

# Lesson #12:

# JavaScript I

Introduction

Events

Control Structures

Windows

# What is JavaScript?

- JavaScript is a client-side scripting language. This means the web surfer's browser will be running the script.
- The main benefit of Javascript is to add additional interaction between the website and its visitors.
- JavaScript, despite the name, is unrelated to Java, although both have the common C syntax, and JavaScript copies many Java names and naming conventions.

# A basic script

- JavaScripts are embedded in the HTML page. No need for special extensions as all processing is done client-side.

```
<html>
```

```
<body>
```

```
<script type="text/JavaScript">
```

```
<!--
```

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

```
//-->
```

```
</script>
```

```
</body>
```

```
</html>
```

optional  
;

COMMENTS

HTML  
JavaScript

# Simple redirection

- JavaScript being client-side, the webmaster is dependent to the client's preferences. It is possible to disable JS in the browser settings. The following script will the visitor stuck in JS in not enabled.

```
<script type="text/JavaScript">
```

```
window.location =
```

```
"http://www.mydomain.com/mysite.html"
```

```
</script>
```

# Scripts in <head> part

- Basic scripts are usually placed in the body part where you need them.
- If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head.
- Many times function definitions will take place in the head section, and function calls in the body section.

# Function definition

```
<head>
```

```
<script type="text/JavaScript">
```

```
<!--
```

Used to hide code from older browsers

```
function popup() {
```

```
    alert("Hello World")
```

```
}
```

```
//-->
```

```
</script>
```

```
</head>
```

The alert function brings up a popup box with the text provided in it.

# Function call

```
<body>  
<input type="button" onclick="popup()  
  value="popup">  
</body>  
</html>
```



Event

# External file

- A script can also be placed in an external file bearing the .js extension. In that case the script in that file is not embedded in HTML, it is pure JavaScript.
- Here is an example of the contents of a file called *pop.js*.

```
function popup()  
{  
    alert("Hello World");  
}
```

# External file

- Here are the head and body sections of a page using that external file.

```
<head>
```

```
<script src="pop.js">
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<input type="button" onclick="popup()"  
value="Click Me!">
```

```
</body>
```

# Operators and Variables

- The operators are the same as those you can find in the C/Java/Perl/PHP family.
- + - \* / %
- < <= > >= == !=
- You don't have to declare variables before you use them.
- Sometimes we use the keyword `var` when using a variable the first time in a program. It is not required, just some sort of convention. Naming rules are similar to Perl's or PHP.

```
var greeting = "Good morning!";
```

# Events

- The building blocks of an interactive web page is the JavaScript event system. An event in JavaScript is something that happens with or on the web page.
- JavaScript has predefined names that cover numerous events that can occur.
- To capture an event and make something happen when that event occurs, you must specify the event, the HTML element that will be waiting for the event, and the function(s) that will be run when the event occurs.

# Common Events

- `onload` / `onUnload`: triggered when the user enters or leaves the page.
- `onFocus` / `onBlur` / `onChange`: triggered when an object is activated (`onFocus`), loses focus (`onBlur`) and when it changes value (`onChange`). Usually goes with form objects.
- `onSubmit`: triggered when a form is submitted. Used to validate a form.
- `onClick`: triggered when an object is clicked, usually a link or a button.
- `onMouseOver` / `onMouseOut`: triggered when moved over an element or off the element.
- See [http://www.w3schools.com/js/js\\_events.asp](http://www.w3schools.com/js/js_events.asp)

# Statements

- Statements in JavaScript are ended by a semicolon or a new line.

```
<script type="text/javascript">
```

```
<!--
```

```
var number = 7;
```

```
if(number == 7){
```

```
    document.write("Lucky 7!");
```

```
}
```

```
//-->
```

```
</script>
```

# If statement

```
<script type="text/javascript">
```

```
<!--
```

```
var number = 10;
```

```
if(number == 7){
```

```
    document.write("Lucky 7!");
```

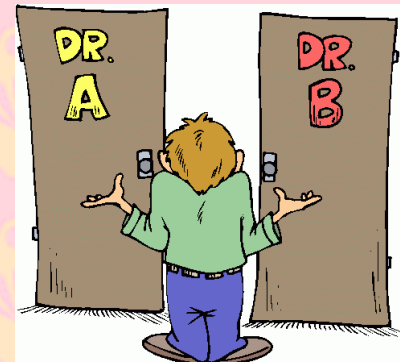
```
}else{
```

```
    document.write("You're not very lucky today...");
```

```
}
```

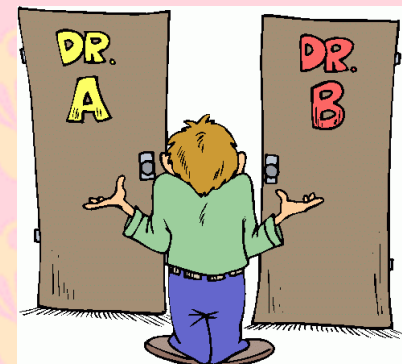
```
//-->
```

```
</script>
```



# Nested if statements

```
<script type="text/javascript">
<!--
var visitor = "principal";
if(visitor == "teacher"){
    document.write("My dog ate my homework...");
}else if(visitor == "principal"){
    document.write("It was the other guy's fault!");
} else {
    document.write("How do you do?");
}
//-->
</script>
```



# While loops

```
<script type="text/javascript">  
var myCounter = 0;  
var linebreak = "<br />";  
document.write(linebreak);  
while(myCounter < 10){  
    document.write("myCounter = " + myCounter);  
    document.write(linebreak);  
    myCounter++;  
}  
</script>
```



# For loops

```
<script type="text/javascript">  
var linebreak = "<br />";  
document.write(linebreak);  
for(i = 0; i < 5; i++){  
    document.write("Counter i = " + i);  
    document.write(linebreak);  
}  
</script>
```



# Arrays

- You have seen arrays in other languages, and they aren't that different in JavaScript. You have to use a special function to create a new array.

```
<script type="text/javascript">
```

```
var myArray = new Array();
```

```
myArray[0] = "Orange";
```

```
myArray[1] = "Apple";
```

```
myArray[2] = "Pumpkin";
```

```
document.write(myArray[0] + myArray[1] +  
myArray[2]);
```

```
</script>
```

OrangeApplePumpkin

# Alerts

- Alerts should be very, very rarely used and even then these following guidelines should be considered when using them. JavaScript alerts are ideal for the following situations:
- If you want to be absolutely sure they see a message before doing anything on the website.
- You would like to warn the user about something.
- An error has occurred and you want to inform the user of the problem.
- When asking users for confirmation of some action.

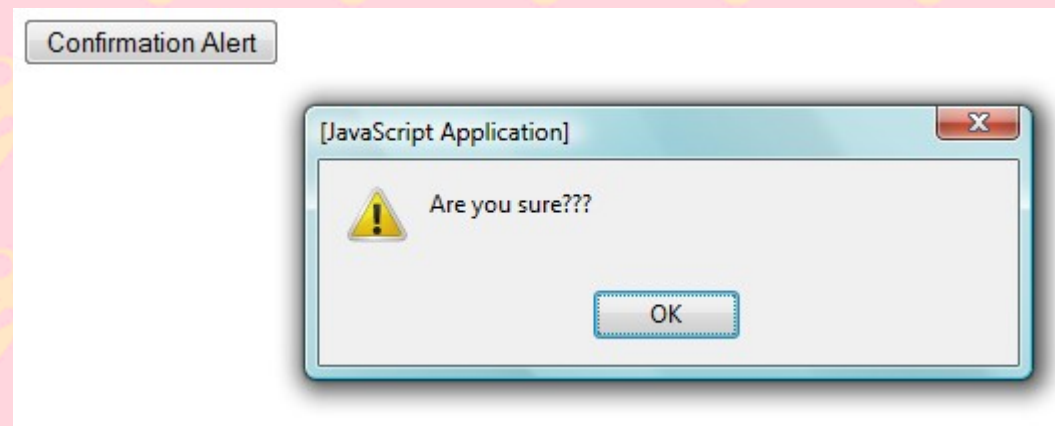
# Alerts

- Here is an alert example to be sure your visitors are fully aware before submitting a form.

```
<form>
```

```
<input type="button" onClick="alert('Are you  
sure???');" value="Confirmation Alert"></input>
```

```
</form>
```

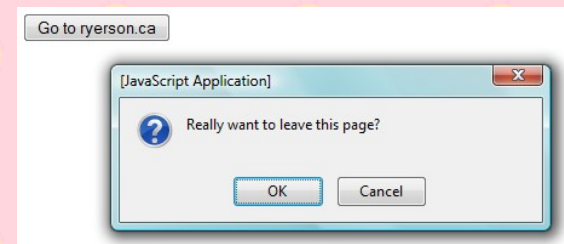


# Confirm

- Because *alert* does not give you the choice of accepting or refusing (besides exiting the browser), there can be a better choice: *confirm*.
- A small dialogue box pops up and appears in front of the web page currently in focus. The confirm box is different from the alert box. It supplies the user with a choice; they can either press OK to confirm the popup's message or they can press cancel and not agree to the popup's request.

# Confirm

```
<head><script type="text/javascript">  
function confirmation() {  
    var answer = confirm("Really want to leave this page?")  
  
    if (answer){  
        alert("Bye bye!")  
  
        window.location = "http://www.ryerson.ca/";  
    }  
    else{  
        alert("Thanks for sticking around!")  
    }  
}  
//-->  
</script></head><body><form>  
<input type="button" onclick="confirmation()" value="Go to  
    ryerson.ca">  
</form></body>
```



# Timed Redirection

- We have seen redirection early on with the `window.location` method. Here is another example but this time using a timing delay.

```
<head><script type="text/javascript">  
function delayer(){  
    window.location = "http://www.toronto.ca"}  
</script></head>  
<body onLoad="setTimeout('delayer()', 5000)">  
<h2>Prepare to be redirected!</h2>  
<p>The city of Toronto website will appear in a few  
seconds</p>  
</body></html>
```



milliseconds

**Prepare to be redirected!**

The city of Toronto website will appear in a few seconds

# Popup windows

- One of the most flagrant misuses of JavaScript are popup windows.
- The `window.open()` function creates a new browser window, customized to your specifications, without the use of an HTML anchor tag.
- There are three arguments that the `window.open` function takes: The relative or absolute URL of the web page to be opened, the text name for the window and a long string that contains all the different properties of the window (key=value separated by commas).

```
window.open( "http://www.cbc.ca/", "pop", "status = 1,  
height = 300, width = 300, resizable = 0")
```

# Window properties

- dependent: subwindow closes if the parent window (the window that opened it) closes.
- fullscreen: display browser in fullscreen mode.
- height: the height of the new window, in pixels.
- width: the width of the new window, in pixels.
- left: pixel offset from the left side of the screen.
- top: pixel offset from the top of the screen.
- resizable: allow the user to resize the window or prevent the user from resizing. (currently broken in Firefox)
- status: Display or don't display the status bar.

# Popup Example

```
<head>
```

```
<script type="text/javascript">
```

```
function pop1() {
```

```
  window.open( "http://www.cs.ryerson.ca/", "ryerson",  
    "status=0, height=400, width=400, left=500, top=10,  
    resizable=0, dependent=1" ) }
```

```
</script>
```

```
</head>
```

```
<body onLoad = "pop1();">
```

```
Main window
```

```
</body>
```



# End of lesson