# *ADVANCED HTML (2)*

## *Dynamic web pages*
## *Javascript*

*Alex Sánchez*

# Contents

- Javascript

# The *Javascript* language

# What is javascript

- JavaScript is a cross-platform, object-oriented scripting language invented in web browsers to make web pages more dynamic and give feedback to your user.
- Adding JavaScript to your HTML code allows you to change completely the document appearance,
- JavaScript is mainly used as a ***client side*** scripting language.

# What can JavaScript do?

- Display contents based on system information such as day or time.
- Detect the visitor's browser
- Control Browsers
- Validate forms data
- Create Cookies
- Add interactivity to your website
- Change page contents dynamically

# How Does It Work?

- Embedded within HTML page
  - View source
- Executes on client
  - Fast, no connection needed once loaded
- Simple programming statements combined with HTML tags
- Interpreted (not compiled)
  - No special tools required

# Linking to a JavaScript file: `script`



```
<script src="filename"
type="text/javascript"></script>
HTML
```

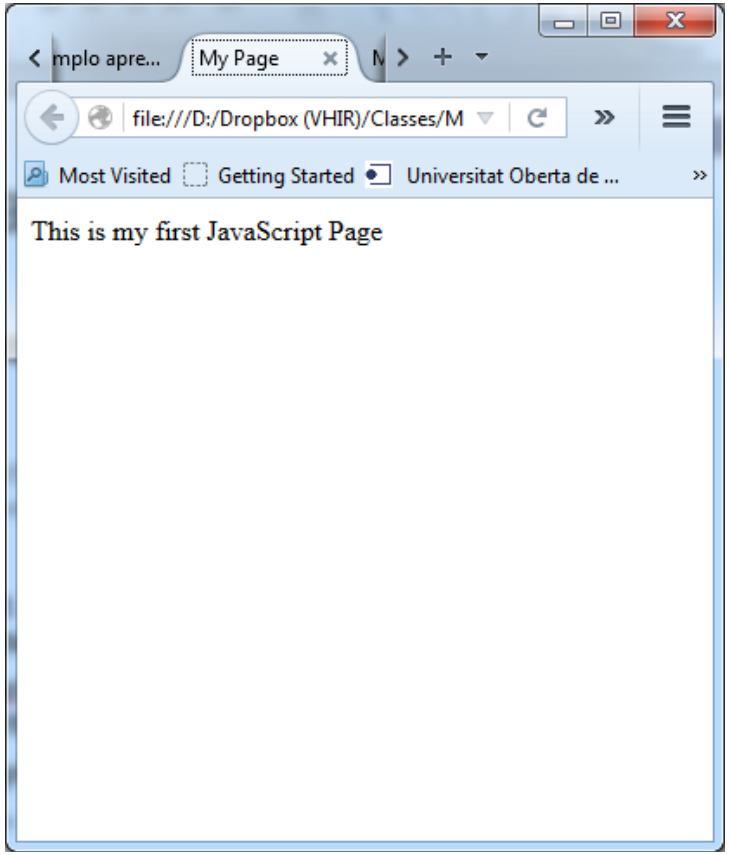
- script tag should be placed in HTML page's head
- script code is stored in a separate .js file
- JS code can be placed directly in the HTML file's body or head (like CSS)
  - but this is bad style (should separate content, presentation, and behavior

# JavaScript Statements

## Source code

```
<html>
<head><title>My Page</title></head>
<body>
<script language="JavaScript">
document.write('This is my first JavaScript Page');
</script>
</body>
</html>
```

## Output




JavascriptExample1.html

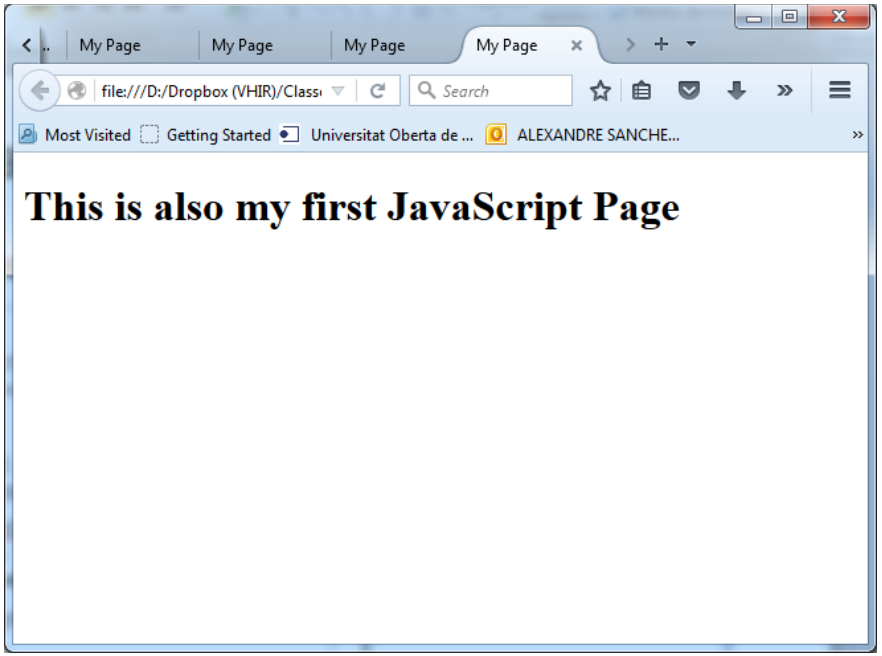
# JavaScript Statements

## Source code

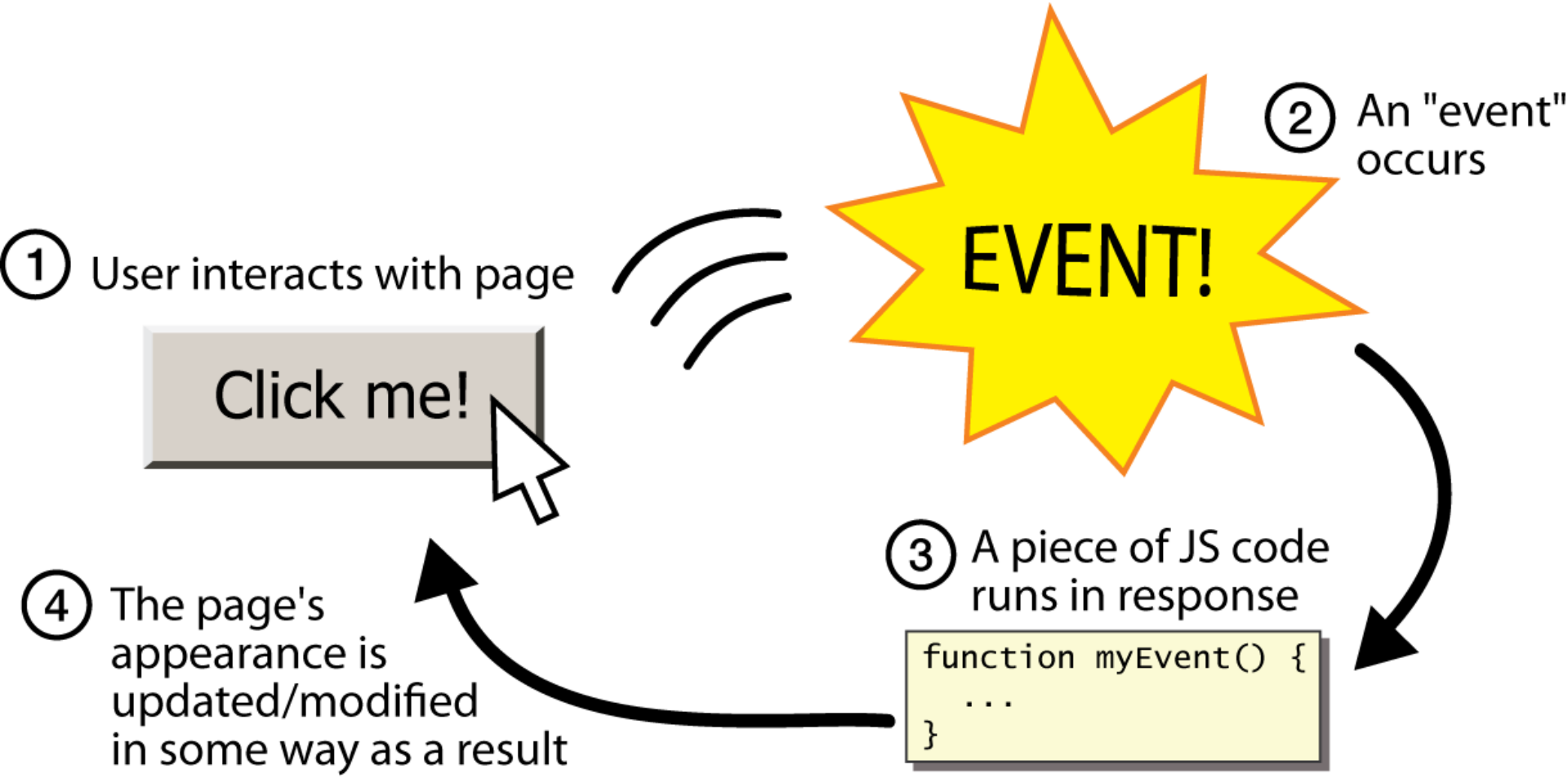
```
<html>
<head><title>My Page</title></head>
<body>
<script language="JavaScript">
document.write('<h1>This is also my first JavaScript Page</h1>');
</script>
</body>
</html>
```

JavascriptExample1b.html

## Output



# Event-driven programming



# JavaScript Statements

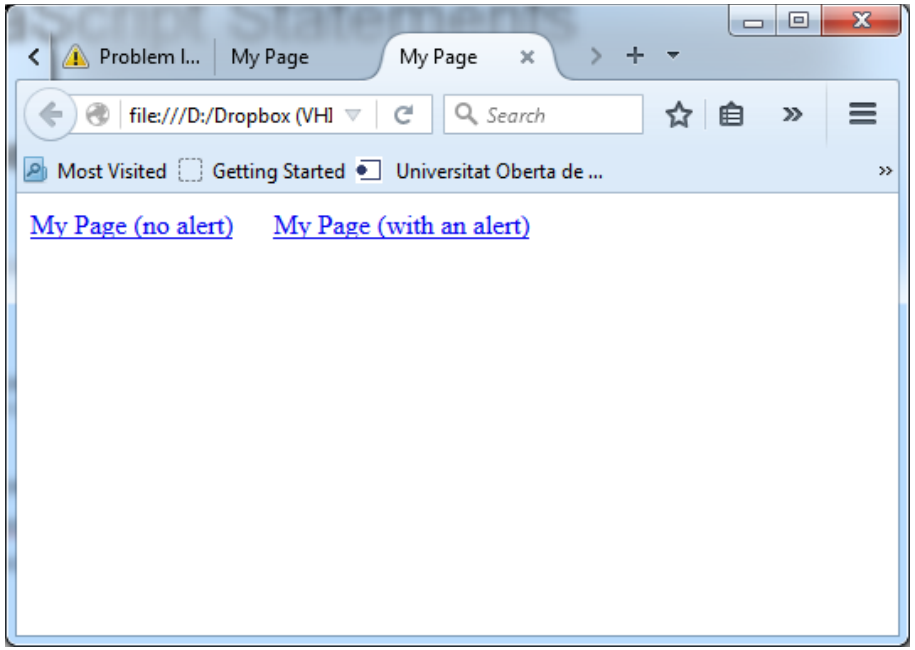
## Source code

```
<html>
<head><title>My Page</title></head>
<body>
<p>
<a href="javascriptEx2.html">My Page (no alert)</a>
 
<a href="javascriptEx2.html"
onMouseover="window.alert('Hello');">
My Page (with an alert)</A>
</p>
</body>
</html>
```

An Event

JavaScript written inside HTML

## Output




JavascriptExample2.html

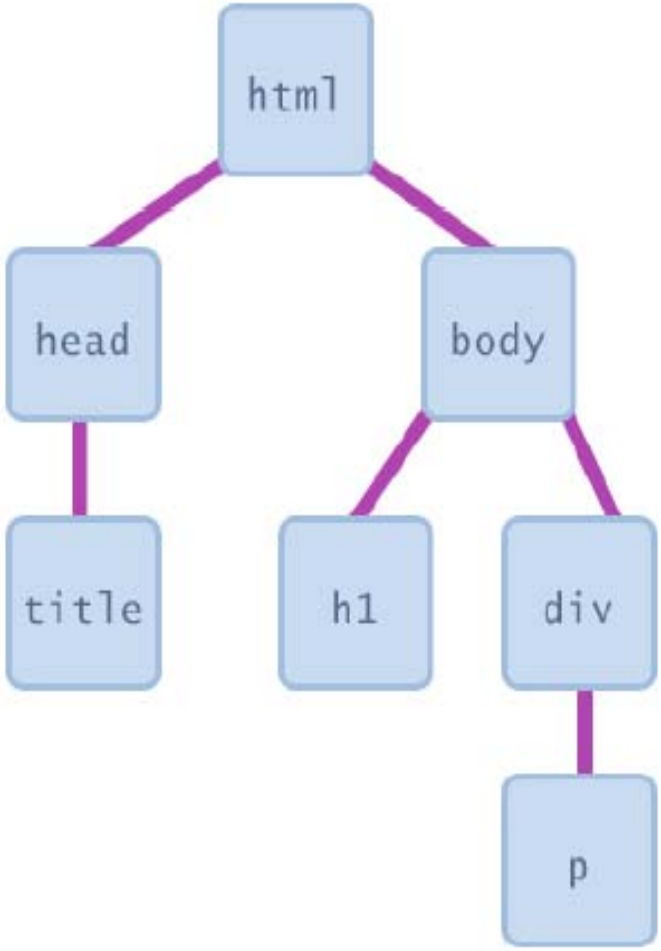
# Event-driven programming



- JavaScript programs do not start with a main method (or implicit main like in PHP)
- JavaScript programs instead wait for user actions called *events* and respond to them
- event-driven programming: writing programs driven by user events

# Document Object Model (DOM)

- The DOM breaks down your HTM document into the following structure
  - The entire document is a "document" object.
  - Every HTML tag can be referenced as an element node in the "document" object
  - Any text contained in the HTML element nodes can be referenced as "text" nodes
  - Every HTML attribute is referenced as an "attribute" node
  - Even comments are referenced as "comment" nodes



# DOM element objects

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```

**DOM Element Object**

| Property | Value |
|---|---|
| tagName | "IMG" |
| src | "octopus.jpg" |
| alt | "an octopus" |
| id | "icon01" |

JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

# Accessing elements: `document.getElementById`



- document.getElementById returns the DOM object for an element with a given id
- can change the text inside most elements by setting the innerHTML property
- can change the text in form controls by setting the value property

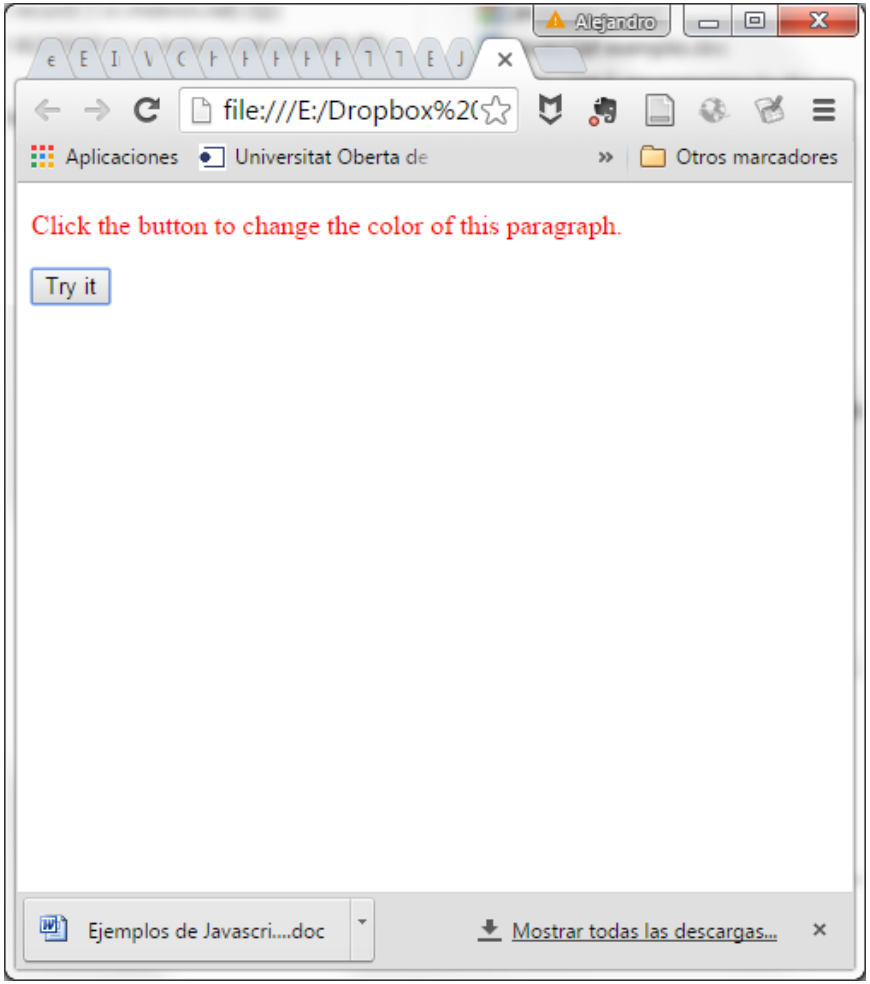
# Changing element style: `element.style`



| Attribute | Property or style object |
|---|---|
| color | color |
| padding | padding |
| background-color | backgroundColor |
| border-top-width | borderTopWidth |
| Font size | fontSize |
| Font famiy | fontFamily |

# Accessing elements: `document.getElementById`

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to change
    the color of this paragraph.</p>
<button onclick="myFunction()">Try
    it</button>
<script>
function myFunction() {
    var x =
    document.getElementById("demo");
    x.style.color = "red";
}
</script>
</body>
</html>
```




JavascriptExample3.html

# HTML FORMS AND JAVASCRIPT

# HTML Forms and JavaScript

- JavaScript is very good at processing user input in the web browser
- HTML `<form>` elements receive input
- Forms and form elements have unique names
  - Each unique element can be identified
  - Uses JavaScript Document Object Model (DOM)

# Example Statements

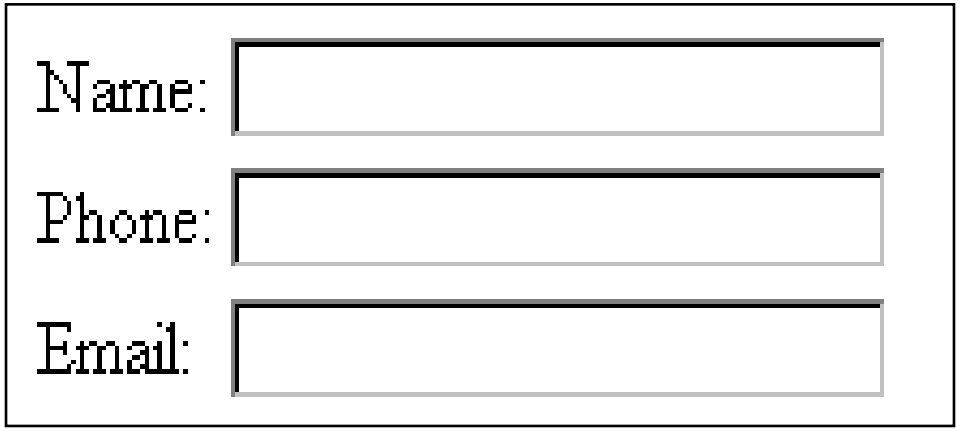
```
<script language="JavaScript">
window.prompt('Enter your name:','');
</script>

<form>
<input type="button" Value="Press"
  onClick="window.alert('Hello');">
</form>
```

Another event

Note quotes: " and '

# Naming Form Elements in HTML



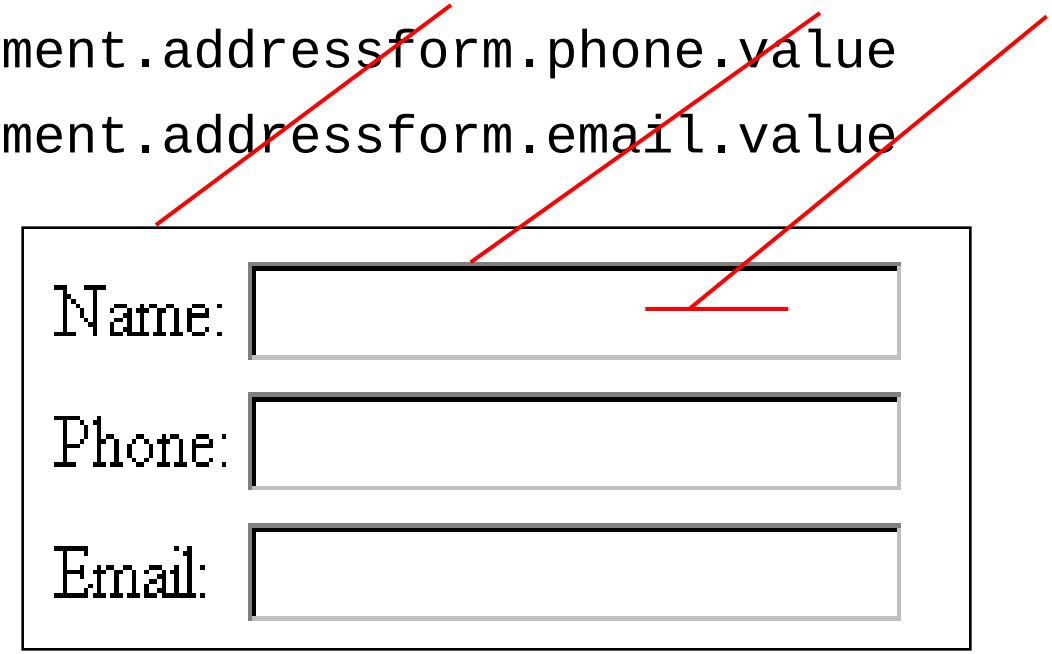

```
<form name="addressform">
Name:  <input name="yourname"><br />
Phone: <input name="phone"><br />
Email: <input name="email"><br />
</form>
```

# Forms and JavaScript

*document.**formname.elementname**.value*

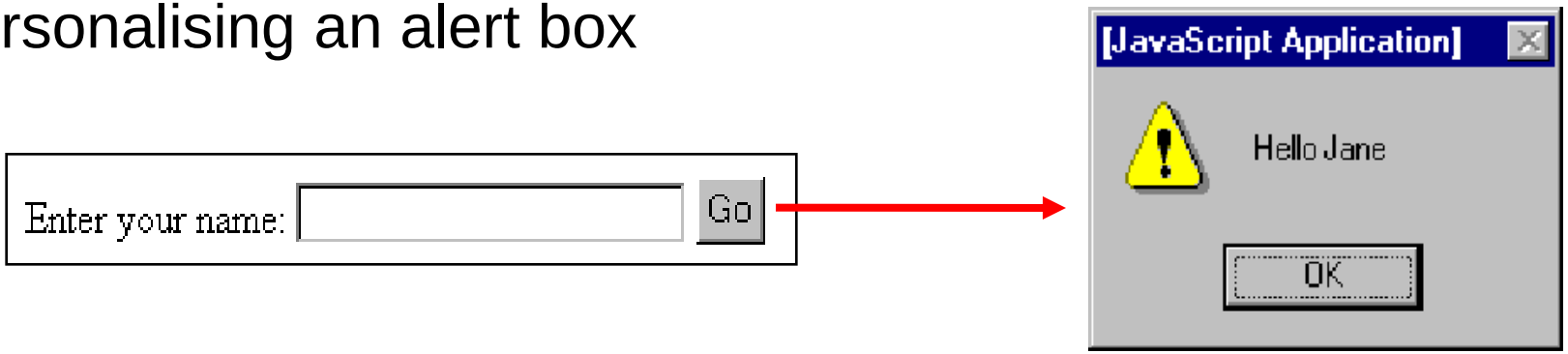
Thus:

```
document.addressform.yourname.value
document.addressform.phone.value
document.addressform.email.value
```



# Using Form Data

Personalising an alert box




```
<form name="alertform">
Enter your name:
<input type="text" name="yourname">
<input type="button" value= "Go"
  onClick="window.alert('Hello ' + →
  document.alertform.yourname.value);">
</form>
```

# Exercise

- Create a simple dynamic web page that performs some type of calculation.
- The page should contain
  - A brief description of what it is intended for (maybe including links to related topics)
  - A form to allow the user provide input
    - The calculation will be executed when the user presses the “submit” button
    - A new page presenting the result will be open
  - Eventually you may provide input validation
- Examples:
  - http://bioinformatics.mdanderson.org/MicroarraySampleSize/

# MORE JAVASCRIPT SYNTAX

# Variables



```
var name = expression;                    JS
```

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;                       JS
```

variables are declared with the var keyword (case sensitive)

types are not specified, but JS does have types ("loosely typed")

- `Number, Boolean, String, Array, Object, Function, Null, Undefined`
- can find out a variable's type by calling `typeof`

# Number type



```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);
                        JS
```

integers and real numbers are the same type (no int vs. double)

same operators: + - * / % ++ -- = += -= *= /= %=

similar precedence to Java

many operators auto-convert types: "2" * 3 is 6

# Comments (same as Java)



```
// single-line comment
/* multi-line comment */
                    JS
```

identical to Java's comment syntax

recall: 4 comment syntaxes

- HTML: <!-- comment -->
- CSS/JS/PHP: /* comment */
- Java/JS/PHP: // comment
- PHP: # comment

# Math object



```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```
*JS*

- methods: `abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan`
- properties: `E, PI`

# Special values: null and undefined



```
var ned = null;
var benson = 9;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined
                    JS
```

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value
- Why does JavaScript have both of these?

# Logical operators



- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !== are strict equality tests; checks both type and value
  - "5.0" === 5 is false

# if/else statement (same as Java)



```
if (condition) {
   statements;
} else if (condition) {
   statements;
} else {
   statements;
}
```

JS

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

# Boolean type



```
var iLike190M = true;
var ieIsGood = "IE6" > 0; // false
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
                    JS
```

- any value can be used as a Boolean
  - "falsey" values: 0, 0.0, NaN, "", null, and undefined
  - "truthy" values: anything else
- converting a value into a Boolean explicitly:
  - `var boolValue = Boolean(otherValue);`
  - `var boolValue = !!(otherValue);`

# for loop (same as Java)



```
var sum = 0;
for (var i = 0; i < 100; i++) {
   sum = sum + i;
}
```
*JS*

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s.length; i++) {
   s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheellllloo"
```
*JS*

# while loops (same as Java)



```
while (condition) {
   statements;
}
```
JS

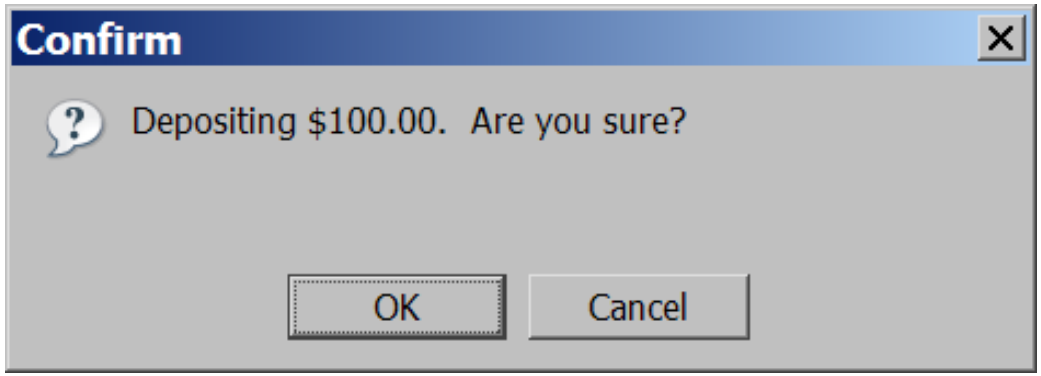
```
do {
   statements;
} while (condition);
```
JS

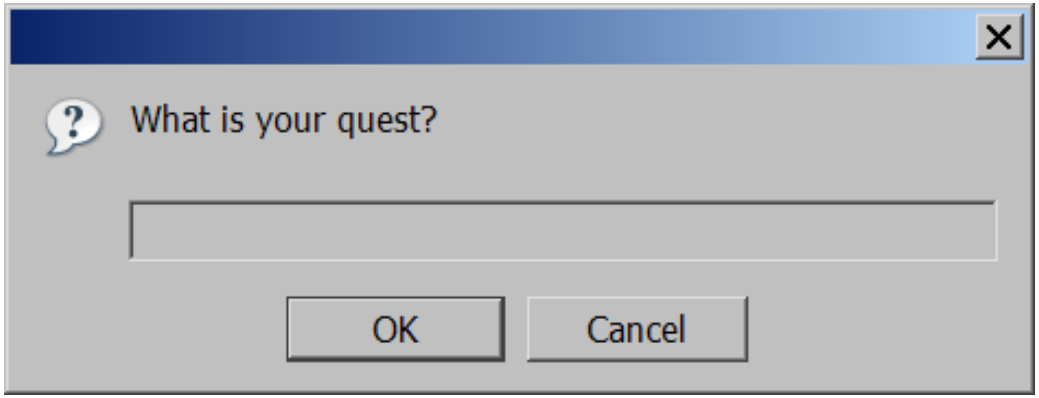
- break and continue keywords also behave as in Java

# Popup boxes



```
alert("message"); // message
confirm("message"); // returns true or false
prompt("message"); // returns user input string
                                        JS
```



# Arrays



```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```

*JS*

```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[4] = "Curly"; // stooges.length is 5
stooges[4] = "Shemp"; // stooges.length is 5
```

*JS*

# Array methods



```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

- array serves as many JS data structures: list, queue, stack, ...
- methods: `concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift`
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

# `String` type



```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
                    JS
```

methods: `charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase`

- charAt returns a one-letter String (there is no char type)

length property (not a method as in Java)

Strings can be specified with "" or ''

concatenation with + :

- 1 + 1 is 2, but "1" + 1 is "11"

# More about `String`



- escape sequences behave as in Java: \' \" \& \n \t \\
- converting between numbers and Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " bananas, ah ah ah!"; // "10
bananas, ah ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN
```
JS

accessing the letters of a String:

```
var firstLetter = s[0]; // fails in IE
var firstLetter = s.charAt(0); // does work in IE
var lastLetter = s.charAt(s.length - 1);
```
JS

# Splitting strings: split and join



```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown",
"fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
                    JS
```

- split breaks apart a string into an array using a delimiter
  - can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them