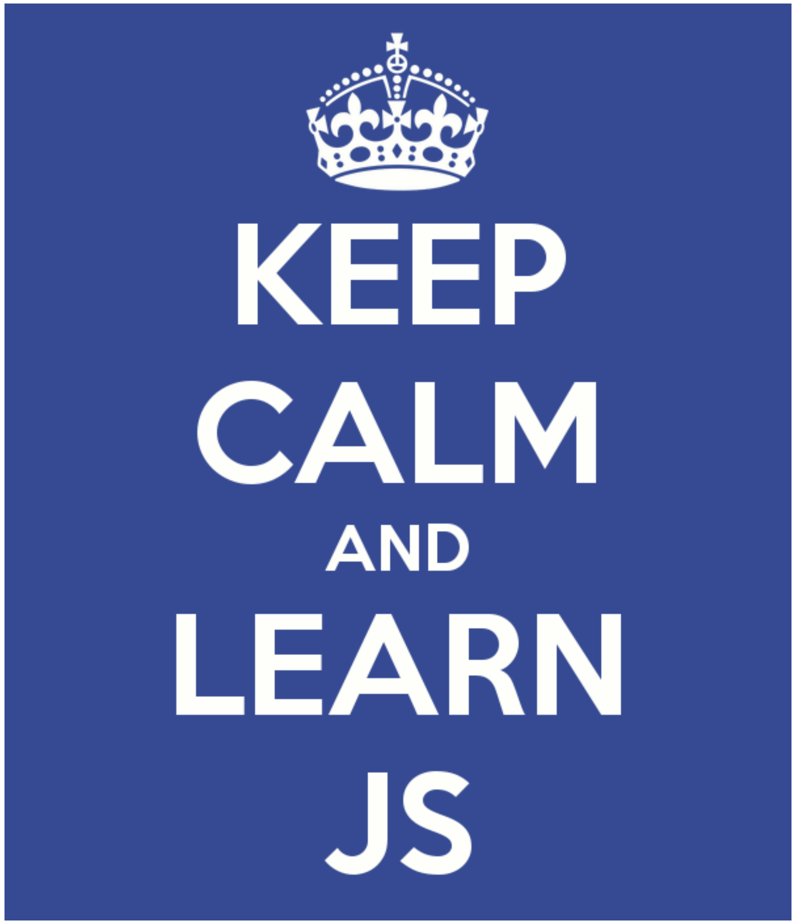
Introduction to JavaScript

Ambient intelligence: technology and design

Fulvio Corno

Politecnico di Torino, 2014/2015



Goal

- Learn about Javascript
- Learn about client-side programming mechanisms

Outline

- Introduction
- Language syntax
- Objects
- Functions
- Events
- The HTML Document Object Model
- Rich Internet Applications and AJAX

Client-side programming in the web

JAVASCRIPT / ECMASCRIPT

Client-side programming

- 4th layer of web architectures
  - Database (SQL)
  - Application server (PHP or JSP)
  - Presentation (HTML+CSS)
  - Interactivity (Javascript+DOM)
- Adds interactive functionality to client-side web pages

Client-side interactivity

- The HTML standard allows only 2 types of interaction with a page
  - Select a link (and jump to a new page)
  - Submit a form
    - Interact with form elements (input, select, ...)
- Every modification to a page requires re-loading it completely
  - Slow
  - Higher demand on the server
  - Decreases usability

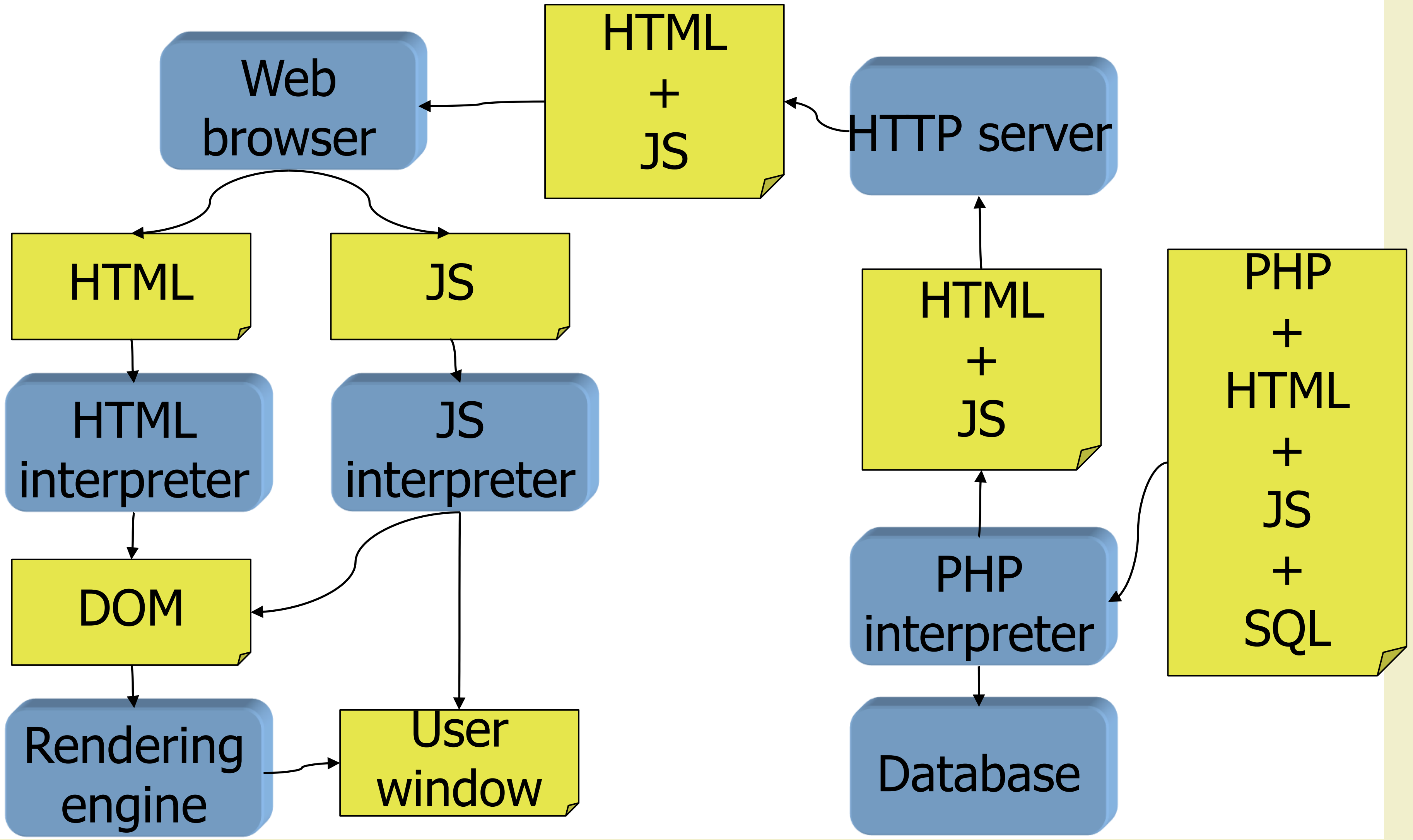
Some common problems

- Form validation
  - Avoid submitting a form unless validation rules are satisfied
  - Show validation errors immediately, and near to the error
- Form filling
  - Pre-load select lists dynamically
- Hide/show some page elements
  - Form filling instructions
  - Menus

The solution

- Add a language interpreter to the browser
- Instructions are embedded in the HTML page
  - “invisible” to the application server
  - “invisible” to the HTML presentation engine
- Instructions are processed by the browser, after HTML has been loaded

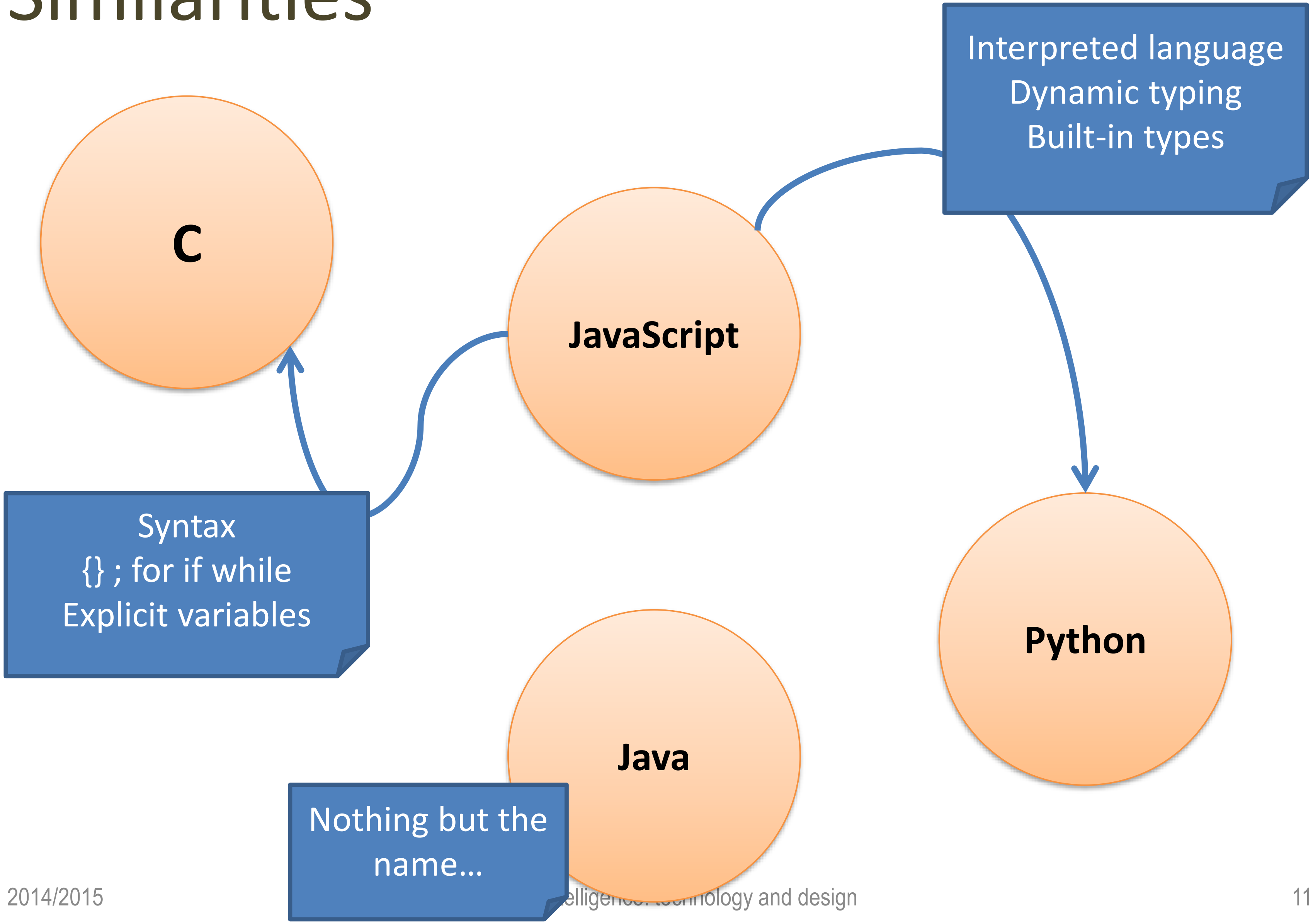
Architecture



The Javascript language

- First developed by Netscape in 1995
  - Nothing to do with the Java language, the name was chosen for marketing reasons
  - Syntax similar to C
  - Semantics of object-oriented language, with non-typed variables
- Similar versions implemented by all other browsers
  - Microsoft calls it Jscript
- Later standardized by ECMA (www.ecma.ch)
  - ECMAScript

Similarities



Embedding JS in HTML

- <script> element
- Embedded or external

Embedded JS

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

    [JavaScript code here]

// -->
</script>
```

HTML

```
<script
type="text/javascript">
// <![CDATA[

    [JavaScript code here]

// ]]>
</script>
```

XHTML

Where to embed JS code?

- In the head section: Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.
- In the body section: Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

External JS

```html
<script
     type="text/javascript"
     src="script.js"></script>
```

```html
<script type="text/javascript" src="script.js">
<!--

    [Page specific JavaScript code here]

// -->
</script>
```

Example 1

```
alert("Hello World!");
```

Exercise 1.1:
Create an HTML page including the above Javascript instruction (embedded)

Example 1

```
alert("Hello World!");
```

Exercise 1.1:
Create an HTML page including the above Javascript instruction (embedded)

Exercise 1.2:
Create a PHP page that includes a Javascript Alert than shows “Good morning” or “Good afternoon” or “Good Night” depending on the time of the day

Example 1

```
alert("Hello World!");
```

Exercise 1.1:
Create an HTML page including the above Javascript instruction (embedded)

Exercise 1.2:
at includes a Javascript
od morning" or "Good
light" depending on the
the day

Exercise 1.3:
Experiment with the following instruction:
confirm("xxx") ;

Example 2

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

Exercise 2.1:
Create an HTML page including the above Javascript instruction (embedded)

Example 2

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

Exercise 2.1:
Create an HTML page including the above Javascript instruction (embedded)

Exercise 2.2:
Create an HTML page that asks the user if it is morning, and then puts the right salutation into the body of the web page.

What more can we do?

- Generate dialog boxes
- Redirect a page
- Open new browser windows (pop-ups)
- Intercept mouse events
  - Clicks on links, buttons, …
  - Mouse-overs
- Read user input in FORMs
- Modify HTML pages
  - Add/remove content
  - Change images
  - Modify FORM controls

What should we learn?

- JS variables and expressions
- JS language constructs (if, while, ...)
- What is a JS object
- Most important builtin objects
- Interacting with the user: mouse, keyboard
- Interacting with the browser: windows, pages
- Interacting with the page: the Document object

Introduction to Javascript

LANGUAGE SYNTAX

Javascript syntax

- The syntax of the Javascript language is very similar to the C language (and to PHP)
  - Choice, Looping and other constructs are equal
  - Blocks delimited by { }
  - Most operators are identical
- Variables are different
  - Variable types
  - ‘Object’ variables

Comments

- Line comments: from // to end of line
- Block comments: from /* to */

```
//this is a comment
document.write("Hello World!")
```

```
/* This is a comment
block. It contains
several lines */
document.write("Hello World!")
```

Variables in Javascript

- A variable is identified by its name
  - Case-sensitive
  - Declared with var
- The same variable may have different values
  - Even of different data types
- Data types are converted as needed
  - If all operands are numeric, then compute a numeric result
  - If some operands are string, then convert numbers to strings

Variable declaration

- var x ;
- var x = 10 ;
- var x = "Hello" ;

Variable assignment

- var x ;
- x = 10 ;
- x = "Hello" ;
- x = x + 1 ;
- x = any complex expression

Types of variables

- Boolean (false, true)
- Numbers
  - var x = 10
  - var y = 3.14
- Strings
  - var name = "Fulvio"
- ‘Objects’
  - var d = new Date()
  - var time = d.getHours()

Main Javascript operators (1/3)

- Numeric operators
  - +
  - -
  - *
  - /
  - % (remainder, or modulus)
- Increment operators
  - ++
  - --
- Assignment operators
  - =
  - += -= *= /= %=

Main Javascript operators (2/3)

- String operator
  - + (concatenation)
- Comparison operators
  - == (same value)
  - === (same value and same type)
  - !=
  - >
  - <
  - >=
  - <=

Main Javascript operators (3/3)

- Boolean and Logic operators
  - && (logical “and”)
  - || (logical “or”)
  - ! (logical “not”)

Warning

- String concatenation operator (+) is identical to numeric addition
  - Possible ambiguity
  - 3 + 2
  - "3" + "2"
- Difference between == and ===
  - 5 == "5"
  - 5 === 5
  - "5" === "5"
  - Not true: 5 === "5"

Choice statements (1/2)

```
if (condition)
{
  ...code...
}
```

```
if (condition)
{
  ...code if true...
}
else
{
  ...code if false...
}
```

```
if (condition1)
{
  ...code if 1 true...
}
else if (condition2)
{
  ...code if 2 true...
}
else
{
  ...if both false...
}
```

Choice statements (2/2)

```
switch(n)
{
case 1:
  code block 1
  break

case 2:
  code block 2
  break

default:
  code to be executed if n is
  different from case 1 and 2
}
```

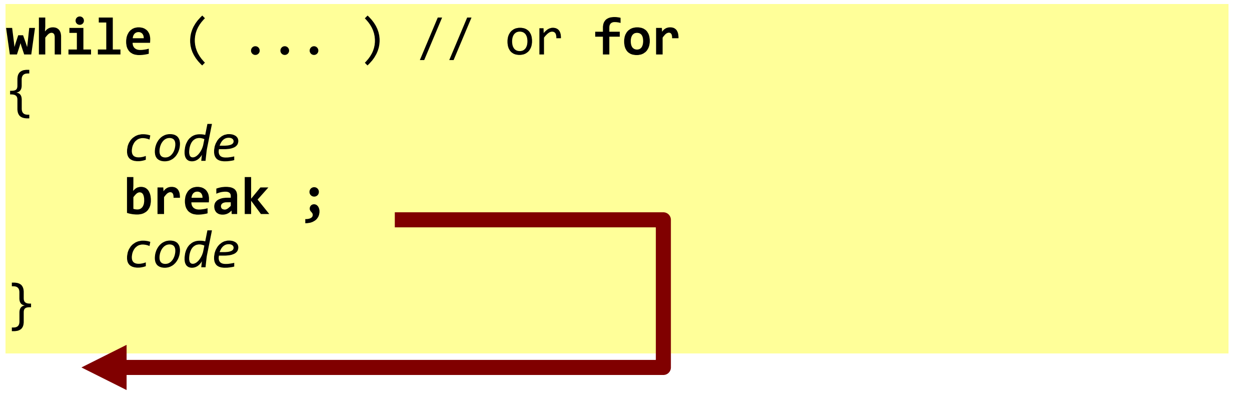
Loop statements (1/2)

```
for ( v = startvalue;
      v < endvalue;
      v = v+increment )
{
    code to be executed
}
```

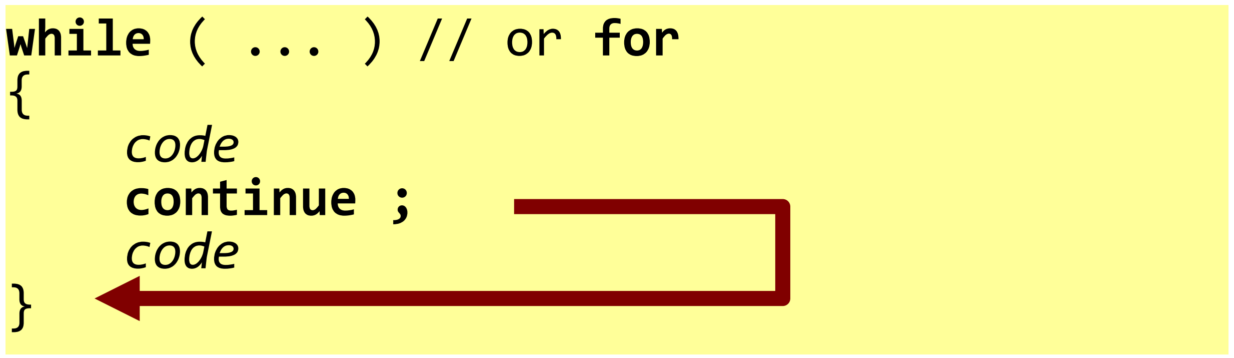
```
while ( condition_is_true )
{
    code to be executed
}
```

```
do {
    code to be executed
} while ( condition_is_true )
```

Loop statements (2/2)



```
while ( ... ) // or for
{
    code
    break ;
    code
}
```



```
while ( ... ) // or for
{
    code
    continue ;
    code
}
```

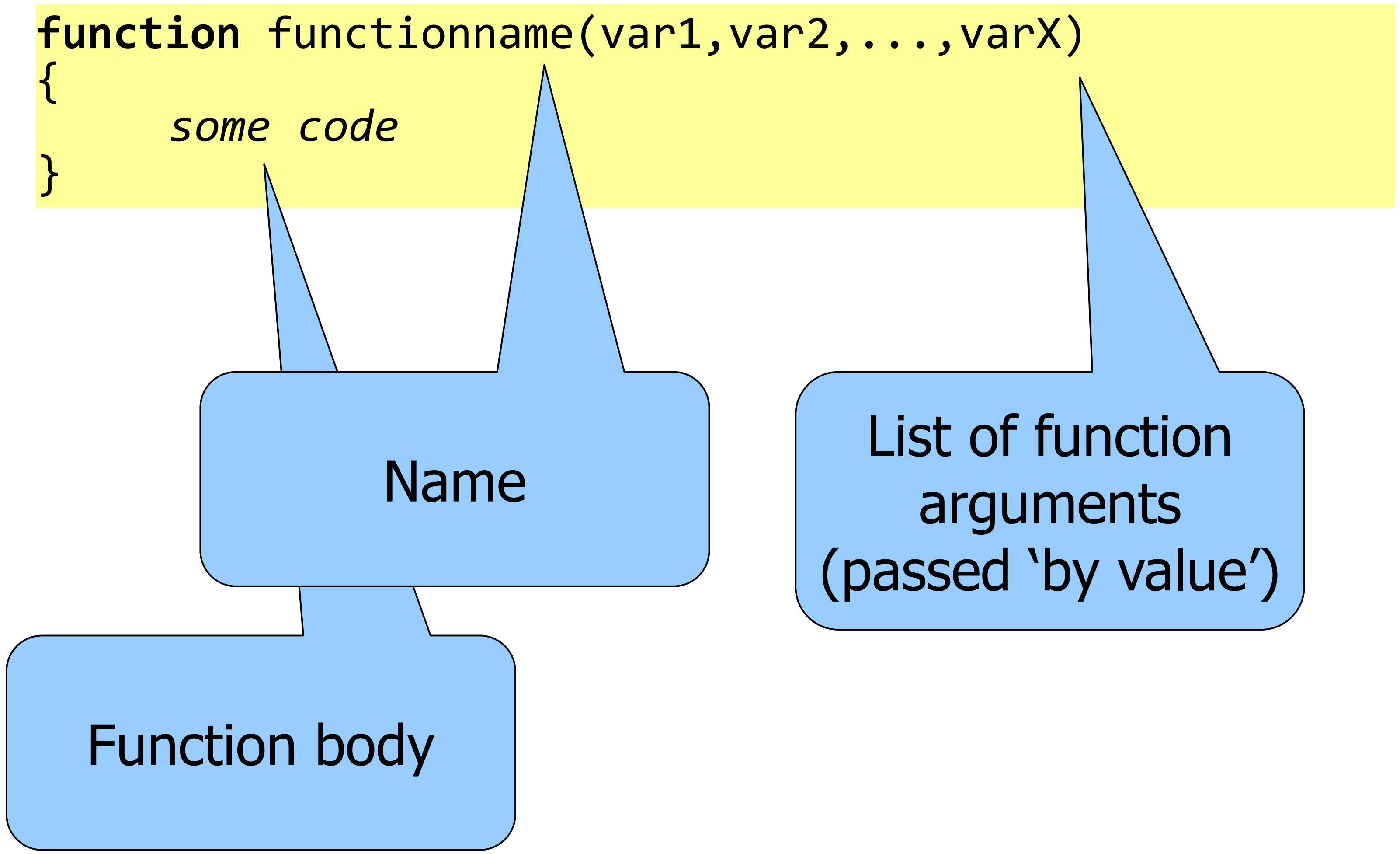
Basic interaction methods

- Popup box (OK to confirm)
  - alert("text")
- Confirm box (OK, cancel)
  - confirm("text")
  - True if user clicked on OK
- Prompt box (let user insert a text)
  - prompt("prompt text", "initial value")
  - Returns a string with the text inserted by the user
  - Returns null if user clicked on Cancel

Introduction to Javascript

FUNCTIONS

Defining a new function (1/2)



```
function functionname(var1,var2,...,varX)
{
     some code
}
```

Defining a new function (2/2)

```
function functionname(var1,var2,...,varX)
{
        some code
}
```

```
function functionname()
{
        some code
}
```

No parameters

Return statement

- A function may return a value to its caller by executing the return statement
  - return value ;
- The value may be of any type (boolean, numeric, string, ...)

Example

```
<html>
<head>
<script type="text/javascript">
     function product(a,b)
     {
       return a*b;
     }
</script>
</head>

<body>
<script type="text/javascript">
       document.write(product(4,3)) ;
</script>
</body>
</html>
```

Introduction to Javascript

OBJECTS

Objects in Javascript

- An object is a complex data type characterized by
  - A current value
    - Sometimes the internal value is “hidden”
  - A set of properties
    - Various values that be read, associated in some way to the object value
    - Some values that may be written, that modify in some way the object value
  - A set of methods
    - Operations (with parameters) that can be asked to the object

Using objects

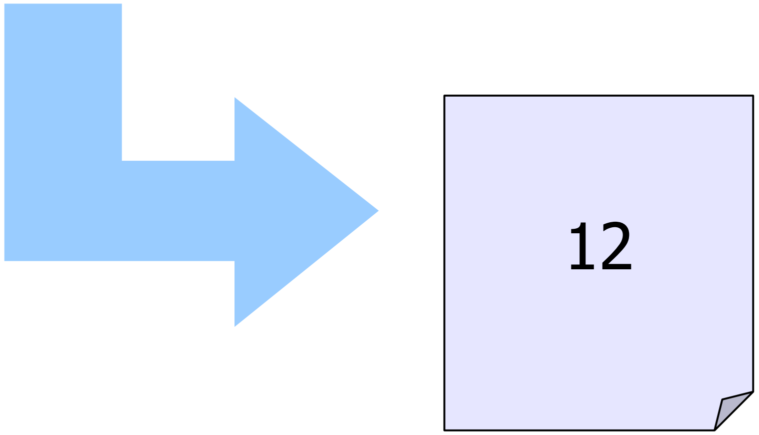
- Creating new objects
  - var d = new Date()
    - Create a new Object of type Date, and use the variable d as a reference to that object
- Properties and methods
  - var day = d.getDay() ;
  - d.setMinutes(34) ;

String objects

- Strings are used to store and manipulate sequences of characters
- Constant values are written between quotes "Hello"
- The only property is
  - .length (the number of characters in the string)
- Many methods implement several string operations

Example

```
var txt="Hello world!"
document.write(txt.length)
```



```
12
```

String methods (1/2)

- Access to the i-th character (starting from 0)
  - s.charAt(i)
- Concatenate two strings
  - s3 = s1.concat(s2)
- Find a substring
  - i = s.indexOf("abc") // -1 if not found
  - j = s.indexOf("abc", i+1)
  - s.lastIndexOf searches from the end
- Replace
  - s = s.replace("Belusconi", "Prodi")

String methods (2/2)

- Extract substring
  - s1 = s.substr(startPos, numChars)
  - s1 = s.substr(startPos) // until the end
  - s1 = s.substring(startPos, endPos)
- Case conversion
  - upper = s.toUpperCase()
  - lower = s.toLowerCase()

String methods for HTML formatting

- The String object has several methods to insert tags around the specified string
  - .big(), .small(), .italic(), .bold(), .fixed()
  - .fontcolor(c), .fontsize(s),
  - .anchor("name"), .link("url")

```
var txt="Hello world!"
document.write(txt.bold())
```

```
<b>Hello world!</b>
```

Exercise 1

- Use a pop-up window to ask the user his/her name
- Write the user’s name in the page heading <h1>

Exercise 2

- Use a pop-up window to ask the user his/her name
- Write the user’s name in the page heading <h1>, properly formatting it in “title case”
  - Example: if name = “fulvio CORNO”, then print “Fulvio Corno”

Date objects

- The Date object is used to work with dates and times
- New objects are created with the current timestamp
  - var d = new Date() // now!
- A specific value may be set
  - d.setFullYear(2007, 04, 23)
  - d.setHours(23, 59, 00)

Date querying methods

- Return numeric components of the date and time stored in the object:
  - .getDate(), .getDay() /*of week*/, .getMonth(), .getFullYear()
  - .getHours(), .getMinutes(), .getSeconds(), .getMilliseconds()
- Return a string representing the date
  - .toString(), .toLocaleString()
- Return milliseconds since 01/01/1970
  - .getTime()

Date setting methods

- Setting date and time from numeric components
  - .setMonth(m), .setDate(day_of_month), .setFullYear(y), .setFullYear(y, m, d)
  - .setHours(h), .setMinutes(m), setSeconds(s), setHours(h, m, s)
- Setting a date from a string
  - Date.parse("Apr 23, 2007") returns the number of milliseconds
  - d.setTime(Date.parse("Apr 23, 2007"))

Exercise 3

- Modify Exercise 2, and write the current date and time in the footer of a web page
- Add a salutation (Good Morning, Good Afternoon, Good Night, ...) according to the current time of the day
  - The salutation must be in the same <h1> as the name

Array objects

- Creating an empty array
  - var a = new Array()
  - var a = new Array(maxsize)
- Setting values
  - a[0] = "Fulvio"
  - a[1] = "Dario"
- Using values
  - document.write(a[0])
  - var s = a[1].toUpperCase()

Array properties

- The property .length returns the number of elements in the array
  - var N = a.length

```
var mycars = new Array()
mycars[0] = "Saab"
mycars[1] = "Volvo"
mycars[2] = "BMW"

for (i=0;i<mycars.length;i++)
{
  document.write(mycars[i] + "<br />")
}
```

Array methods (1/2)

- Concatenate two arrays
  - a3 = a1.concat(a2)
  - Creates a new array with all elements from a1, followed by all elements from a2
- Extract a sub-array
  - a2 = a1.slice(start_index, end_index)
- Sort in alphabetical order
  - a2 = a.sort()

Array methods (2/2)

- Convert an array to a string
  - var s = a.join() // "abc,def"
  - var s = a.join("-") // "abc-def"
- Convert a string to an array
  - var a = s.split(",")

Esercise 4

- Collect a set of number from the user
  - Each number in inserted in a pop-up window
  - The insertion is terminated by pressing Cancel
- Print in the HTML page the list of all inserted numbers
- Print in the HTML page the maximum, minimum and average of the inserted numbers

Math object

- The Math object is a special object: no variables may be created, but a lot of methods are defined, that may be called
- Think of Math as a “library” of mathematical functions

Math contants

- Math.E
- Math.PI
- Math.SQRT2 // √2
- Math.SQRT1_2 // √(1/2)
- Math.LN2 // loge(2)
- Math.LN10 // loge(10)
- Math.LOG2E // log2(e)
- Math.LOG10E // log10(e)

Math functions (1/2)

- Trigonometric
  - Math.cos(x), Math.sin(x), Math.tan(x), Math.acos(x), Math.asin(x), Math.atan(x), Math.atan2(y, x)
- Exponential and logarithmic
  - Math.exp(x), Math.log(x), Math.pow(base,exp), Math.sqrt(x)

Math functions (2/2)

- Truncation and rounding
  - Math.ceil(x), Math.floor(x), Math.round(x)
- Signs and comparisons
  - Math.abs(x), Math.max(a,b), Math.min(a.b)
- Random
  - Math.random() // random number in interval [0,1)

Exercise 5

- Write a Javascript program to play the “Guess a number” game
- The program must generate a secret number between 1 and 100
- The user inserts a set of guesses into a pop-up windows
- Each time, the program tells the user if the guess was too high or too low
- The HTML page, at the end, will show the list of all guesses, and the number of attempts

Introduction to Javascript

EVENTS

Javascript event model

- An event is the indication that something happened on a web page
  - Some user interaction (click, move mouse, ...)
  - Some browser action (load page, ...)
- In Javascript, you may attach an event handler to most events
  - Any Javascript function
  - The Javascript interpreter calls the function anytime the event is generated

Example

```
<html>
  <head>
    <script type="text/javascript">
      function sayHello()
      {
        alert("Hello!")
      }
    </script>
  </head>

  <body>
    <form>
      <input type="button" onclick="sayHello()"
        value="Press me">
    </form>
  </body>
</html>
```

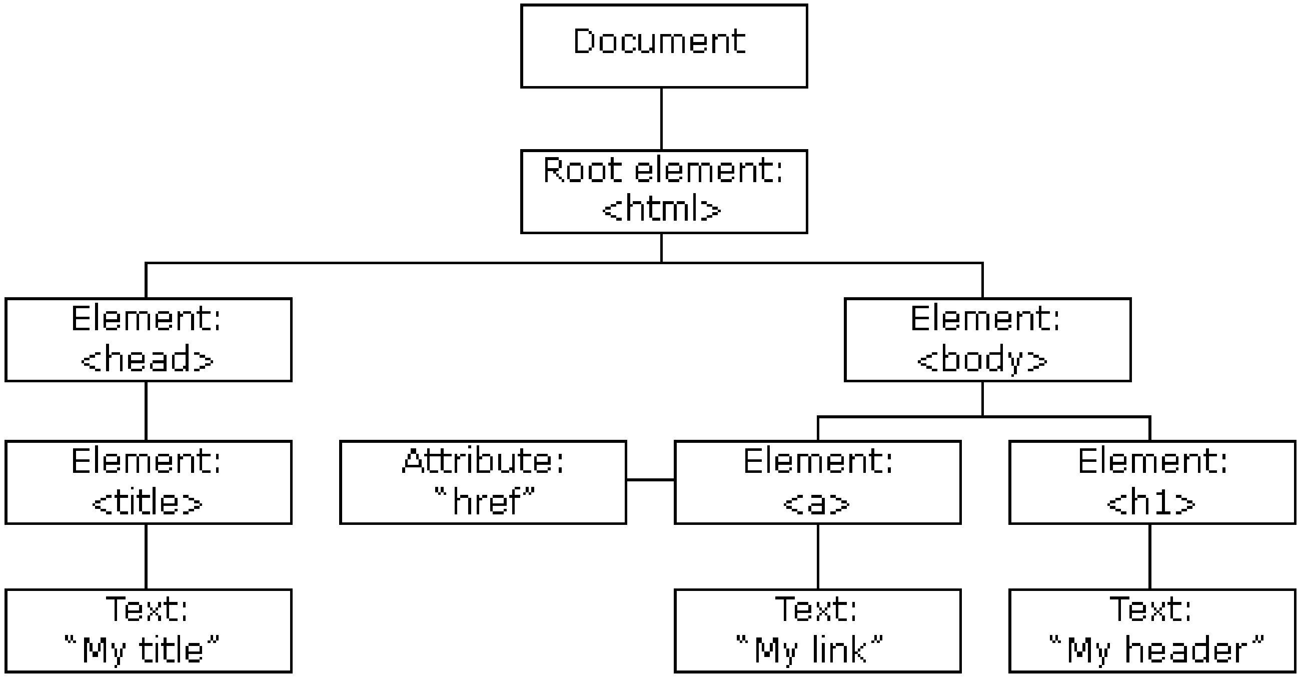
HTML Document Object Model (DOM)

HTML DOCUMENT OBJECT MODEL (DOM)

Document Object Model

- The HTML Document Object Model (HTML DOM) defines a standard way for accessing and manipulating HTML documents.
- The DOM presents an HTML document as a tree-structure (a node tree), with elements, attributes, and text.

DOM example


Document
Root element: <html>
Element: <head>
Element: <body>
Element: <title>
Attribute: "href"
Element: <a>
Element: <h1>
Text: "My title"
Text: "My link"
Text: "My header"

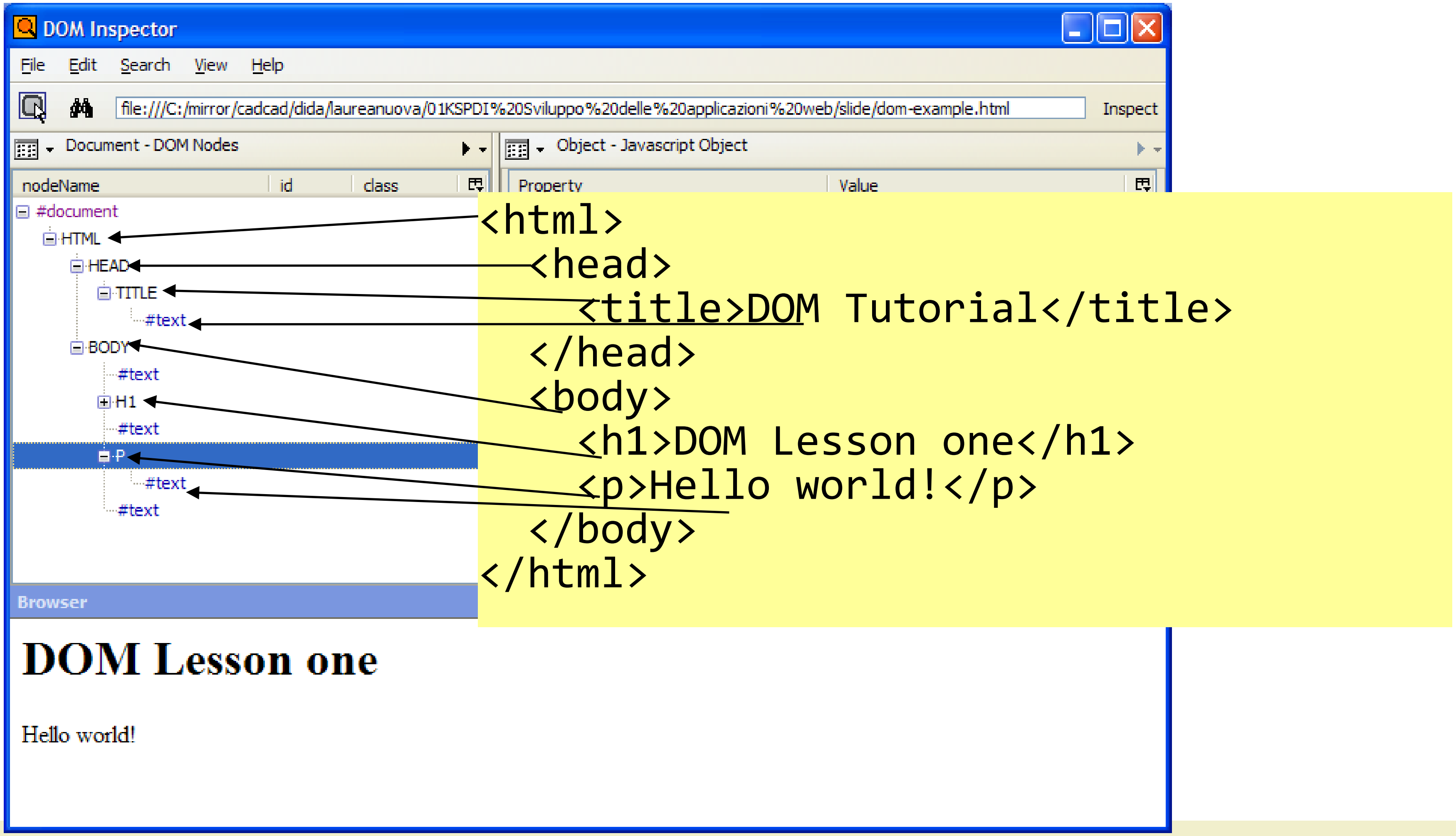
DOM structure

- The entire document is a document node
- Every HTML tag is an element node
- The texts contained in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes
- Nodes have a hierarchical relationship to each other

Example

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

Example


DOM Inspector
File Edit Search View Help
file:///C:/mirror/cadcad/dida/laureanuova/01KSPDI%20Sviluppo%20delle%20applicazioni%20web/slide/dom-example.html
Inspect
Document - DOM Nodes
Object - Javascript Object
nodeName
id
class
Property
Value
Browser

```
#document
  HTML
    HEAD
      TITLE
        #text
    BODY
      #text
      H1
      #text
      P
        #text
      #text
```

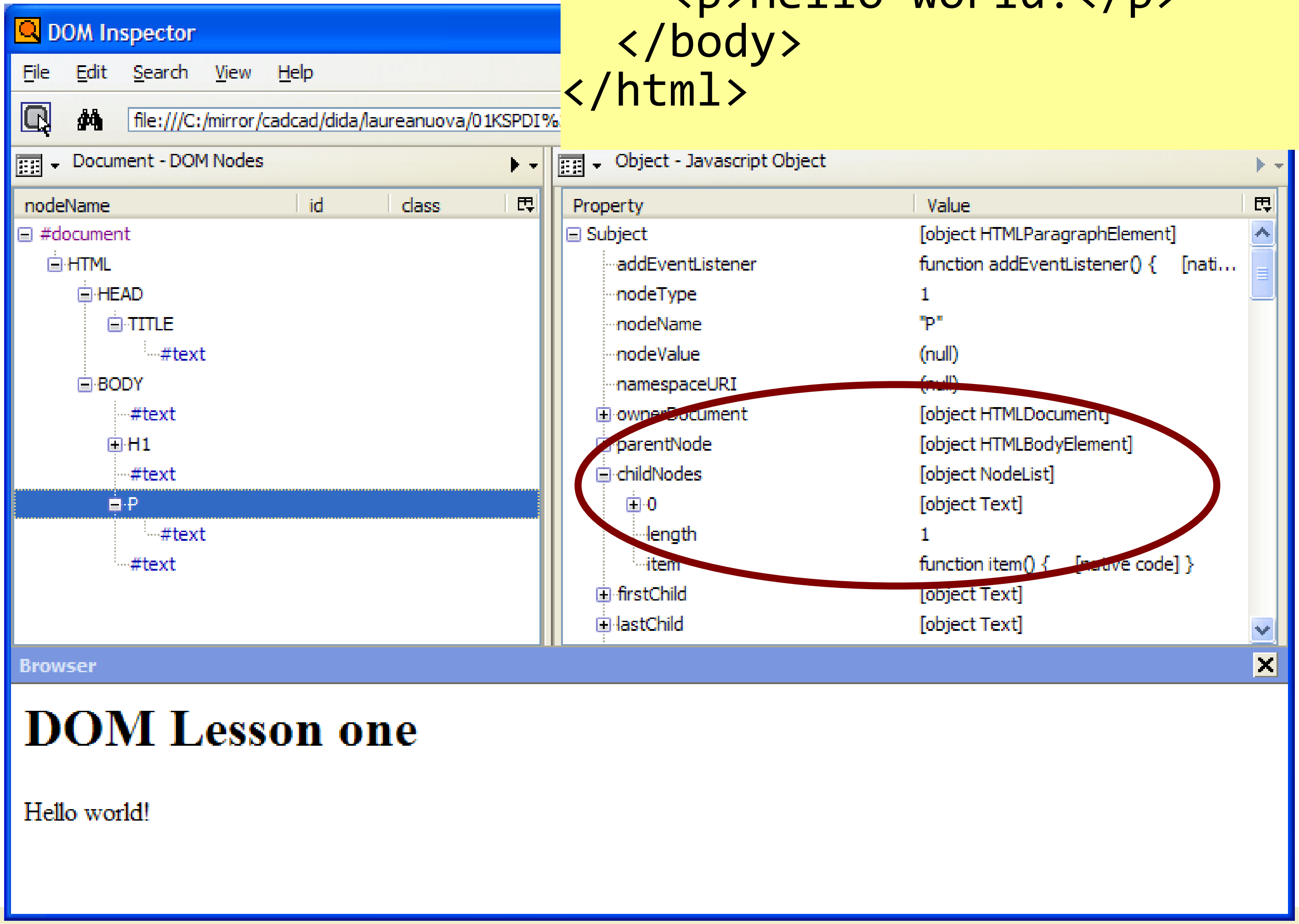
```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

DOM Lesson one

Hello world!

Example

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```


DOM Inspector
File Edit Search View Help
file:///C:/mirror/cadcad/dida/laureanuova/01KSPDI%
Document - DOM Nodes
Object - Javascript Object
Browser

```
nodeName            id    class
#document
  HTML
    HEAD
      TITLE
        #text
    BODY
      #text
      H1
      #text
      P
        #text
      #text
```

Property	Value
Subject	[object HTMLParagraphElement]
addEventListener	function addEventListener() { [nati...
nodeType	1
nodeName	"P"
nodeValue	(null)
namespaceURI	(null)
ownerDocument	[object HTMLDocument]
parentNode	[object HTMLBodyElement]
childNodes	[object NodeList]
0	[object Text]
length	1
item	function item() { [native code] }
firstChild	[object Text]
lastChild	[object Text]

DOM Lesson one

Hello world!

Javascript and the DOM

- Each node in the HTML DOM is automatically available as a corresponding Javascript object
- Methods and properties of the object correspond to content and attributes of the HTML element
- Any modification to the object fields are immediately reflected in the HTML page
- The object “document” is the top of the HTML page

Finding objects

- Alternative methods
  - Navigating through children and siblings, starting from the document node
  - Identifying specific elements by their tag name
    - Use getElementsByTagName("tag")
    - Returns all the elements with that tag
  - Identifying specific elements by their “id” attribute (recommended!)
    - Add an “id” attribute, with a unique value, to any HTML tag
    - Use getElementById("id")

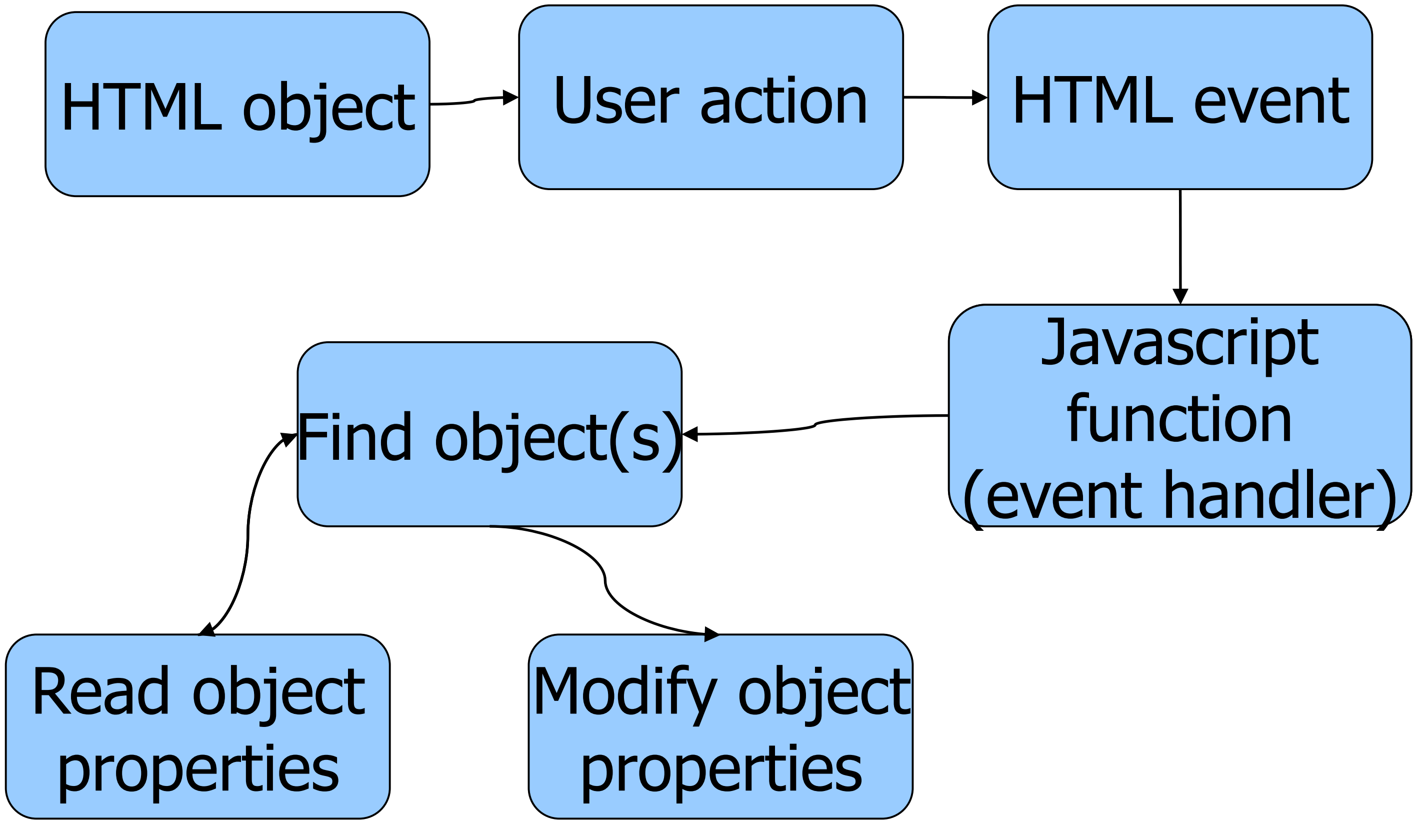
Example (1/2)

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1 id="banner">DOM Lesson two</h1>
    <p  id="mytext">Hello world!</p>

    <script>...</script>

  </body>
</html>
```

Example (2/2)

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

  var x = document.getElementById("banner") ;
  alert( x.firstChild.nodeValue ) ;

  var y = document.getElementById("mytext") ;
  y.firstChild.nodeValue = "Hello again...." ;

</script>
```

Control sequence

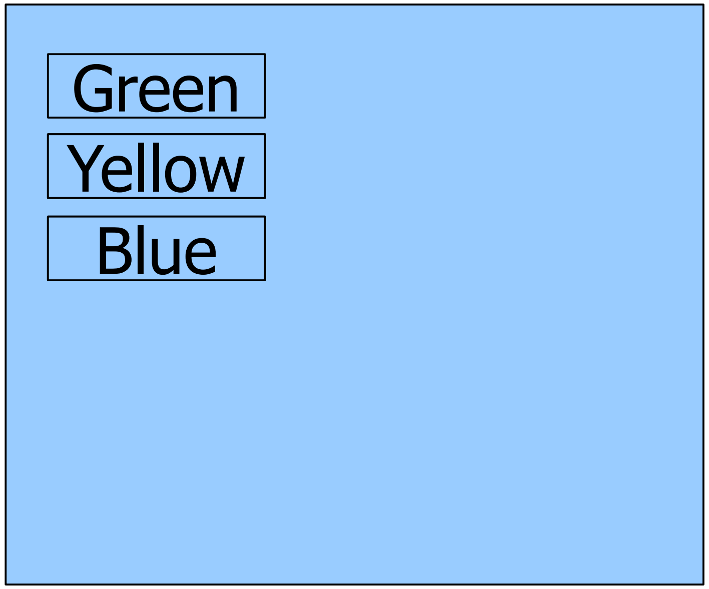


HTML events

<body>	`onload`
<body>	`onunload`
Form elements	`onchange`
Form elements	`onsubmit`
Form elements	`onreset`
Form elements	`onselect`
Form elements	`onblur`
Form elements	`onfocus`
Any element – keyboard	`onkeydown`
Any element – keyboard	`onkeypress`
Any element – keyboard	`onkeyup`
Any element – mouse	`onclick`
Any element – mouse	`ondblclick`
Any element – mouse	`onmousedown`
Any element – mouse	`onmousemove`
Any element – mouse	`onmouseover`
Any element – mouse	`onmouseout`
Any element – mouse	`onmouseup`

Exercise 6

- Create an HTML page with variable-color background.
- The background color is selected by the user by clicking on suitable text sentences



Form submission

- The submission of FORM data may be intercepted by the onsubmit event
- The event procedure may check for any errors
  - If everything is ok, the function returns true -> the browser takes the form action
  - In case of errors, the function returns false -> the form is not submitted

Exercise 7

- Create an HTML form for entering a username/password pair
- Do not allow the user to press the submit button unless:
  - Both username and password are present
  - Password is more than 4 characters long

Exercise 7b

- Create an HTML form for entering a username/password pair
- Do not allow the user to press the submit button unless:
  - Both username and password are present
  - Password is more than 4 characters long
- Whenever the user commits an error, display a message just besides the text box

Exercise 8

- Create an HTML form for selecting an item from a list of categories, including a “Other...” option
- If the user selects “Other...”, then he must fill a text box for specifying
- Otherwise, the text box should be invisible

References

- JavaScript Tutorial, http://www.w3schools.com/js/default.asp
- http://www.quirksmode.org/js/contents.html
- JavaScript Reference, http://www.w3schools.com/jsref/default.asp
- Standard ECMA-262 (3r d Edition - December 1999), http://www.ecma-international.org/publications/standards/Ecma-262.htm

Web development

Client-side programming

Rich Internet Applications and AJAX

Rich Internet Application

Rich Internet applications (RIA) are web applications that have the features and functionality of traditional desktop applications. RIAs typically

- transfer the processing necessary for the user interface to the web client
- keep the bulk of the data (i.e., maintaining the state of the program, the data etc) back on the application server.

Main goals of RIAs

Most sophisticated RIAs exhibit a look and feel approaching a desktop environment.

- **Richer**. User-interface behaviors not obtainable using only the HTML widgets available to standard browser-based Web applications: drag and drop, using a slider to change data, calculations performed by the client and which do not need to be sent back to the server, ...
- **More responsive**. The interface behaviors are typically much more responsive than those of a standard Web browser that must always interact with a remote server.

Performance of RIAs

Client/Server balance. The demand for client and server computing resources is better balanced. This frees server resources, allowing the same server hardware to handle more client sessions concurrently.

Performance of RIAs

Asynchronous communication. The client engine can interact with the server without waiting for the user to perform an interface action such as clicking on a button or link. This allows the user to view and interact with the page asynchronously from the client engine's communication with the server.

- Example: prefetching (an application anticipates a future need for certain data, and downloads it to the client before the user requests it)

Performance or RIAs

Network efficiency. Network traffic may be significantly reduced because an application-specific client engine can be more intelligent than a Web browser when deciding what data needs to be exchanged with servers.

- Less data is being transferred for each interaction, and overall network load is reduced.
- However, use of asynchronous prefetching techniques can neutralize or even reverse this potential benefit.

AJAX definition

Asynchronous JavaScript And XML.
AJAX is a type of programming made popular in 2005 by Google (with Google Suggest).
AJAX is not a new programming language, but a new way to use existing standards.
With AJAX you can create better, faster, and more user-friendly web applications.
AJAX is based on JavaScript and HTTP requests.

Key enabling technology

With AJAX, your JavaScript can communicate directly with the server, using the JavaScript XMLHttpRequest object.
By using the XMLHttpRequest object, a web developer can update a page with data from the **server -- after** the page has loaded!
The XMLHttpRequest object is supported in Internet Explorer 5.0+, Safari 1.2, Mozilla 1.0 / Firefox, Opera 8+, and Netscape 7.
http://www.w3.org/TR/XMLHttpRequest/

XMLHttpRequest – the name

The name of the object is *wrong*, but maintained for historical reasons:

- May receive any text-based content, not just XML
- May use also HTTPS, not just HTTP protocol
- May handle both Requests and Responses, of all HTTP methods

Standard definition

```
interface XMLHttpRequest {
  // event handler
        attribute EventListener onreadystatechange;
  // state
  const unsigned short UNSENT = 0;
  const unsigned short OPENED = 1;
  const unsigned short HEADERS_RECEIVED = 2;
  const unsigned short LOADING = 3;
  const unsigned short DONE = 4;
  readonly attribute unsigned short readyState;
```

Standard definition

```
  // request
  void open(in DOMString method, in DOMString url);
  void open(in DOMString method, in DOMString url, in boolean async);
  void open(in DOMString method, in DOMString url, in boolean async, in
DOMString user);
  void open(in DOMString method, in DOMString url, in boolean async, in
DOMString user, in DOMString password);
  void setRequestHeader(in DOMString header, in DOMString value);
  void send();
  void send(in DOMString data);
  void send(in Document data);
  void abort();
```

Standard definition

```
  // response
  DOMString getAllResponseHeaders();
  DOMString getResponseHeader(in DOMString header);
  readonly attribute DOMString responseText;
  readonly attribute Document responseXML;
  readonly attribute unsigned short status;
  readonly attribute DOMString statusText;
};
```

Request states

UNSENT = 0

    The request is not initialized

OPENED = 1

    The request has been set up

HEADERS_RECEIVED = 2

    The request has been sent

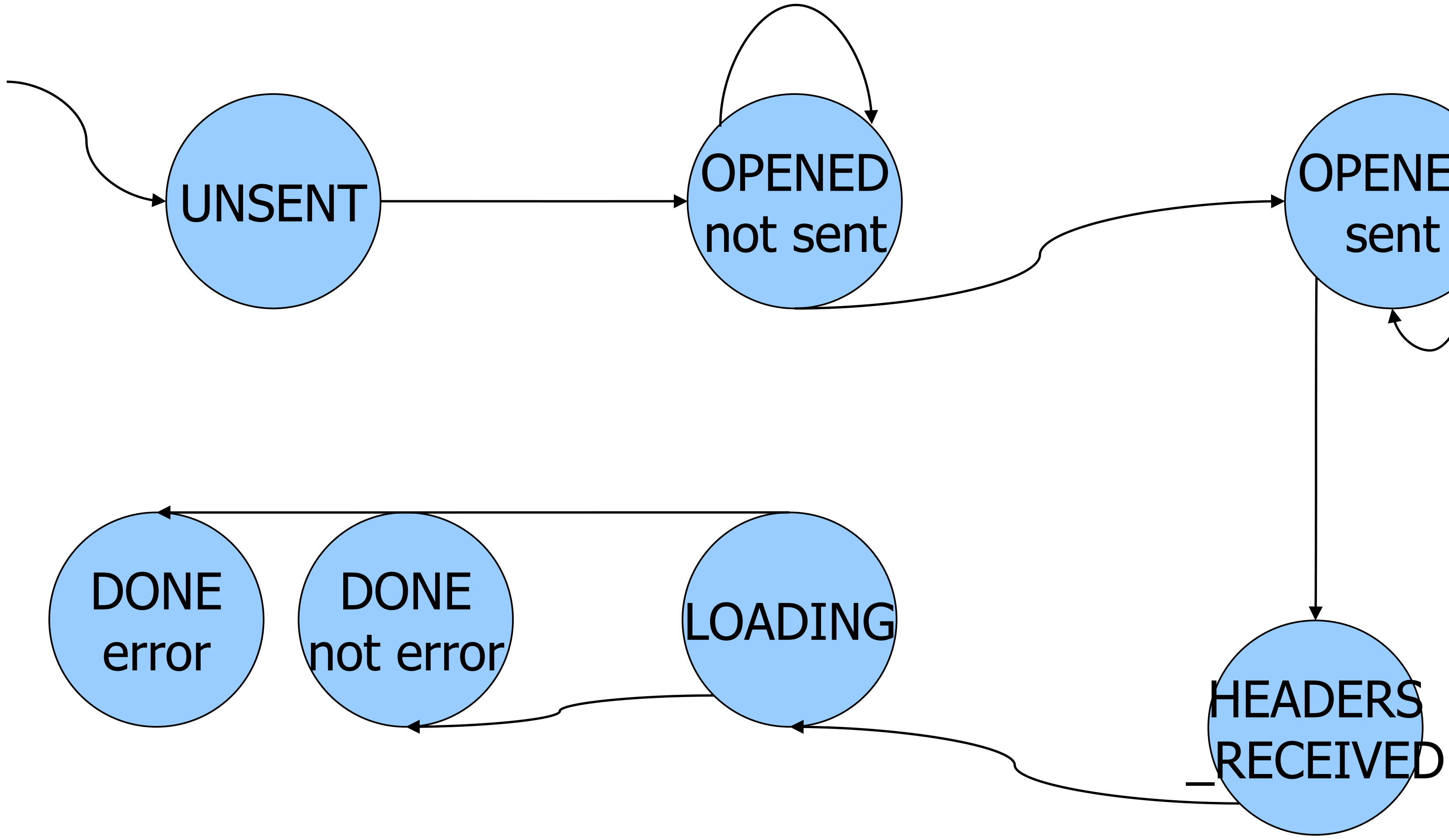
LOADING = 3

    The request is in process

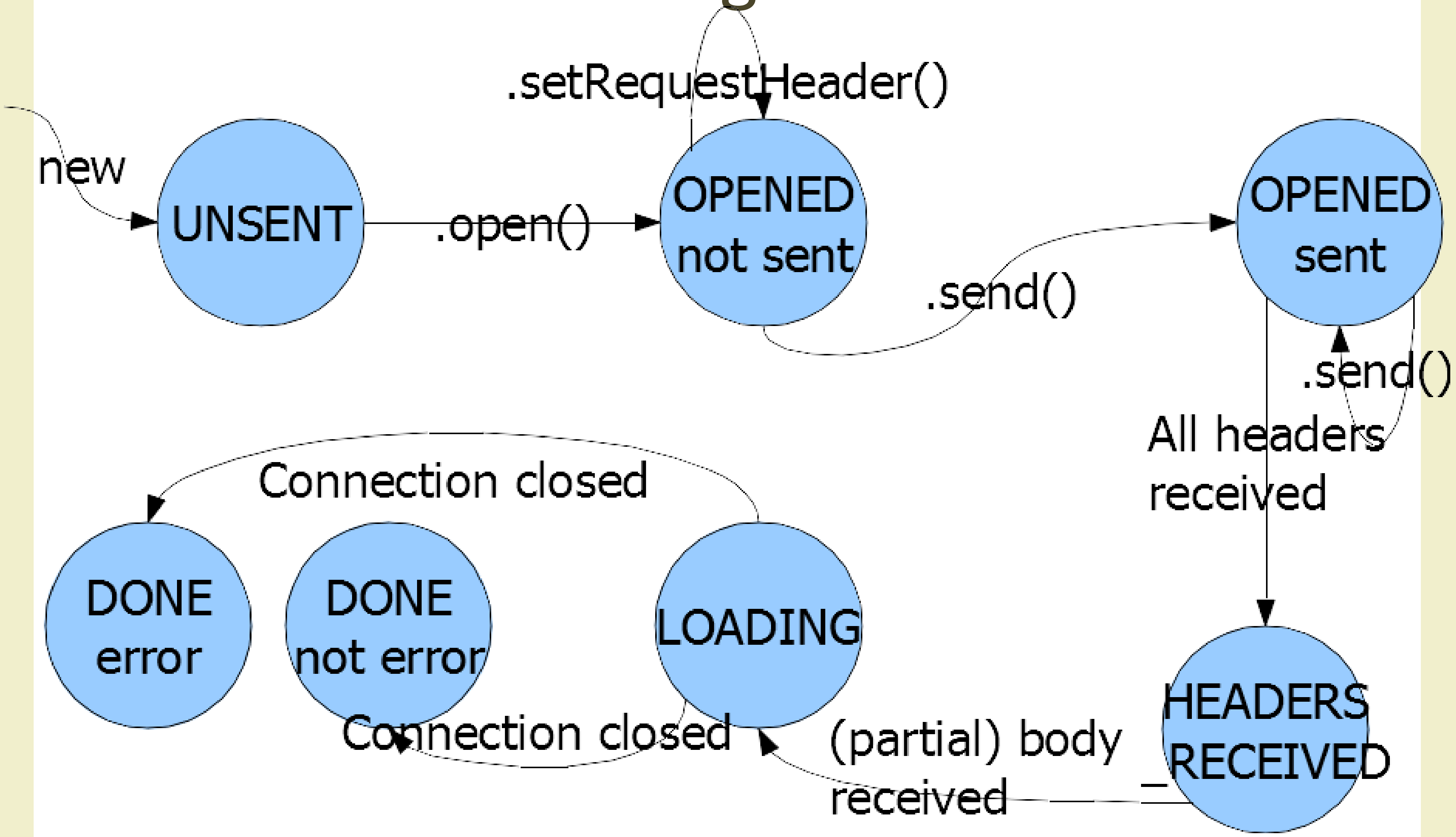
DONE = 4

    The request is complete

State transition diagram



State Transition Diagram



XMLHttpRequest properties

onreadystatechange

stores the **function** that will process the response from a server

```
xmlHttp.onreadystatechange =
  function() { ... }
```

readyState

holds the status of the server's response. Each time readyState changes, the onreadystatechange function will be executed.

responseText

the data sent back from the server can be retrieved with the responseText property

Methods

- open(method, url, async, user, password)
  - method = “GET”, “POST”
  - url = complete URL to request
  - async = true/false (optional, default=true)
  - user, password (optional)
  - Interrupts any on-going send()
- setRequestHeader(header, value)
  - Adds a new header to the HTTP Request
  - Content-Type is one common header to send
    - Examples: text/xml, application/xml

Methods

send(data)

- Initiates the request
- data = HTTP request body (optional)
  - May be a Document or DOMString
- The URL was already given in open()
- send() terminates immediately if async==true, but transfer continues in the background
  - Generates readystatechange events
- send() transfers data synchronously if async==false

Methods

getAllResponseHeaders()

- Return all response headers as a single string, with headers separated by CR+LF
- Invalid if UNSENT or OPENED

getResponseHeader(header)

- Returns the value of a single header
- Invalid if UNSENT or OPENED

Receiving the response body

- responseText of type DOMString
  - If LOADING (partial body) or DONE
  - Allow access to a “raw string” of the response body
- responseXML of type Document
  - Only if DONE
  - For text/xml (or application/xml or *+xml) content types, otherwise null
  - Allows access to the DOM of the XML document

Example

Create a standard HTML form with two text fields: username and time.
The username field will be filled in by the user and the time field will be filled in using AJAX.
No submit button is needed.

Example

```
<html>
<body> <form name="myForm">
Name: <input type="text" name="username" />
Time: <input type="text" name="time" />
</form> </body>
</html>
```

Creating an XMLHttpRequest object

```
<script type="text/javascript">
function ajaxFunction()
{
  var xmlHttp;
  xmlHttp=new XMLHttpRequest();

  ...
}
</script>
```

Supporting all browsers

```
<script type="text/javascript">
function ajaxFunction()
{
var xmlHttp;
try {
  // Firefox, Opera 8.0+, Safari
  xmlHttp=new XMLHttpRequest();
  }
catch (e) {
  // Internet Explorer
  try { // Internet Explorer 6.0+
    xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    }
  catch (e) {
    try { // Internet Explorer 5.5+
      xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
      }
    catch (e) {
      alert("Your browser does not support AJAX!");
      return false;
      }
    }
  }
}
</script>
```

Calling the server

```
xmlHttp.open("GET","time.jsp",true);
xmlHttp.send(null);
```

Processing the response

```
xmlHttp.onreadystatechange=function()
{
if(xmlHttp.readyState==4)
  {
  // Get the data from the server's response
  document.myForm.time.value=xmlHttp.responseText;
  }
}
```

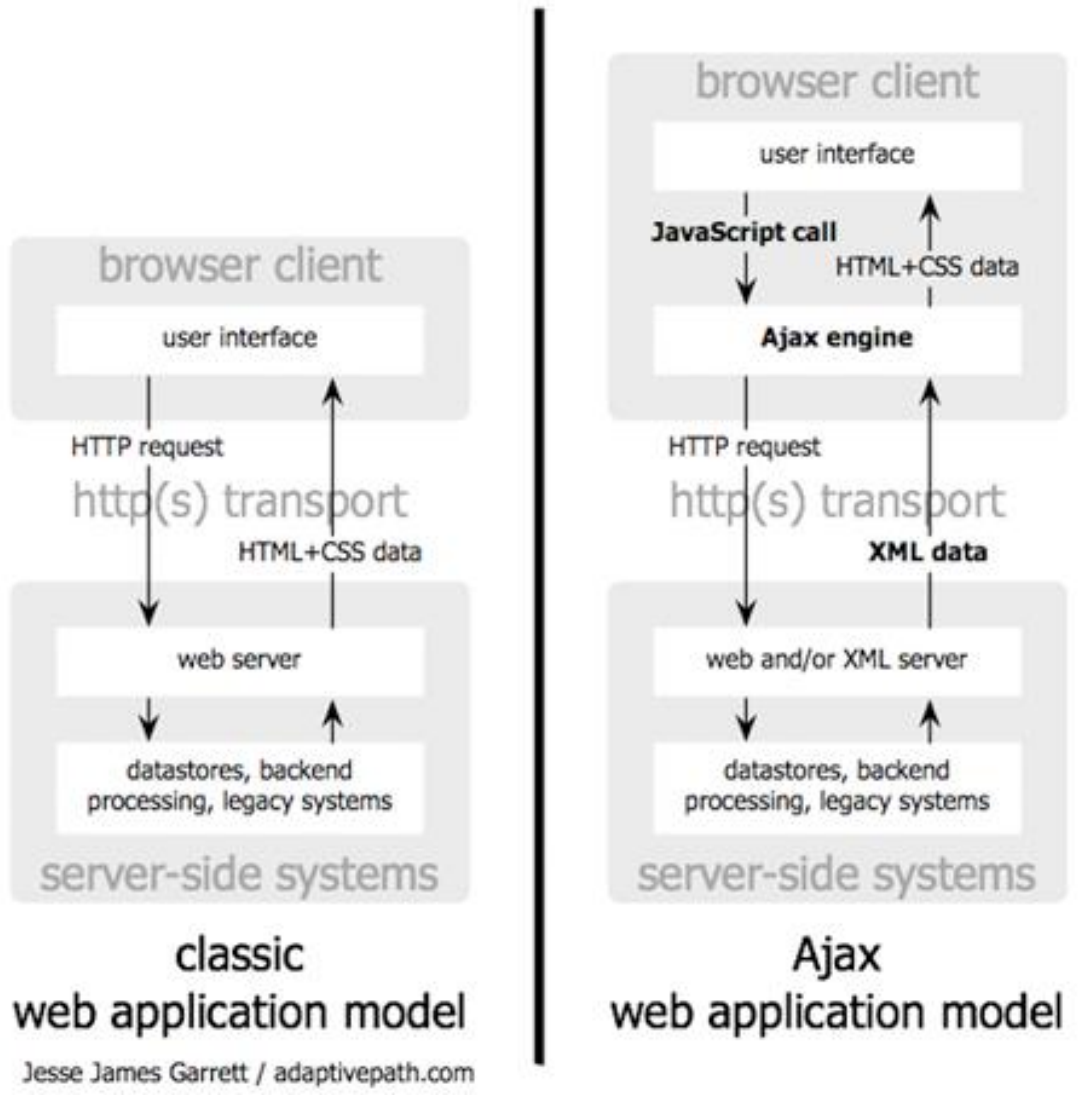
Attaching to an event

```
<form name="myForm">
Name: <input type="text"
onkeyup="ajaxFunction();" name="username" />
Time: <input type="text" name="time" />
</form>
```

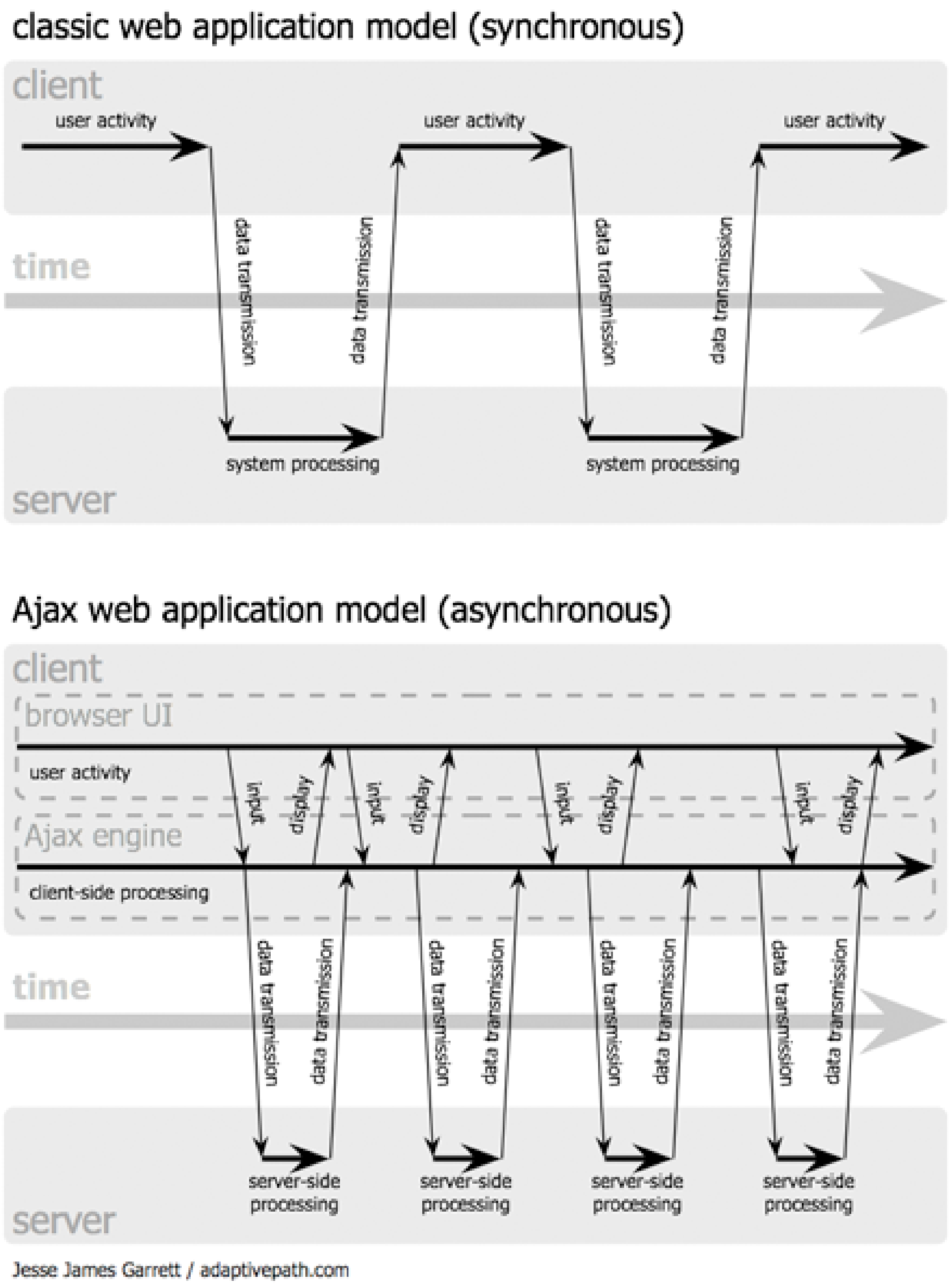
Complete example

```
<html>
<body>
<script type="text/javascript">
function ajaxFunction()
{
  var xmlHttp=new XMLHttpRequest();


  xmlHttp.onreadystatechange=function()
    {
    if(xmlHttp.readyState==4)
      {
      document.myForm.time.value=xmlHttp.responseText;
      }
    }


  xmlHttp.open("GET","time.asp",true);
  xmlHttp.send(null);
  }
</script>
<form name="myForm">
Name: <input type="text"
onkeyup="ajaxFunction();" name="username" />
Time: <input type="text" name="time" />
</form> </body>
```

AJAX architecture



AJAX behavior



Exercise 1

Create an auto-complete feature for entering the name in a FORM

For every typed letter, an associated text must be updated, reflecting the list of all possible names with those initial(s)

Once submitted, the name adds up to the list

Clicking on the suggestion auto-fills the box

Name Jo

Suggestions: Joe, Joseph, John

SUBMIT

Exercise 2

Create a FORM for entering the name of a city, based on two drop-down menus (<select> tags).

- The first <select> contains the list of all *provinces* (AO, BO, CN, MI, TO, ...)
- The second <select> contains the list of all *cities* in the province

Every time the user changes the province, then the list of cities MUST be updated

The form may be submitted only if information is complete

References

http://en.wikipedia.org/wiki/Rich_Internet_Applications
http://en.wikipedia.org/wiki/AJAX
http://www.w3schools.com/ajax/
http://www.w3.org/TR/XMLHttpRequest/

License