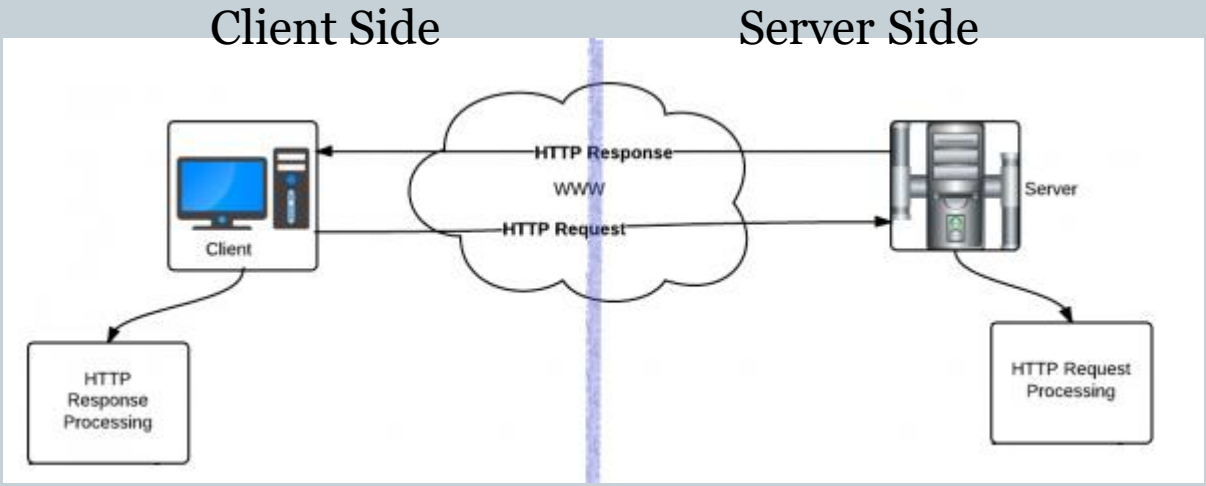
# Client Side Scripting
# Επισκόπηση DOM, Javascript

ΑΝΤΏΝΗΣ ΣΙΔΗΡΌΠΟΥΛΟΣ

# Client Side vs. Server Side





# Client Side Programming



- Το «πρόγραμμα» εκτελείται μέσα στον browser του χρήστη και χρησιμοποιεί τους πόρους του υπολογιστή του χρήστη.
- Διάφορες κατηγορίες από client Side προγράμματα:
  - Scripts (Javascript, VBScript, PerlScript, …)
  - Precompiled (Java Applets, ActiveX Controls, Flash - ActionScript, …)
- Τα εκτελεί:
  - ο browser ο ίδιος (Javascript)
  - ή κάποια virtual machine που είναι εγκατεστημένη ως add-on στον browser (Java VM, Flash Player)
  - Ή κάποιο add-on ή extension του browser

# Υπόσταση



- Είτε αποτελούν αυτόνομα αντικείμενα μέσα σε μια ιστοσελίδα (πχ Java applet)
  - Έχουν τον δικό τους χώρο/εμφάνιση στην ιστοσελίδα ως ανεξάρτητα αντικείμενα.
  - Συνήθως δεν μπορούν να αλληλεπιδράσουν με τα υπόλοιπα στοιχεία της ιστοσελίδας.
- Είτε αλληλεπιδρούν με τα στοιχεία της ιστοσελίδας (πχ JavaScript)
  - ΔΕΝ έχουν τον δικό τους χώρο/εμφάνιση στην ιστοσελίδα ως ανεξάρτητα αντικείμενα.
  - Αντιλαμβανόμαστε ότι υπάρχουν από την επίδρασή τους σε στοιχεία (αντικείμενα) της ιστοσελίδας.

# Γλώσσα σεναρίου Javascript



- Γλώσσα σεναρίου (script language) για διαδραστικές (interactive) ιστοσελίδες
- αναπτύχθηκε από τη Netscape, το 1995.
- Τρέχει σε όλους τους φυλλομετρητές, ανεξάρτητα λειτουργικών συστημάτων.
- Ο κώδικας εισάγεται στο HTML έγγραφο (με διάφορους τρόπους)
- Τα scripts διερμηνεύονται από τον φυλλομετρητή (client) και δεν μεταγλωττίζονται.

# Γλώσσα σεναρίου Javascript



- Προσθέτει διαδραστικότητα και δυναμικά στοιχεία στην ιστοσελίδα, ελέγχει τις φόρμες και τις γραφικές διεπαφές χρήστη (GUI), κ.ά.
- Κάνει χρήση αντικειμένων (π.χ. παράθυρα, μαθηματικές συναρτήσεις, πίνακες, κ.ά).
- Χειρίζεται συμβάντα (event handling) και οπτικά εφέ του interface.
- Ο κώδικας της JavaScript μπορεί να ανιχνεύσει ενέργειες χρηστών (πχ: onclick, onmouseover etc.)

# ECMAScript

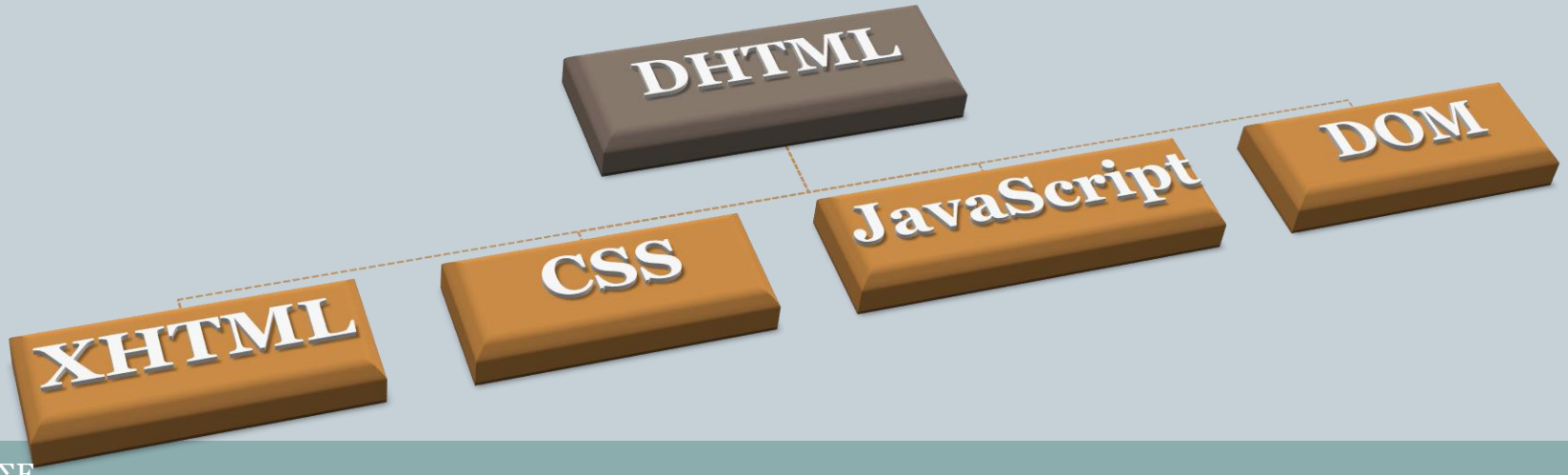


- The responsibility for the development of a scripting standard has been transferred to an international body called the European Computer Manufacturers Association (ECMA).
- The standard developed by the ECMA is called ECMAScript, though browsers still refer to it as JavaScript.
- The latest version is ECMA-262, which is supported by the major browsers.

# What is DHTML?



- Dynamic HTML (DHTML)
  - Makes possible a Web page to react and change in response to the user's actions
- DHTML = HTML + CSS + JavaScript



# DTHML = HTML + CSS + JavaScript



- **HTML** defines Web sites content through semantic tags (headings, paragraphs, lists, …)
- **CSS** defines 'rules' or 'styles' for presenting every aspect of an HTML document
  - Font (family, size, color, weight, etc.)
  - Background (color, image, position, repeat)
  - Position and layout (of any object on the page)
- **JavaScript** defines dynamic behavior
  - Programming logic for interaction with the user, to handle events, etc.

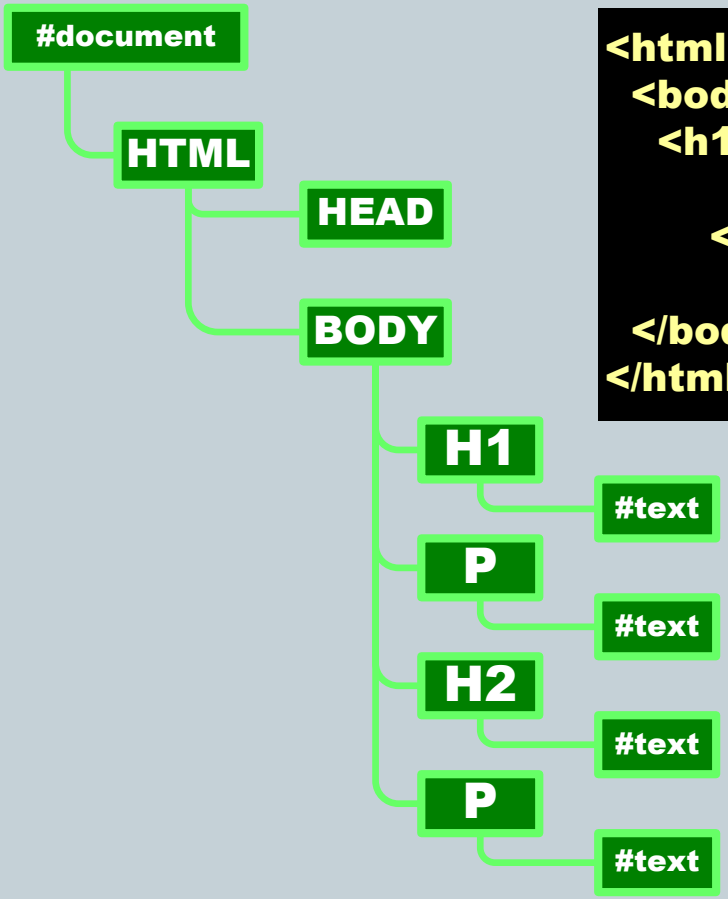
# Javascript και DOM



- Η ασυμβατότητα της JavaScript με κάποιους browsers έφερε την δημιουργία του πρωτοκόλλου **Document Object Model (DOM)**.
- πρωτόκολλο της W3C
- είναι το API της αλληλεπίδρασης της JavaScript με το περιεχόμενο της ιστοσελίδας.
- χρησιμοποιείται για την ανάγνωση και επεξεργασία των εγγράφων HTML και XML.
- οι προγραμματιστές μπορούν να δημιουργήσουν έγγραφα, να αλλάξουν την δομή τους, και να προσθέσουν, τροποποιήσουν ή διαγράψουν στοιχεία και περιεχόμενο.

# Document Tree Structure

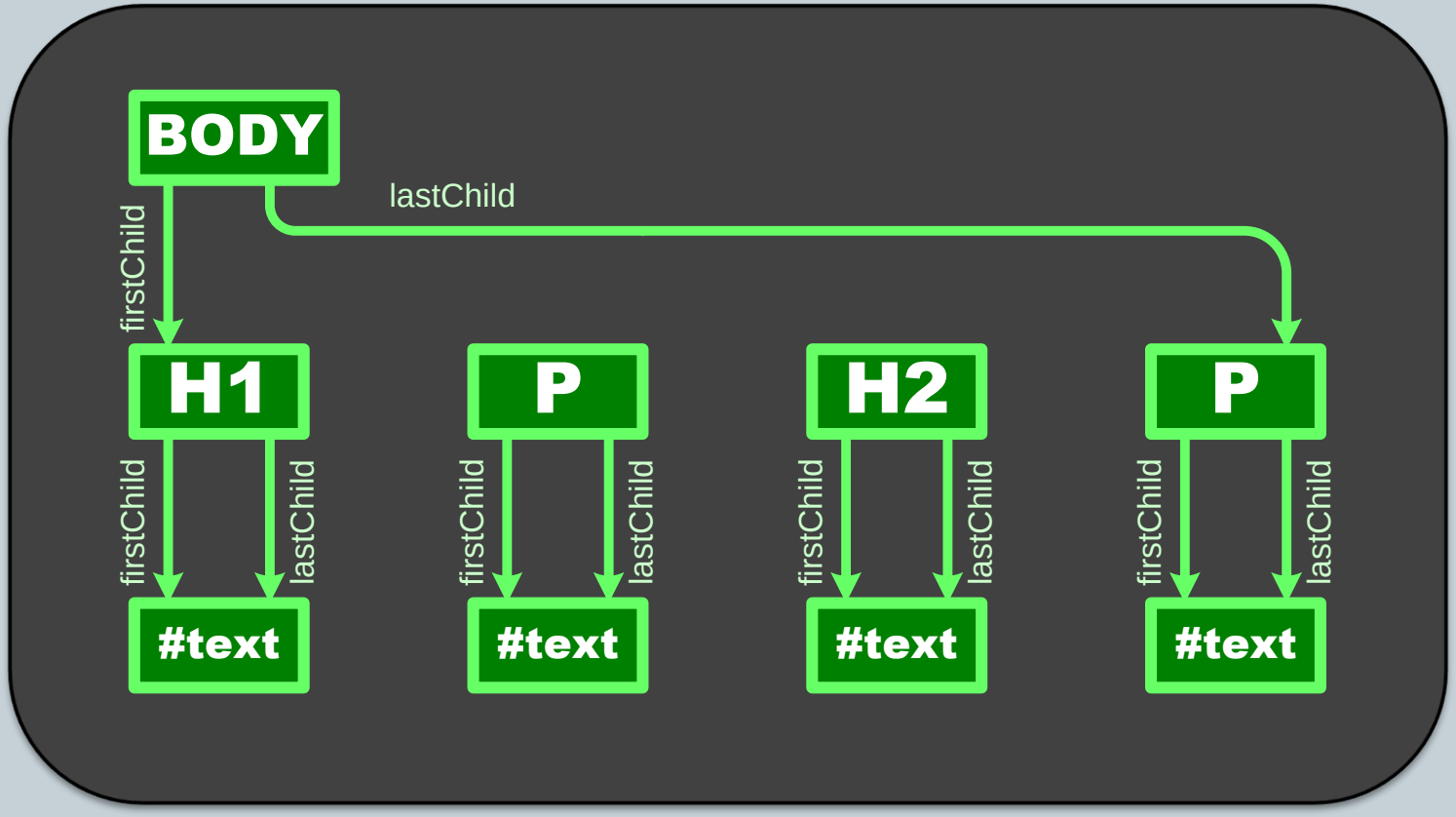





```
<html>
  <body>
    <h1>Heading 1</h1>
          <p>Paragraph.</p>
       <h2>Heading 2</h2>
          <p>Paragraph.</p>
  </body>
</html>
```

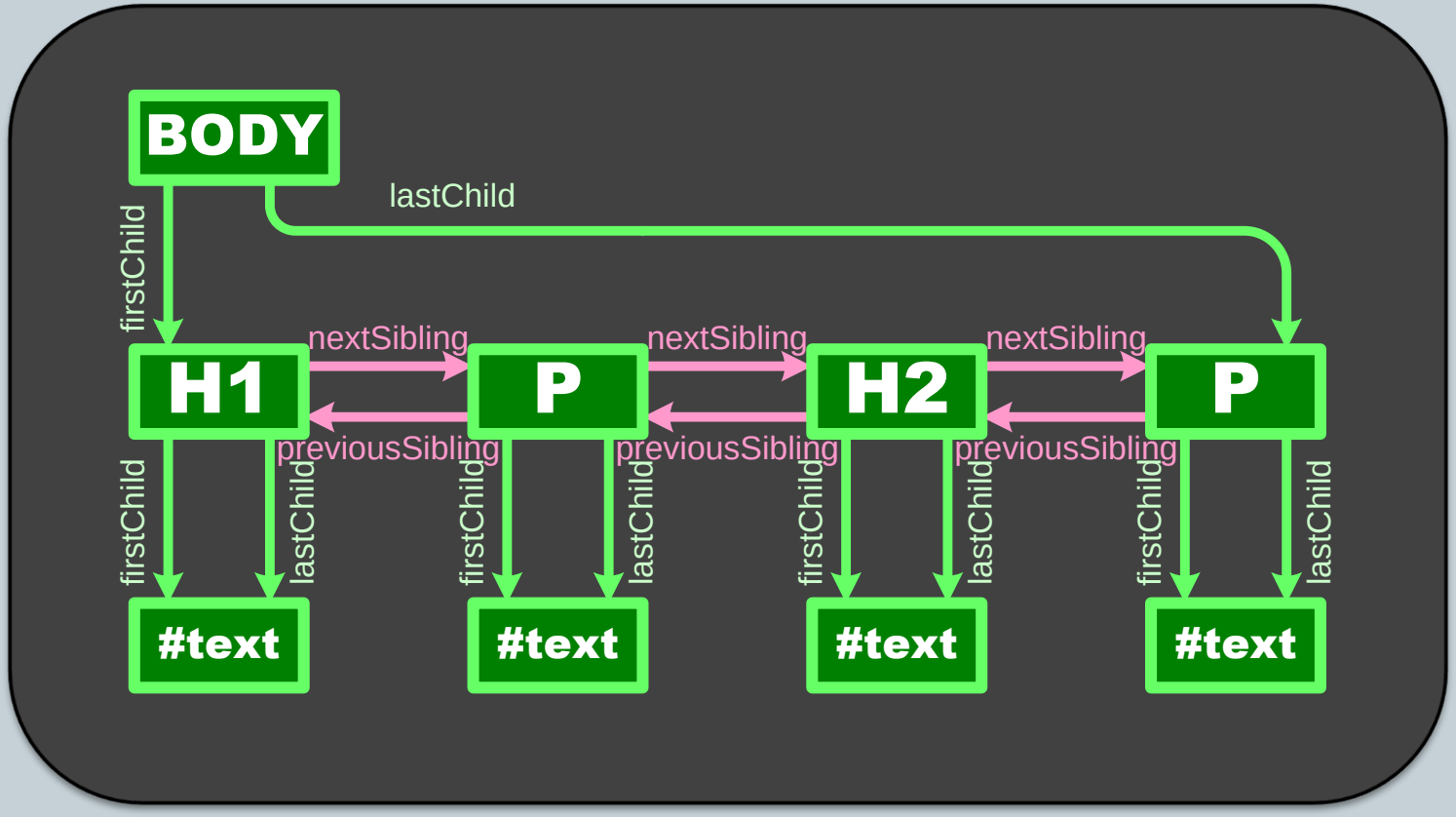
# child, sibling, parent





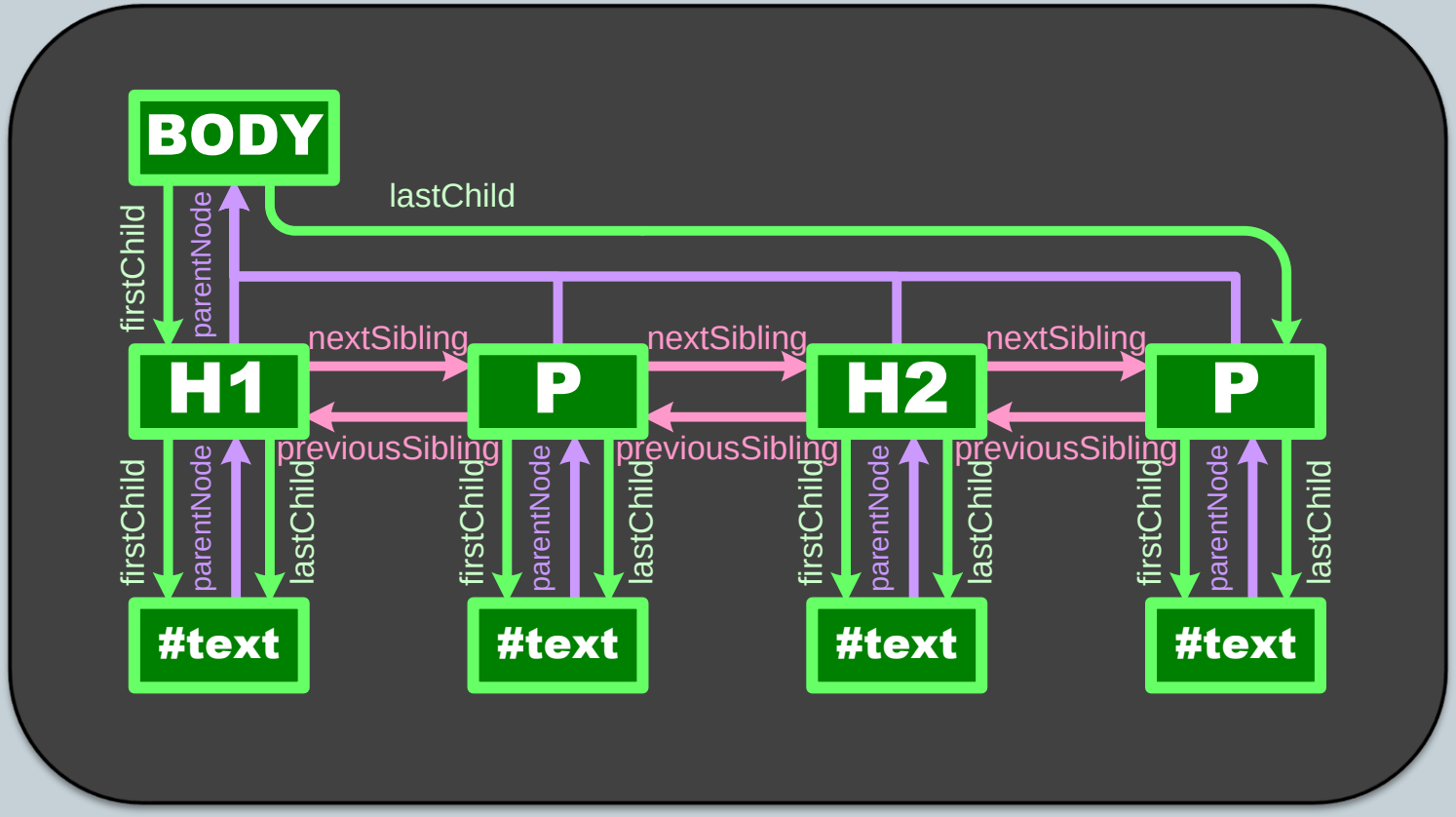
# child, sibling, parent





# child, sibling, parent





# Walk the DOM



- Using recursion, follow the `firstChild` node, and then the `nextSibling` nodes.

```
function walkTheDOM(node) {
    // do something on node;
    node = node.firstChild;
    while (node) {
        walkTheDOM(node);
        node = node.nextSibling;
    }
}
```


# JavaScript

## DYNAMIC BEHAVIOR IN A WEB PAGE

# JavaScript



- 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
- Powerful to manipulate the DOM

# JavaScript Advantages



- JavaScript allows interactivity such as:
  - Implementing form validation
  - React to user actions, e.g. handle keys
  - Changing an image on moving mouse over it
  - Sections of a page appearing and disappearing
  - Content loading and changing dynamically
  - Performing complex calculations
  - Custom HTML controls, e.g. scrollable table
  - Implementing AJAX functionality

# What Can JavaScript Do?



- Can handle events
- Can read and write HTML elements and modify the DOM tree
- Can validate form data
- Can access / modify browser cookies
- Can detect the user's browser and OS
- Can be used as object-oriented language
- Can handle exceptions
- Can perform asynchronous server calls (AJAX)

# The First Script

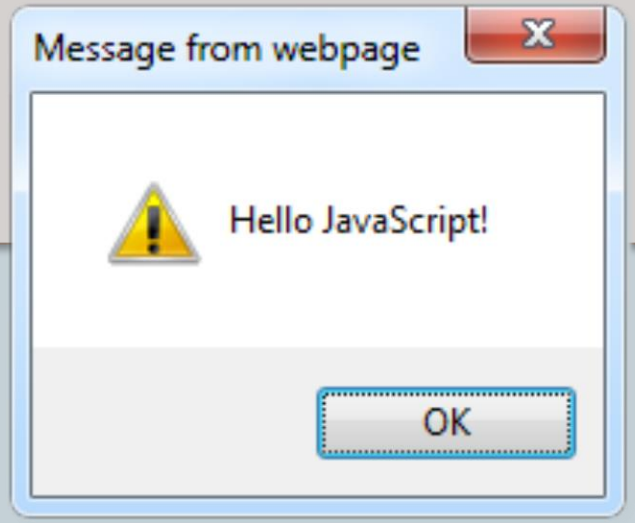


first-script.html

```
<html>

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

</html>
```



# Another Small Example

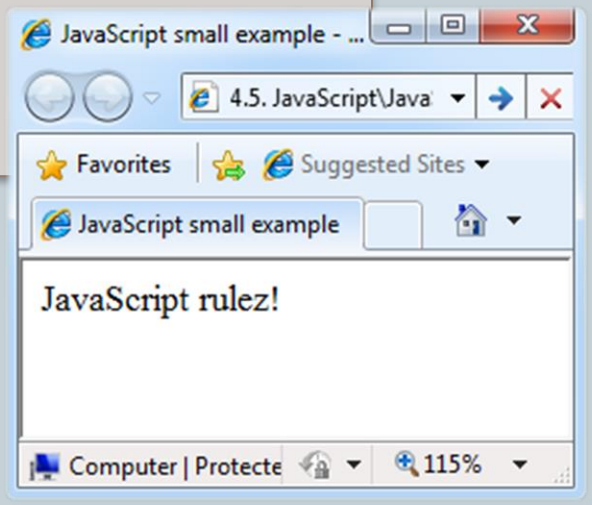
small-example.html



```
<html>

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

</html>
```



# Using JavaScript Code



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

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

    - Highly recommended
    - The `.js` files get cached by the browser
    - Often placed before the </body> tag

Γιατί?????

Δεν είναι η καλύτερη μέθοδος

# JavaScript – When is Executed?



- JavaScript code is executed during the page loading or when the browser fires an event
  - Two blocks of code can't run simultaneously (no threads)
  - 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!')" />
```

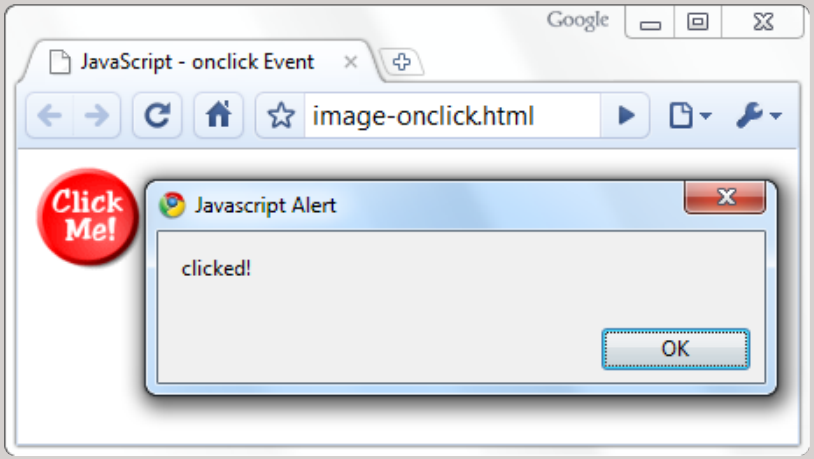
# Calling a JavaScript Function from Event Handler – Example



image-onclick.html

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

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

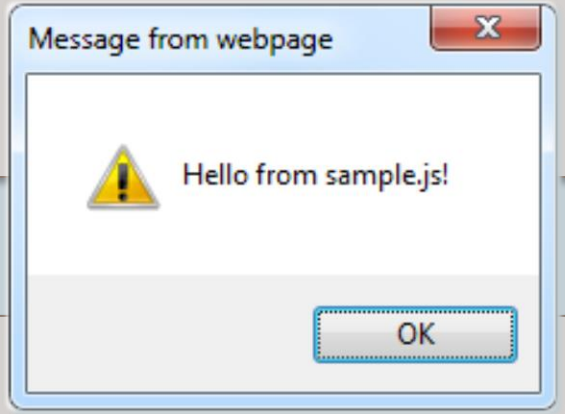


# Using External Script Files



- Using external script files:

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



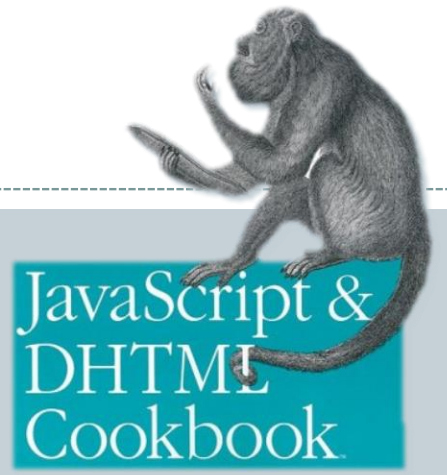

- External JavaScript file:

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

sample.js

# The JavaScript Syntax

```
if (pop < 10)
{
     map.graphics.add(features[i].setSymbol(onePopSymbol));
}
else if (pop >= 10 && pop < 95)
{
     map.graphics.add(features[i].setSymbol(twoPopSymbol));
}
else if (pop >= 95 && pop < 365)
{
     map.graphics.add(features[i].setSymbol(threePopSymbol));
}
else if (pop >= 365 && pop < 1100)
{
     map.graphics.add(features[i].setSymbol(fourPopSymbol));
}
else
{
     map.graphics.add(features[i].setSymbol(fivePopSymbol));
}
```




JAVA SCRIPT

# JavaScript Syntax



- The JavaScript syntax is similar to C# and Java
  - Operators **(+, *, =, !=, &&, ++,** …)
  - Variables (**typeless**)
  - Conditional statements (**`if`, `else`)**
  - Loops (**`for`, `while`**)
  - Arrays (**`my_array[]`**) and associative arrays (**`my_array['abc']`)**
  - Functions (**can return value**)
  - Function variables

# Data Types



- JavaScript data types:
  - Numbers (integer, floating-point)
  - Boolean (true / false)
- String type – string of characters

```
var myName = "You can use both single or double quotes for strings";
```

- Arrays

```
var my_array = [1, 5.3, "aaa"];
var my_arr2 = new Array();
my_arr2[2]="aaa"
```

- Associative arrays (hash tables)

```
var my_hash = {a:2, b:3, c:"text"};
var my_arr2 = new Array();
my_arr2['c']="text"
```

# Everything is Object



- Every variable can be considered as object
  - For example strings and arrays have member functions:

**objects.html**

```
var test = "some string";
alert(test[7]); // shows letter 'r'
alert(test.charAt(5)); // shows letter 's'
alert("test".charAt(1)); //shows letter 'e'
alert("test".substring(1,3)); //shows 'es'
```

```
var arr = [1,3,4];
alert (arr.length); // shows 3
arr.push(7); // appends 7 to end of array
alert (arr[3]); // shows 7
```

# String Operations



- The + operator joins strings

```
string1 = "fat ";
string2 = "cats";
alert(string1 + string2);  // fat cats
```

- What is "9" + 9?

```
alert("9" + 9);  // 99
```

- Converting string to number:

```
alert(parseInt("9") + 9);  // 18
```

# Arrays Operations and Properties



- Declaring new empty array:

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

- Declaring an array holding few elements:

```
var arr = [1, 2, 3, 4, 5];
```

- Appending an element / getting the last element:

```
arr.push(3);
var element = arr.pop();
```

- Reading the number of elements (array length):

```
arr.length;
```

# 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);
```

# Sum of Numbers – Example

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>
```

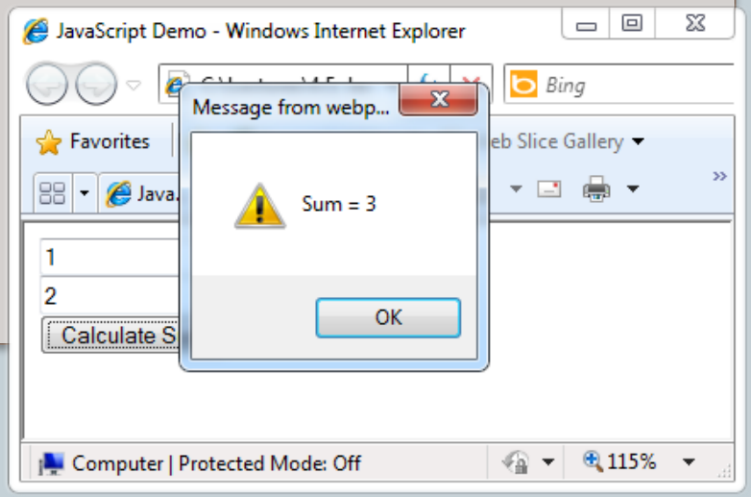
# Sum of Numbers – Example (2)



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="calcSum()" />
    <input type="text" name="textBoxSum"
      readonly="readonly"/>
  </form>
</body>

</html>
```

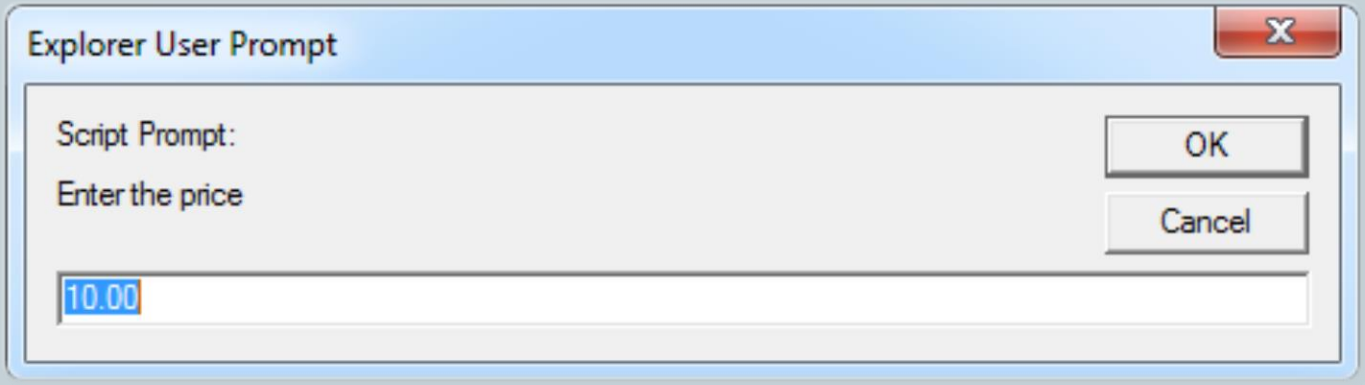


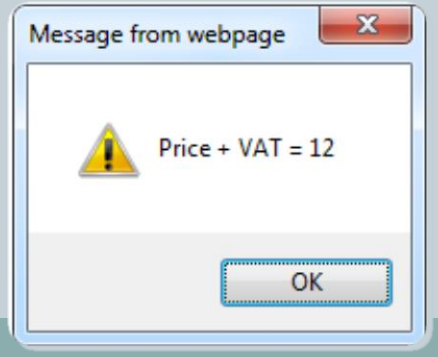
# JavaScript Prompt – Example



## prompt.html

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



# Conditional Statement (`if`)



```
unitPrice = 1.30;
if (quantity > 100) {
  unitPrice = 1.20;
}
```

| Symbol | Meaning |
|---|---|
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| == | Equal |
| != | Not equal |

# Conditional Statement (`if`) (2)



- The condition may be of Boolean or integer type:

**conditional-statements.html**

```
var a = 0;
var b = true;
if (typeof(a)=="undefined" || typeof(b)=="undefined") {
  document.write("Variable a or b is undefined.");
}
else if (!a && b) {
  document.write("a==0; b==true;");
} else {
  document.write("a==" + a + "; b==" + b + ";");
}
```

# Switch Statement



- The `switch` statement works like in C# / Java:

**switch-statements.html**

```
switch (variable) {
  case 1:
    // do something
    break;
  case 'a':
    // do something else
    break;
  case 3.14:
    // another code
    break;
  default:
    // something completely different
}
```

# Loops



- Like in C# / Java / C++
  - `for` loop
  - `while` loop
  - `do … while` loop

```
var counter;
for (counter=0; counter<4; counter++) {
  alert(counter);
}
while (counter < 5) {
  alert(++counter);
}
```

**loops.html**

# Functions



- Code structure – splitting code into parts
- Data comes in, processed, result returned

```
function average(a, b, c)
{
    var total;
    total = a+b+c;
    return total/3;
}
```

**Parameters come in here.**

**Declaring variables is optional. Type is never declared.**

**Value returned here.**

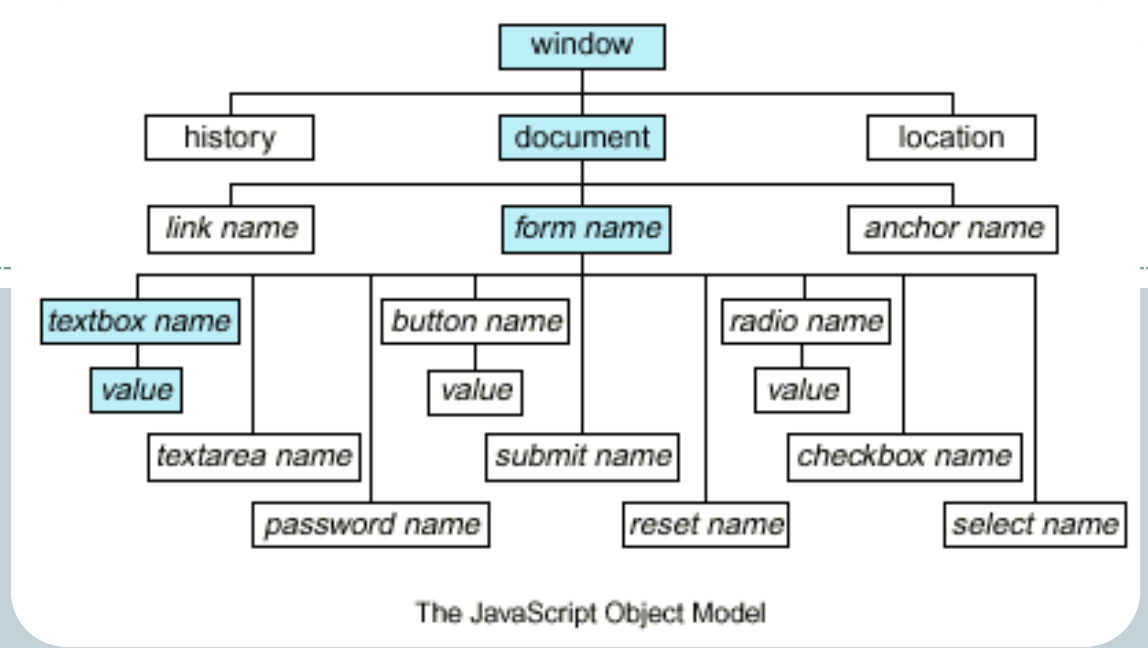
# Function Arguments and Return Value



- Functions are not required to return a value
- When calling function it is not obligatory to specify all of its arguments
  - The function has access to all the arguments passed via `arguments` array

```
function sum() {
  var sum = 0;
  for (var i = 0; i < arguments.length; i ++)
    sum += parseInt(arguments[i]);
  return sum;
}
alert(sum(1, 2, 4));
```

**functions-demo.html**




The JavaScript Object Model

# Document Object Model (DOM)

# Document Object Model (DOM)



- Every HTML element is accessible via the JavaScript DOM API
- Most DOM objects can be manipulated by the programmer
- The event model lets a document to react when the user does something on the page
- Advantages
  - Create interactive pages
  - Updates the objects of a page without reloading it

# Accessing Elements



- Access elements via their ID attribute

```
var elem = document.getElementById("some_id")
```

- Via the name attribute

```
var arr = document.getElementsByName("some_name")
```

- Via tag name

```
var imgTags = el.getElementsByTagName("img")
```

  - Returns array of descendant `<img>` elements of the element "el"

# DOM Manipulation



- Once we access an element, we can read and write its attributes

**DOM-manipulation.html**

```
function change(state) {
  var lampImg = document.getElementById("lamp");
  lampImg.src = “test_" + state + ".png";
  var statusDiv =
    document.getElementById("statusDiv");
  statusDiv.innerHTML = "The lamp is " + state";
}
…
<img id=“lamp” src="test_on.gif"
onmouseover="change('off')"
  onmouseout="change('on')" />
```

# Common Element Properties

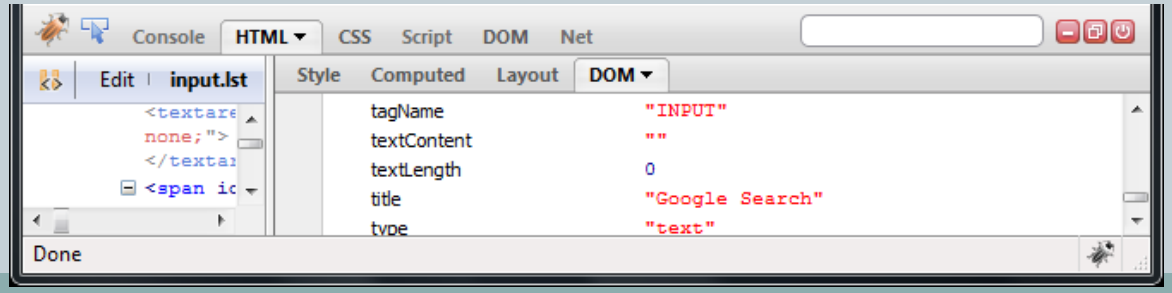


- Most of the properties are derived from the HTML attributes of the tag
  - E.g. **`id`**, **`name`**, **`href`**, **`alt`**, **`title`**, **`src`**, etc…
- `style` property – allows modifying the CSS styles of the element
  - Corresponds to the inline style of the element
    - Not the properties derived from embedded or external CSS rules
  - Example: **`style.width`**, **`style.marginTop`**, **`style.backgroundImage`**

# Common Element Properties (2)



- **className** – the `class` attribute of the tag
- **`innerHTML`** – holds all the entire HTML code inside the element
- Read-only properties with information for the current element and its state
  - **`tagName`**, **`offsetWidth`**, **`offsetHeight`**, **`scrollHeight`**, **`scrollTop`**, **`nodeType`**, etc…



# Accessing Elements through the DOM Tree Structure



- We can access elements in the DOM through some tree manipulation properties:
  - `element.childNodes`
  - `element.parentNode`
  - `element.nextSibling`
  - `element.previousSibling`
  - `element.firstChild`
  - `element.lastChild`

# Accessing Elements through the DOM Tree – Example



```
var el = document.getElementById('div_tag');
alert (el.childNodes[0].value);
alert (el.childNodes[1].
  getElementsByTagName('span').id);
…
<div id="div_tag">
  <input type="text" value="test text" />
  <div>
    <span id="test">test span</span>
  </div>
</div>
```

**accessing-elements-demo.html**

- Warning: may not return what you expected due to browser differences

# DOM Events

# DOM Events



- JavaScript can register event handlers
  - Events are fired by the Browser and are sent to the specified JavaScript event handler function
  - Can be set with HTML attributes:

```
<img src="test.gif" onclick="imageClicked()" />
```

  - Can be accessed through the DOM:

```
<img src="test.gif" id='myImage'/>

// js αρχείο:

var img = document.getElementById("myImage");
img.onclick = imageClicked;
```

Καλύτερη λύση. Με αυτόν τ
τρόπο δεν εμπλέξεται η
javascript μέσα στην HTM

# DOM Events (2)



- All event handlers receive one parameter
  - It brings information about the event
  - Contains the type of the event (mouse click, key press, etc.)
  - Data about the location where the event has been fired (e.g. mouse coordinates)
  - Holds a reference to the event sender
    - E.g. the button that was clicked

# DOM Events (3)



- Holds information about the state of [Alt], [Ctrl] and [Shift] keys
- IE does not send this object, but places it in the `window.event`
- Some of the names of the event's object properties are browser-specific

# Common DOM Events



- Mouse events:
  - `onclick`, `onmousedown`, `onmouseup`
  - `onmouseover`, `onmouseout`, `onmousemove`
- Key events:
  - `onkeypress`, `onkeydown`, `onkeyup`
  - Only for input fields
- Interface events:
  - `onblur`, `onfocus`
  - `onscroll`

# Common DOM Events (2)



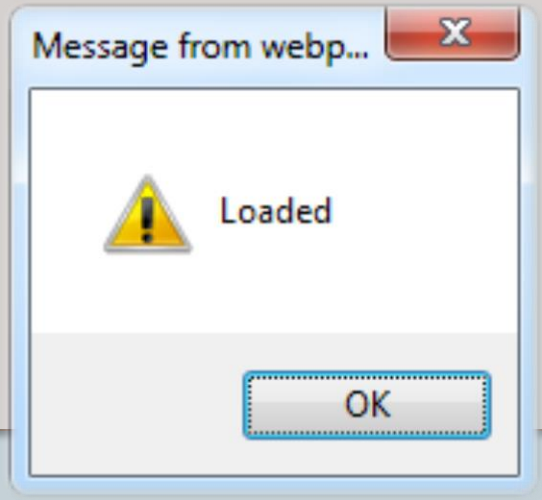
- Form events
  - `onchange` – for input fields
  - `onsubmit`
    - Allows you to cancel a form submission
    - Useful for form validation
- Miscellaneous events
  - `onload, onunload`
    - Allowed only for the `<body>` element
    - Fires when all content on the page was loaded / unloaded

# onload Event – Example



- onload event

```
<html>
<head>
  <script type="text/javascript">
    function greet() {
      alert("Loaded.");
    }
  </script>
</head>
<body onload="greet()" >
</body>
</html>
```

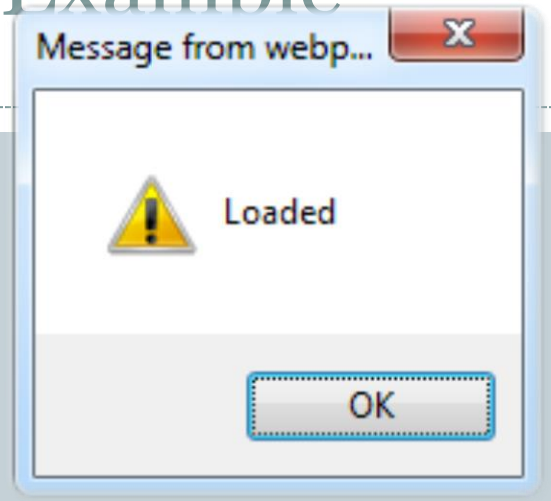


# onload Event – Example



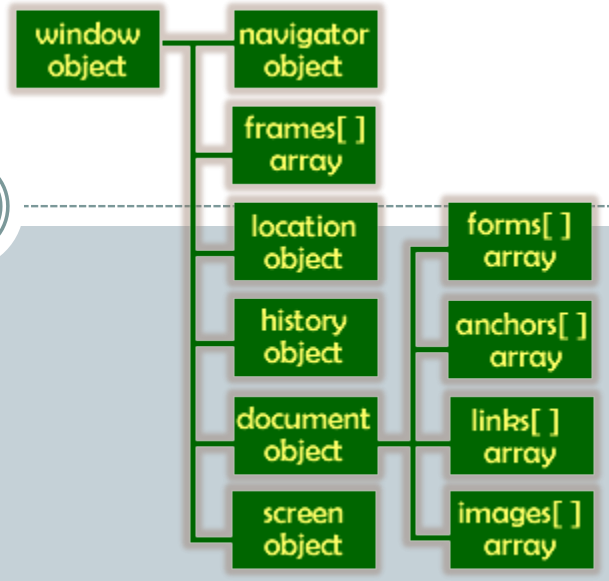
- onload event

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




```
// mycode.js
function greet() {
      alert("Loaded.");
}

document.onload=greet;
```

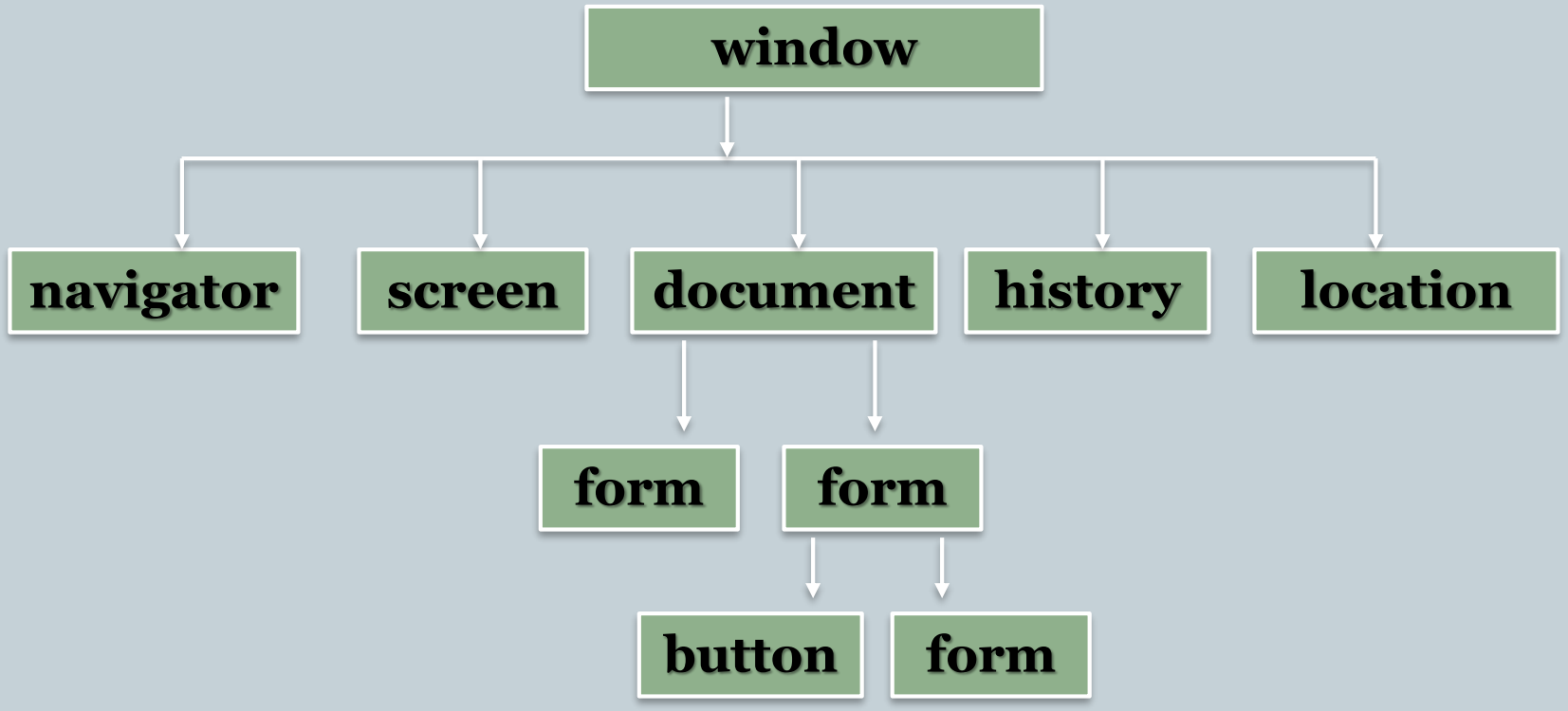



# The Built-In Browser Objects

# Built-in Browser Objects

- The browser provides some read-only data via:
  - `window`
    - The top node of the DOM tree
    - Represents the browser window
  - `document`
    - holds information the current loaded document
  - `screen`
    - Holds the user display properties
  - `browser`
    - Holds information about the browser

# DOM Hierarchy – Example

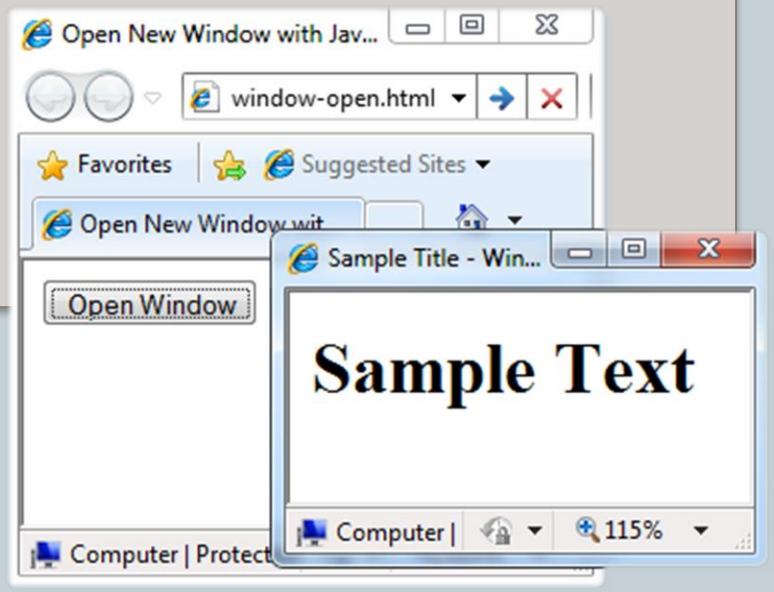


# Opening New Window – Example

- window.open()

```
var newWindow = window.open("", "sampleWindow",
  "width=300, height=100, menubar=yes,
  status=yes, resizable=yes");

newWindow.document.write(
  "<html><head><title>
  Sample Title</title>
  </head><body><h1>Sample
  Text</h1></body>");
```



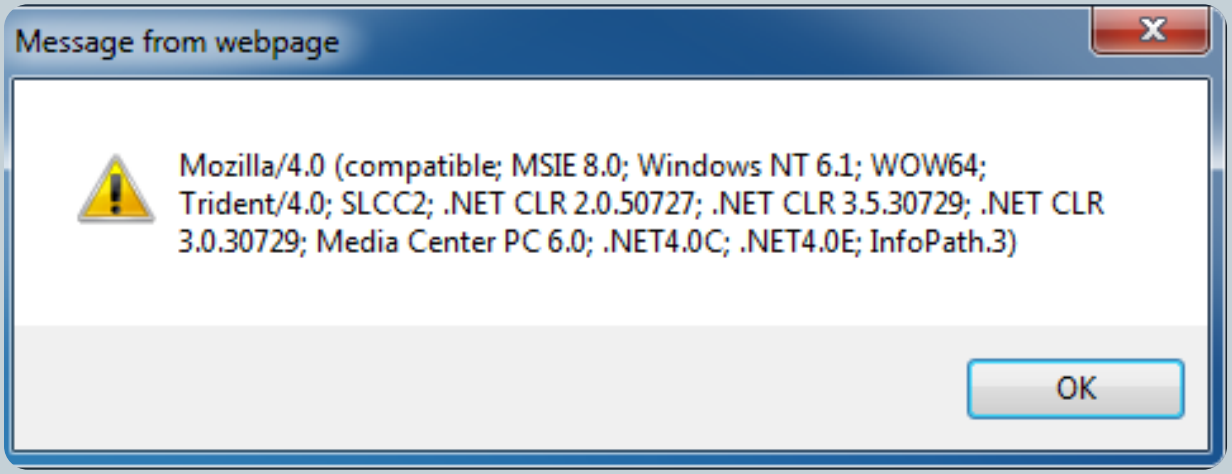
# The Navigator Object

```
alert(window.navigator.userAgent);
```

**The browser window**

**The navigator in the browser window**

**The userAgent (browser ID)**



# The Screen Object

- The `screen` object contains information about the display

```
window.moveTo(0, 0);
x = screen.availWidth;
y = screen.availHeight;
window.resizeTo(x, y);
```

# Document and Location

- `document` object
  - Provides some built-in arrays of specific objects on the currently loaded Web page

```
document.links[0].href = "yahoo.com";
document.write("This is some <b>bold text</b>");
```

- `document.location`
  - Used to access the currently open URL or redirect the browser

```
document.location = "http://www.yahoo.com/";
```

# Form Validation – Example

**form-validation.html**

```
function checkForm()
{
  var valid = true;
  if (document.mainForm.firstName.value == "") {
    alert("Please type in your first name!");
    document.getElementById("firstNameError").
      style.display = "inline";
    valid = false;
  }
  return valid;
}
…
<form name="mainForm" onsubmit="return checkForm()">
  <input type="text" name="firstName" />
  …
</form>
```

- Η jQuery είναι μια ελαφριά, cross-browser JavaScript βιβλιοθήκη (με JavaScript functions) που δίνει έμφαση στην αλληλεπίδραση μεταξύ JavaScript και HTML/DOM.
- Ανοιχτού κώδικα λογισμικό υπό διπλή άδεια χρήσης MIT License και GNU General Public Lisence, Version 2.
- Επιτρέπει  τον εύκολο χειρισμό
  - DOM – HTML
  - CSS,
  - Events
  - Ajax
  - δημιουργία  animations, οπτικών εφέ και plugins
- Μπορούμε να την προσθέσουμε στην σελίδα μας με τον ακόλουθο τρόπο:

```
<head>
<script type="text/javascript" src="http://code.jquery.com/jquery-1.10.2.js">
</script>
</head>
```