"/>
```

```
</form>
```

- How do you read the value of username in Javascript?

```
document.frm_login.txt_name.value
```

```
document.forms[0].txt_name.value
```

# Example 9 – Generating random numbers and links

```
<!DOCTYPE html>
```

```
<html><body>
```

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



```
<script>
```

```
var r=Math.random();
```

```
var x=document.getElementById("demo")
```

```
if (r>0.5){
```

```
x.innerHTML="Visit W3Schools";
```

```
}
```

```
else{
```

```
x.innerHTML="Visit UoM Website";
```

```
}
```

```
</script>
```

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

# Classwork 3

- Write the code for the page cw2.html given the following:
  - When the page is loaded, the user is prompted to enter a number n
  - A message as follows is displayed n times

Number: 1

Number: 2

.

.

# Example 10- Simple animation

```
<html>
<head>
<title>JavaScript Animation</title>
<script>
var imgObj = null;
function init(){
 imgObj =
document.getElementById('myImage');
 imgObj.style.position= 'relative';
 imgObj.style.left = '0px';
}
function moveRight(){
 imgObj.style.left =
parseInt(imgObj.style.left) + 10 + 'px';
}
window.onload =init;
</script>
</head>
<body>
<form>

<p>Click button below to move the
image to right</p>
<input type="button" value="Click
Me" onclick="moveRight();" />
</form>
</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:
- `setTimeout(function, milliseconds)`
  - Executes a function, after waiting a specified number of milliseconds.
  - The **window.setTimeout()** method can be written without the window prefix.
  - The first parameter is a function to be executed.
  - The second parameter indicates the number of milliseconds before execution.
- `setInterval(function, milliseconds)`
  - Same as `setTimeout()`, but repeats the execution of the function continuously.
  - The **window.setInterval()** method can be written without the window prefix.
  - The first parameter is the function to be executed.
  - The second parameter indicates the length of the time-interval between each execution.



# Classwork 4

- Modify the above code such that clicking on the button simply starts the animation and moves the image to the right by 10px every 20s. There is also a stop button that stops the animation.

# 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 **Document Object Model** of the page.
- 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

# Finding HTML Elements

- Often, with JavaScript, you want to manipulate HTML elements.
- To do so, you have to find the elements first. There are a couple of ways to do this:
  - Finding HTML elements by id - `document.getElementById`
  - Finding HTML elements by tag name - `document.getElementsByTagName`
  - Finding HTML elements by class name - `document.getElementsByClassName`
  - Finding HTML elements by CSS selectors - `document.querySelectorAll`
  - Finding HTML elements by HTML object collections – `document.forms[0]`

# References

- CSE 2003Y Lecture Notes
- W3Schools.com website
- <http://www.tutorialspoint.com> website