



POLITECNICO
DI TORINO



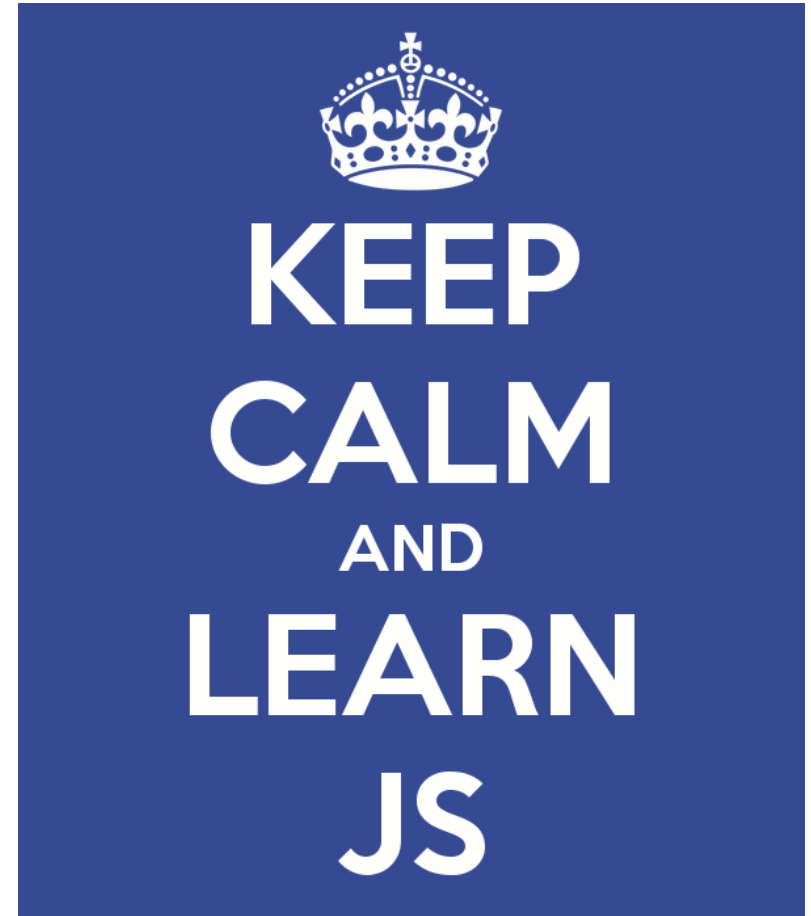
e-Lite

Introduction to JavaScript

Ambient intelligence

Fulvio Corno

Politecnico di Torino, 2018/2019



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

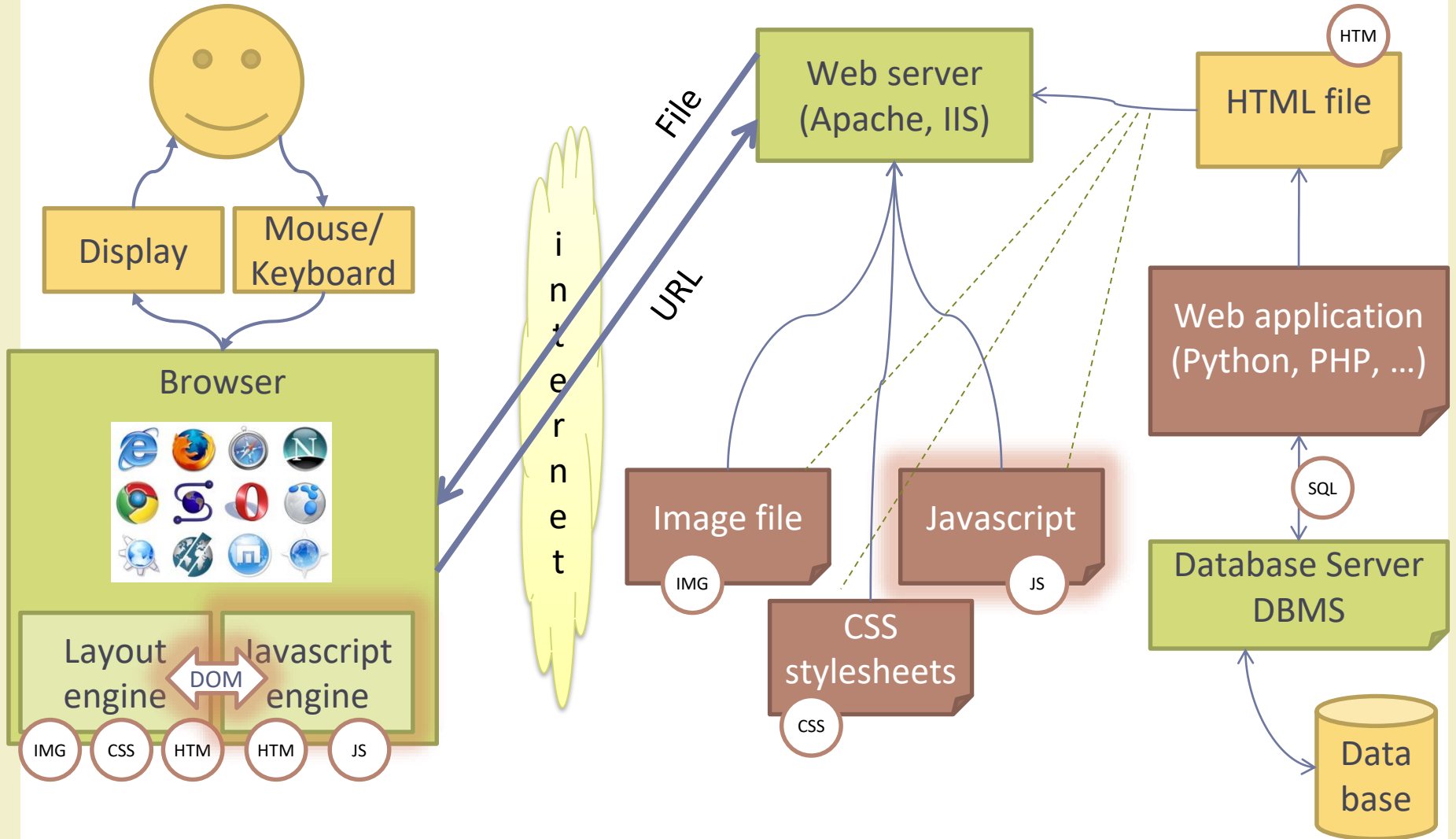
Client-side programming

- 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

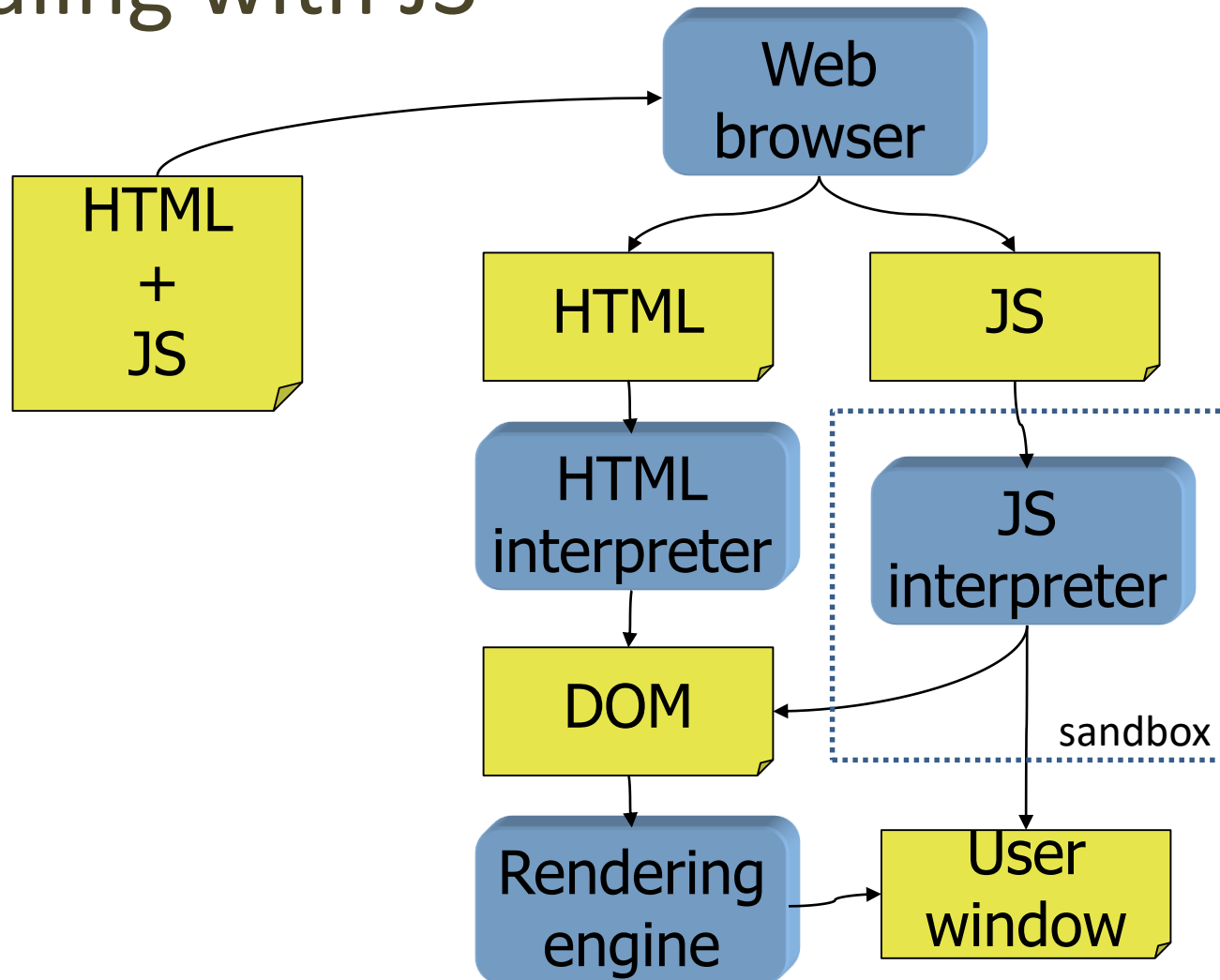
The solution

- 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

General Architecture



Dealing with JS



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

Embedding JS in HTML

Embedded `<script>` element

```
<script>  
<!--  
  
    [JavaScript code here]  
  
// -->  
</script>
```

External JS file

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

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. Better to include them **at the very bottom** of the body (so that other content is already loaded).

What 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

https://www.w3schools.com/js/default.asp

w3schools.com THE WORLD'S LARGEST WEB DEVELOPER SITE

HTML CSS **JAVASCRIPT** SQL PHP BOOTSTRAP JQUERY ANGULAR MORE ▾ REFERENCES ▾ EXAMPLES ▾

JS Tutorial

- JS HOME**
- JS Introduction
- JS Where To
- JS Output
- JS Syntax
- JS Statements
- JS Comments
- JS Variables
- JS Operators
- JS Arithmetic
- JS Assignment
- JS Data Types
- JS Functions
- JS Objects
- JS Scope
- JS Events
- JS Strings
- JS String Methods
- JS Numbers
- JS Number Methods
- JS Math
- JS Random
- JS Dates
- JS Date Formats
- JS Date Methods
- JS Arrays
- JS Array Methods
- JS Array Sort
- JS Booleans
- JS Comparisons
- JS Conditions
- JS Switch

JavaScript Tutorial

[< Home](#) [Next >](#)

JavaScript is the programming language of HTML and the Web.

JavaScript is easy to learn.

This tutorial will teach you JavaScript from basic to advanced.

Examples in Each Chapter

With our "Try it Yourself" editor, you can change all examples and view the results.

Example

My First JavaScript

[Click me to display Date and Time](#)

[Try it Yourself >](#)

We recommend reading this tutorial, in the sequence listed in the left menu.

COLOR PICKER

LEARN MORE

- Tabs
- Dropdowns
- Accordions
- Convert Weights
- Animated Buttons
- Side Navigation
- Top Navigation
- JS Animations
- Modal Boxes
- Progress Bars
- Parallax
- Login Form
- HTML Includes
- Google Maps
- Loaders
- Tooltips
- Slideshow
- Filter List
- Sort List

SHARE

[f](#) [t](#) [G+](#)

[👍](#)

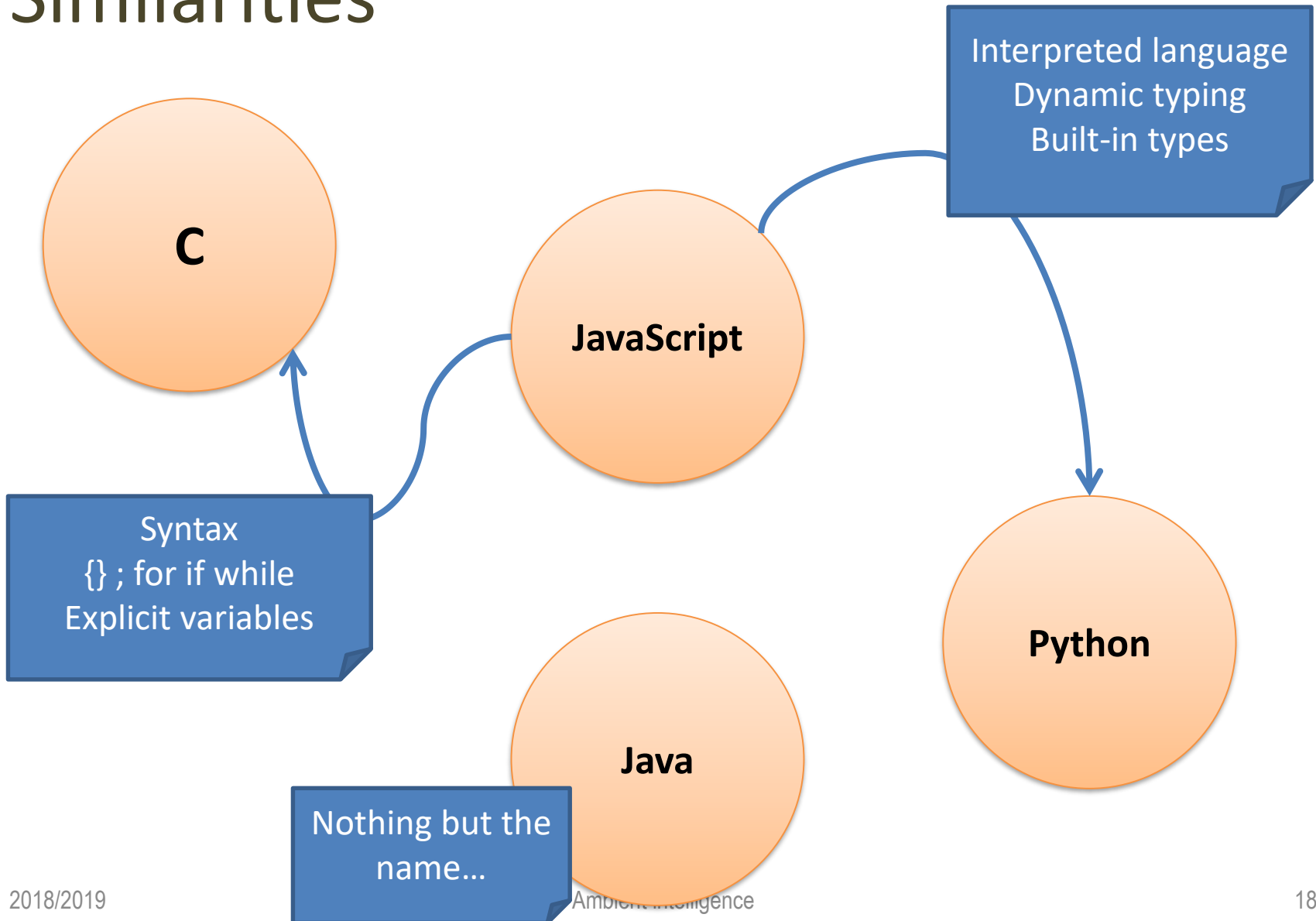
CERTIFICATES

The screenshot shows the MDN web docs page for JavaScript. At the top left is the MDN logo with the text "MDN web docs" and "mozilla". To its right are navigation links: "Technologies", "References & Guides", and "Feedback". On the top right, there is a "Sign in" link and a search icon. Below the navigation is a header section with the title "JavaScript" in large, bold letters. To the right of the title are three buttons: "Languages", "Edit", and a settings gear icon. Below the header is a main content area. On the left side of this area is a sidebar with a "Web technology for developers" breadcrumb and "JavaScript" link. Below that are sections for "Related Topics" (with a link to "JavaScript"), "Tutorials:" (with a list of links: "Complete beginners", "JavaScript Guide", "Intermediate", "Advanced"), and "References:" (with a list of links: "Built-in objects", "Expressions & operators", "Statements & declarations", "Functions", "Classes", "Errors", "Misc", "New in JavaScript"). The main content area contains a paragraph defining JavaScript (JS) as a lightweight interpreted or just-in-time compiled programming language with first-class functions. It mentions that many non-browser environments use it, such as Node.js, Apache CouchDB, and Adobe Acrobat. It states that JavaScript is a prototype-based, multi-paradigm, dynamic language supporting object-oriented, imperative, and declarative styles. Below this is a section titled "This section is dedicated to the JavaScript language itself..." followed by a paragraph about the standard for JavaScript being ECMAScript, mentioning the 2015 update to ES6. At the bottom of the main content area are two sections: "Tutorials" and "Reference", each with a link icon. The "Tutorials" section has a sub-link "Learn how to program in JavaScript with guides and tutorials". The "Reference" section has a sub-link "Browse the complete JavaScript reference documentation".

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

Similarities



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 refer to 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)

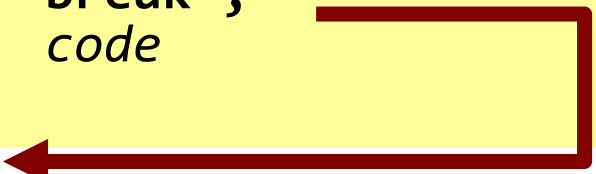
```
for ( v = startvalue;  
      v < endvalue;  
      v = v+increment )  
{  
    code to be executed  
}
```

```
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
}
```

Name

Function body

List of function arguments
(passed 'by value')

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

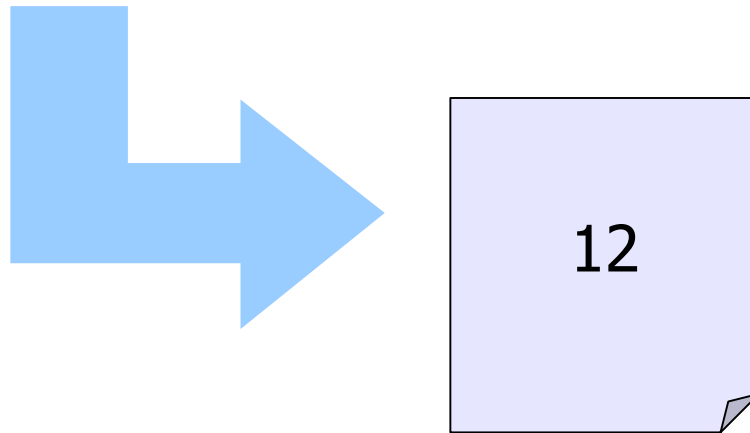
- Creating new objects
 - `var d = new Date()`
 - Create a new Object of type Date, and use the variable d as a reference to that object
- Properties and methods
 - `var day = d.getDay() ;`
 - `d.setMinutes(34) ;`

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)
```



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()`

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 constants

- `Math.E`
- `Math.PI`
- `Math.SQRT2` // $\sqrt{2}$
- `Math.SQRT1_2` // $\sqrt{1/2}$
- `Math.LN2` // $\log_e(2)$
- `Math.LN10` // $\log_e(10)$
- `Math.LOG2E` // $\log_2(e)$
- `Math.LOG10E` // $\log_{10}(e)$

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

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

Full list: <https://developer.mozilla.org/en-US/docs/Web/Events>

Example

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

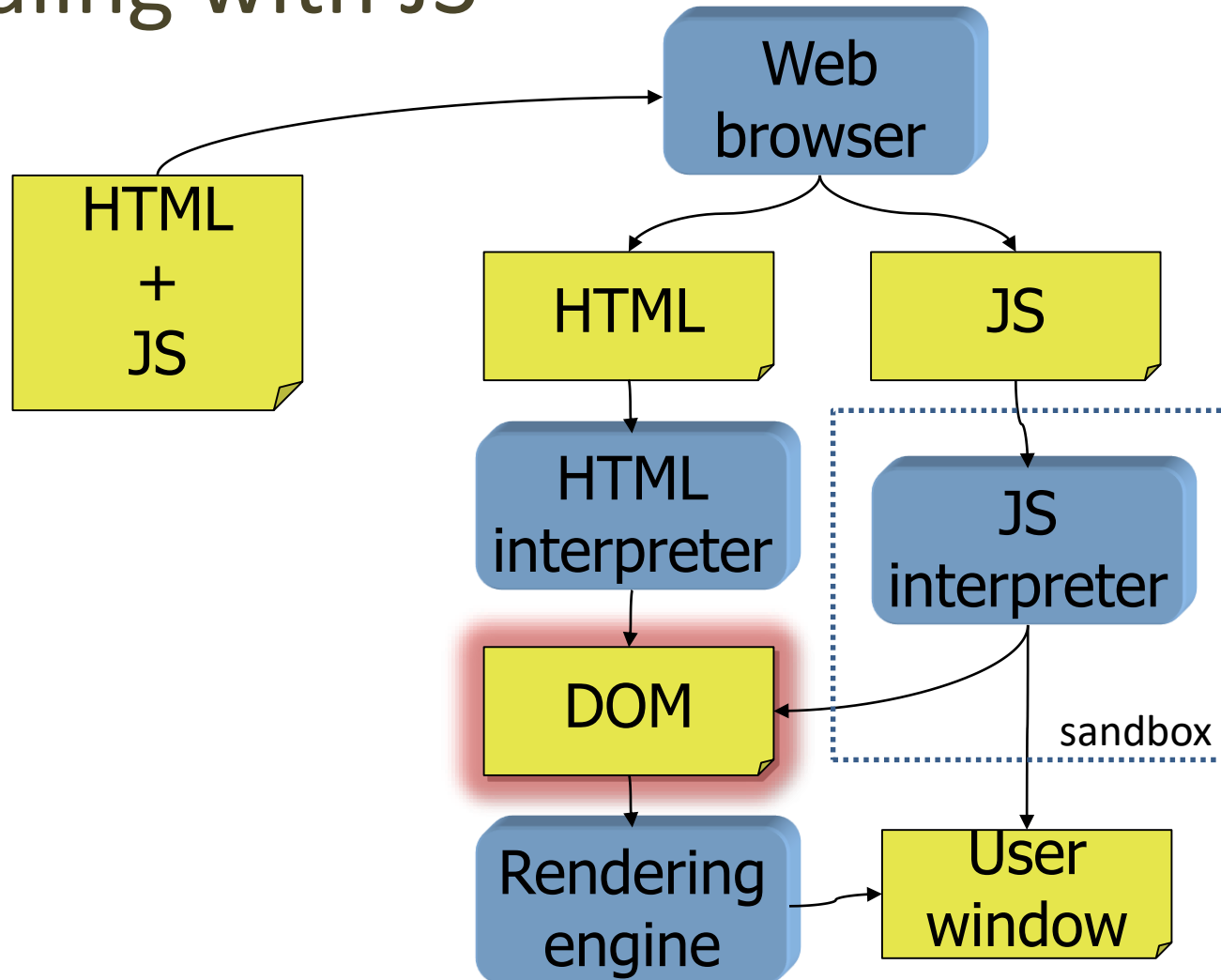
  <body>
    <form>
      <input type="button" onclick="sayHello()"
        value="Press me">
    </form>
  </body>
</html>
```




HTML Document Object Model (DOM)

HTML DOCUMENT OBJECT MODEL (DOM)

Dealing with JS



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.

Document Object Model (DOM)

Languages

Edit



Jump to: [DOM interfaces](#) [HTML DOM](#) [SVG interfaces](#) [Specifications](#) [See also](#)

Web technology for developers > [Web APIs](#) >
[Document Object Model \(DOM\)](#)

Related Topics

Document Object Model

▼ Guides

[Events and the DOM](#)

[Examples of web and XML development using the DOM](#)

[How to create a DOM tree](#)

[Introduction to the DOM](#)

[Locating DOM elements using selectors](#)

[Traversing an HTML table with JavaScript and DOM Interfaces](#)

[Using the W3C DOM Level 1 Core](#)

[Whitespace in the DOM](#)

▼ Interfaces

[AbortController](#)

The **Document Object Model (DOM)** connects web pages to scripts or programming languages by representing the structure of a document—such as the HTML representing a web page—in memory. Usually that means JavaScript, although modelling HTML, SVG, or XML documents as objects is not part of the JavaScript language, as such.

The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them you can change the document's structure, style, or content. Nodes can also have event handlers attached to them; once an event is triggered, the event handlers get executed.

To learn more about what the DOM is and how it represents documents, see our article [introduction to the DOM](#).

DOM interfaces [↗](#)

[Attr](#)

[CDATASection](#)

[CharacterData](#)

[childNodes](#)

[DOMStringList](#)

[DOMTokenList](#)

[Element](#)

[Event](#)

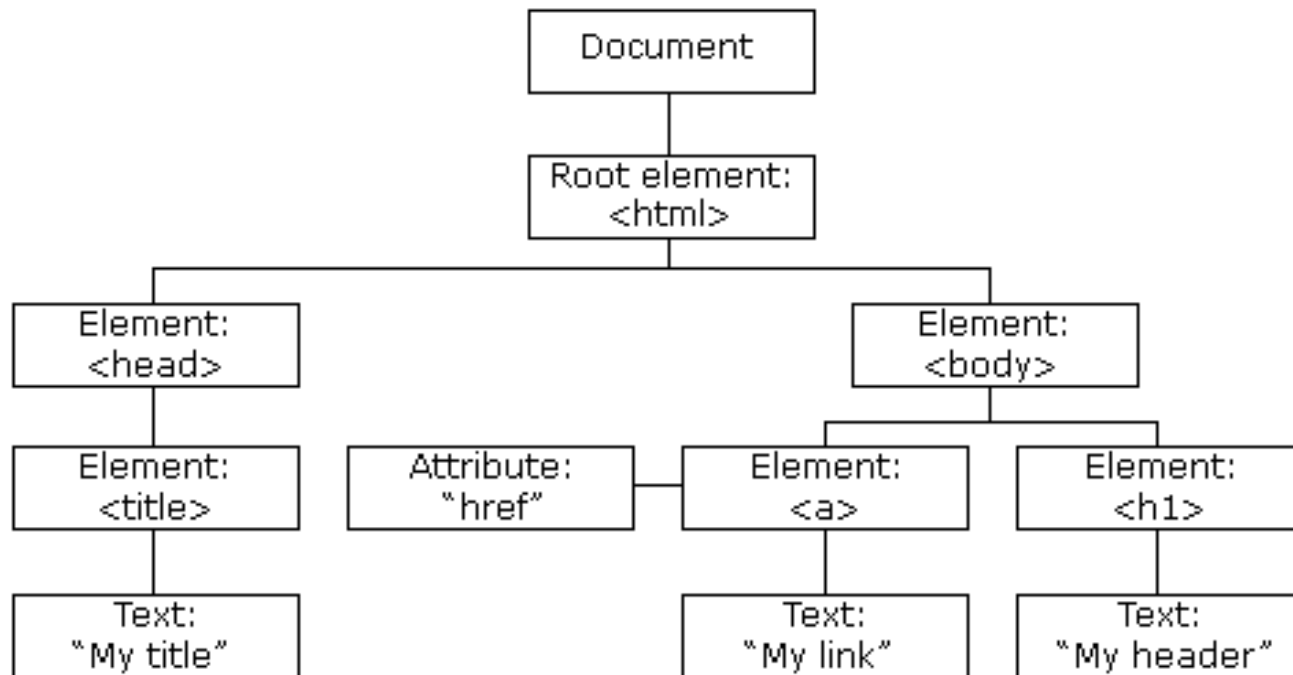
[ParentNode](#)

[ProcessingInstruction](#)

[Selection](#)

[Range](#)

DOM example



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

Fundamental Data Types

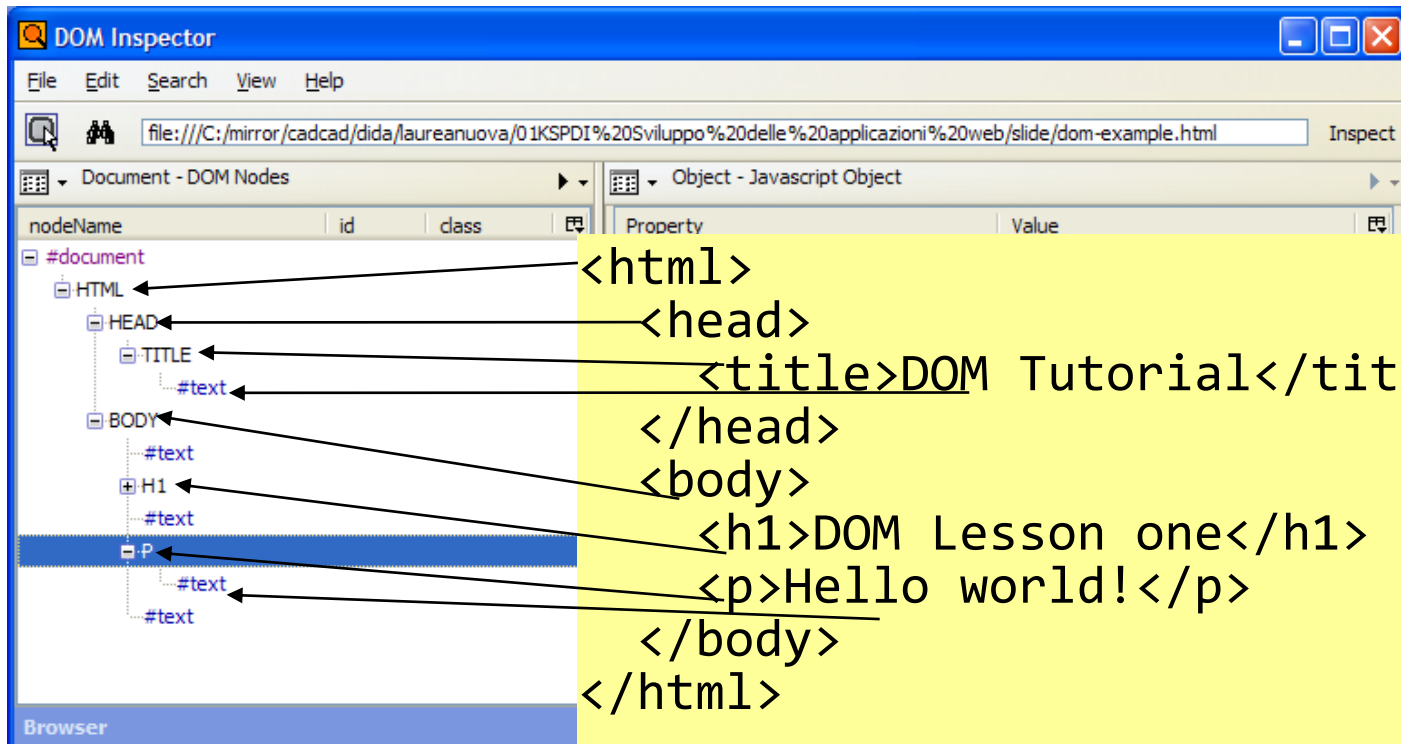
Data type (Interface)	Description
<code>Document</code>	When a member returns an object of type <code>document</code> (e.g., the <code>ownerDocument</code> property of an element returns the <code>document</code> to which it belongs), this object is the root <code>document</code> object itself. The DOM <code>document</code> Reference chapter describes the <code>document</code> object.
<code>Node</code>	Every object located within a document is a node of some kind. In an HTML document, each node is an element.
<code>Element</code>	The <code>element</code> type is based on <code>node</code> . It refers to an element or a node of type <code>element</code> returned by a member of the DOM API. Rather than saying, for example, that the <code>document.createElement()</code> method returns an object reference to a <code>node</code> , we just say that this method returns the <code>element</code> that has just been created in the DOM. <code>element</code> objects implement the DOM <code>Element</code> interface and also the more basic <code>Node</code> interface, both of which are included together in this reference. In an HTML document, elements are further enhanced by the HTML DOM API's <code>HTMLElement</code> interface as well as other interfaces describing capabilities of specific element kinds of elements (for instance, <code>HTMLTableElement</code> for <code><table></code> elements).
<code>NodeList</code>	<p>A <code>nodeList</code> is an array of elements, like the kind that is returned by the method <code>document.getElementsByTagName()</code>. Items in a <code>nodeList</code> are accessed by index in either of two ways:</p> <ul style="list-style-type: none">• <code>list.item(1)</code>• <code>list[1]</code> <p>These two are equivalent. In the first, <code>item()</code> is the single method on the <code>nodeList</code> object. The latter uses the typical array syntax to fetch the second item in the list.</p>
<code>Attribute</code>	When an <code>attribute</code> is returned by a member (e.g., by the <code>createAttribute()</code> method), it is an object reference that exposes a special (albeit small) interface for attributes. Attributes are nodes in the DOM just like elements are, though you may rarely use them as such.
<code>NamedNodeMap</code>	A <code>namedNodeMap</code> is like an array, but the items are accessed by name or index, though this latter case is merely a convenience for enumeration, as they are in no particular order in the list. A <code>namedNodeMap</code> has an <code>item()</code> method for this purpose, and you can also add and remove items from a <code>namedNodeMap</code> .

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
Ambient intelligence

Example

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


Example



The screenshot shows the DOM Inspector window with a tree view on the left and a property table on the right. The tree view shows the following structure:

- #document
 - HTML
 - HEAD
 - TITLE
 - #text
 - BODY
 - #text
 - H1
 - #text
 - P (highlighted)
 - #text
 - #text

The property table on the right shows the following structure:

Property	Value
Property	Value

Arrows point from the DOM tree nodes to the corresponding HTML code in the yellow box:

- #document → `<html>`
- HTML → `<head>`
- HEAD → `<title>DOM Tutorial</title>`
- TITLE → `</head>`
- BODY → `<body>`
- #text (under BODY) → `<h1>DOM Lesson one</h1>`
- H1 → `<p>Hello world!</p>`
- P → `</body>`
- #text (under P) → `</html>`

DOM Lesson one

Hello world!

2018/2019

Ambient intelligence

73

Example

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

The screenshot shows the DOM Inspector tool. On the left, the DOM tree is displayed with the following structure:

- #document
 - HTML
 - HEAD
 - TITLE
 - #text
 - BODY
 - #text
 - H1
 - #text
 - P (selected)
 - #text

On the right, the JavaScript Object console shows the properties of the selected P element:

Property	Value
Subject	[object HTMLParagraphElement]
addEventListener	function addEventListener() { [native code] }
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!

2018/2019

Ambient intelligence

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>

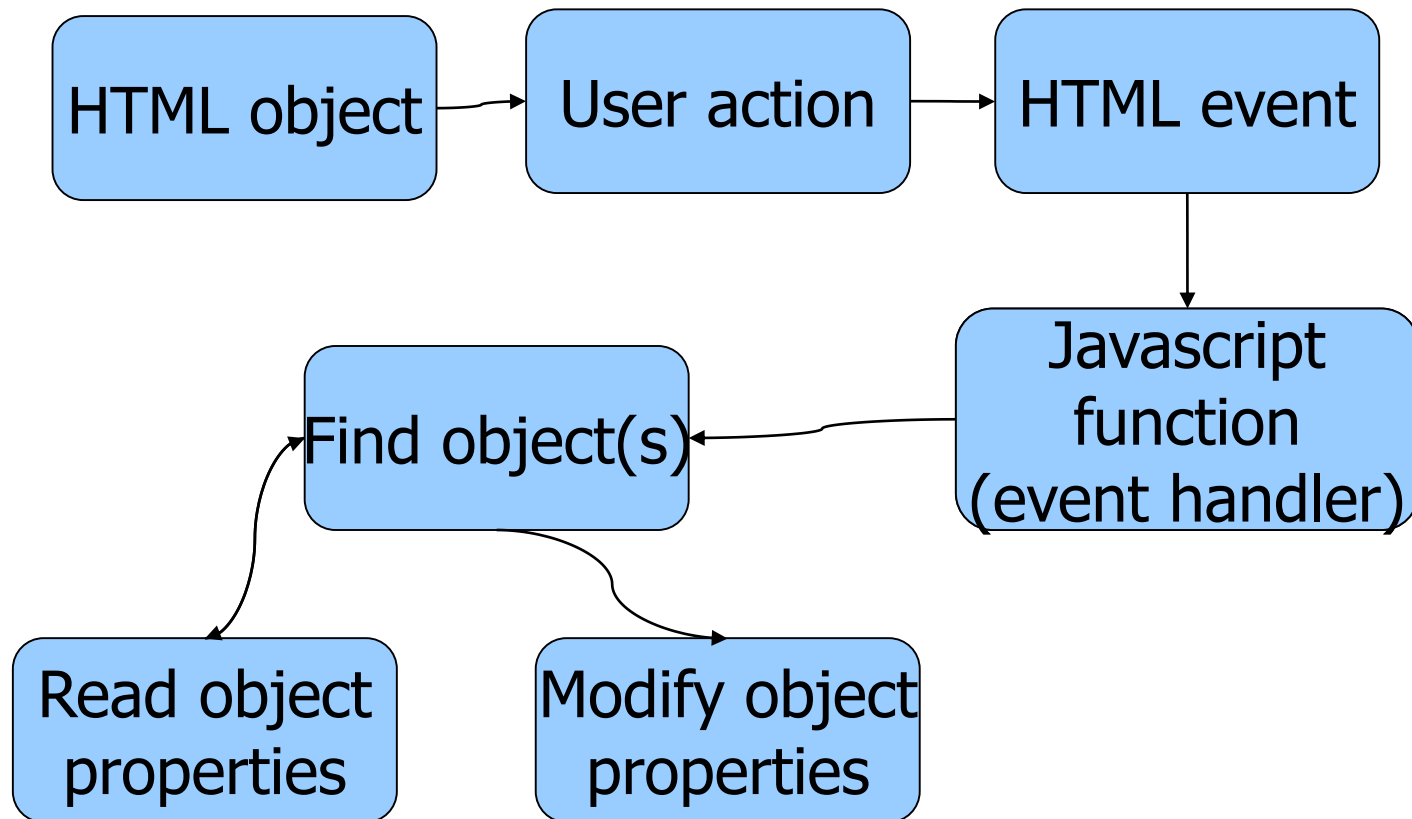
    <script>...</script>

  </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



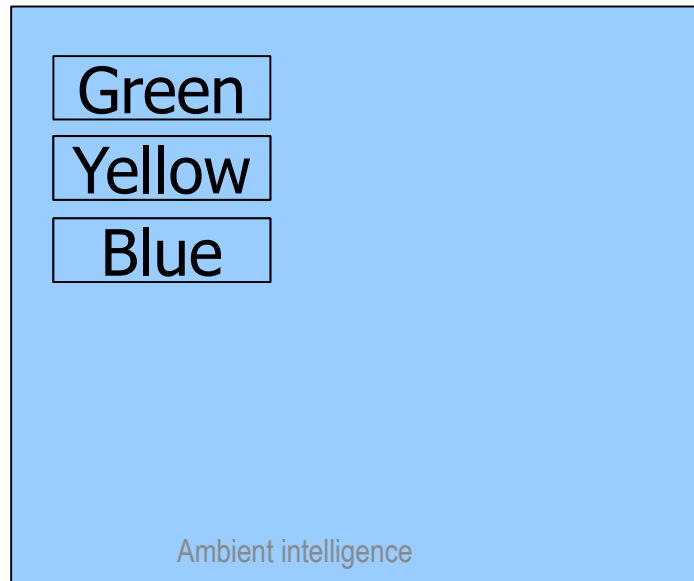
HTML events

<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

Full list: <https://developer.mozilla.org/en-US/docs/Web/Events>

Exercise

- 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

- 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

- 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

- 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 (3rd Edition - December 1999),
<http://www.ecma-international.org/publications/standards/Ecma-262.htm>



Asynchronous JavaScript programming

AJAX

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

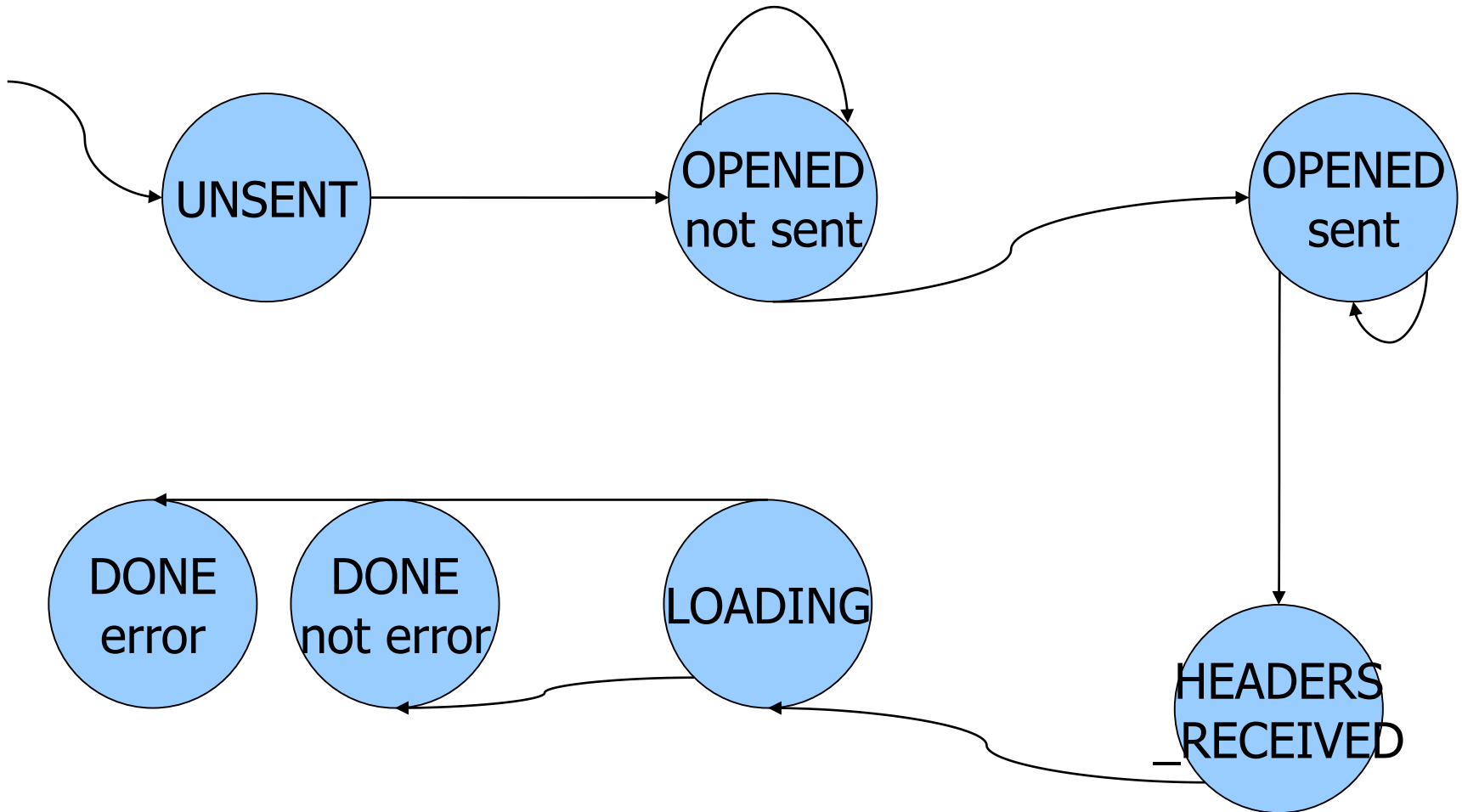
LOADING = 3

The request is in process

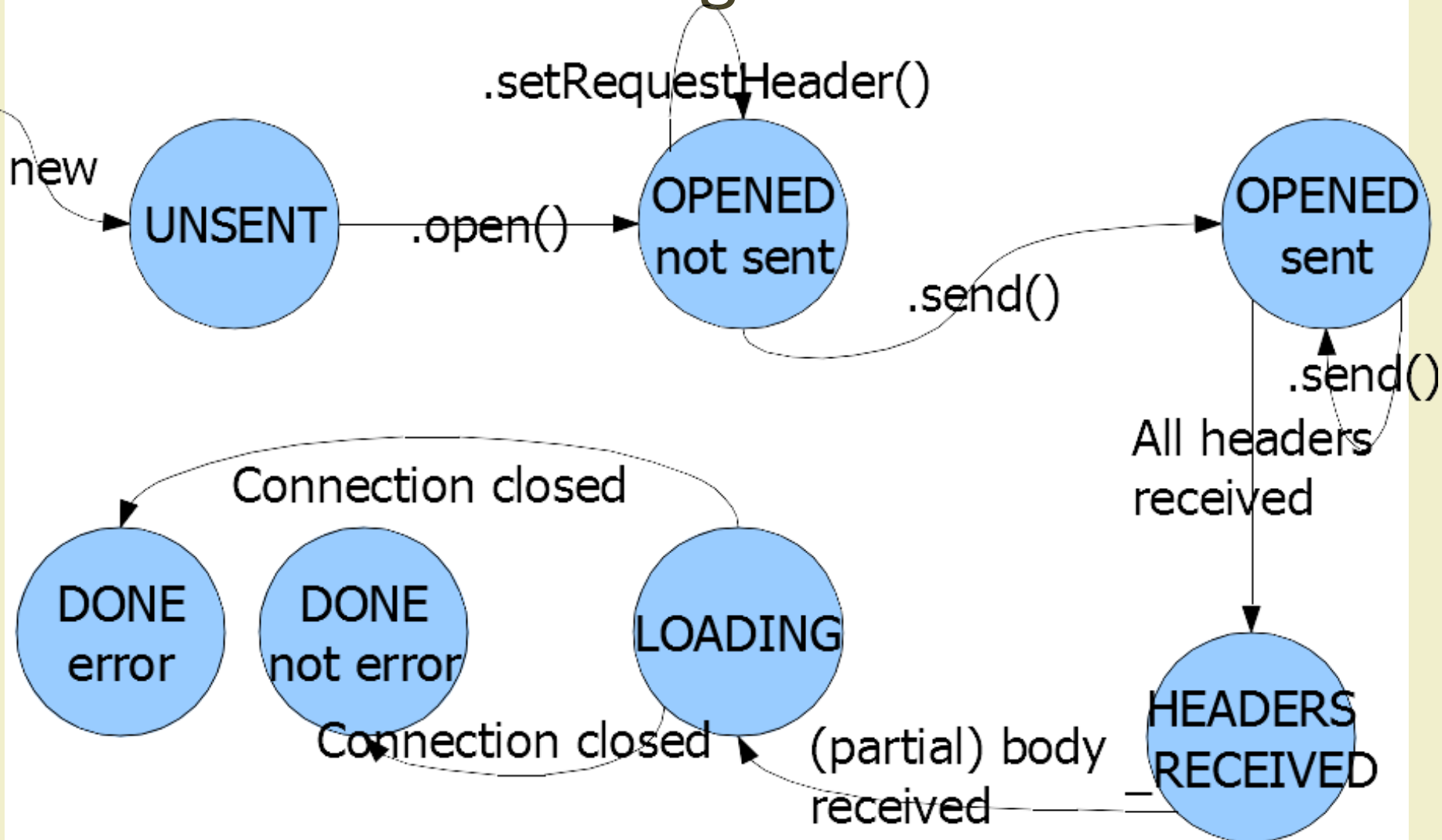
DONE = 4

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;  
        }  
    }  
}  
}  
2018/2019  
</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>
```

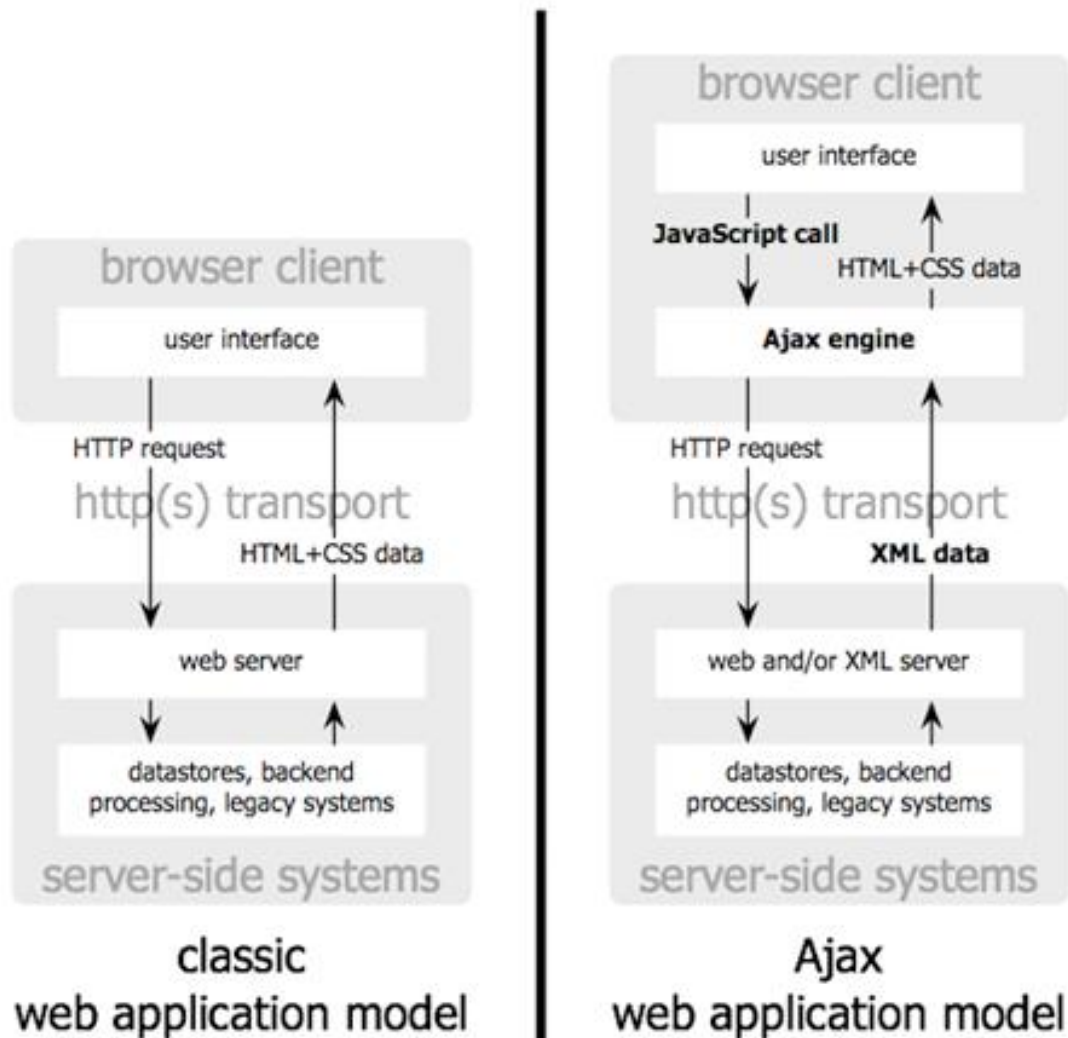
Complete example

```
<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;
    }
  }

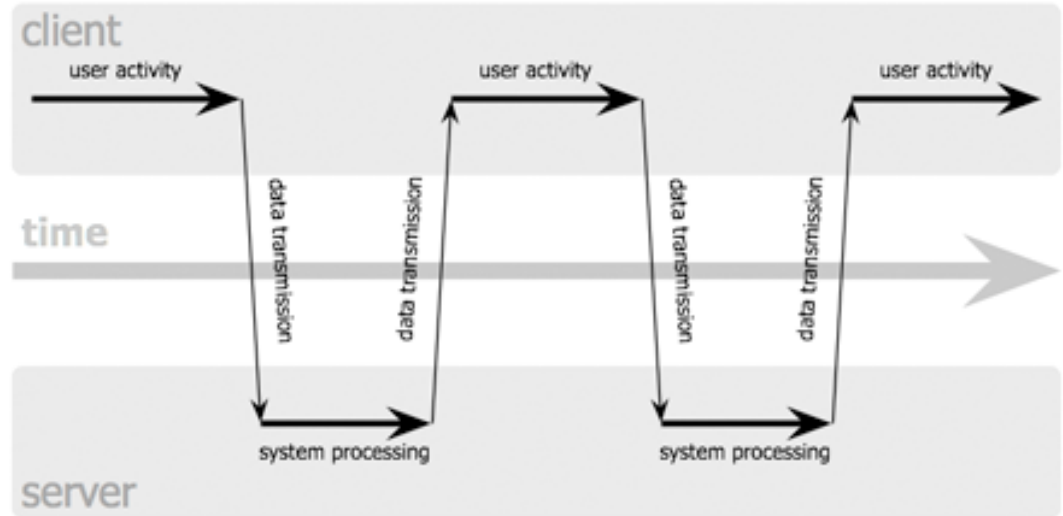
  xmlhttp.open("GET","time.asp",true);
  xmlhttp.send(null);
}
</script>
<form name="myForm">
Name: <input type="text"
onkeyup="ajaxFunction();" name="username" />
Time: <input type="text" name="time" />
</form> </body>
```

AJAX architecture

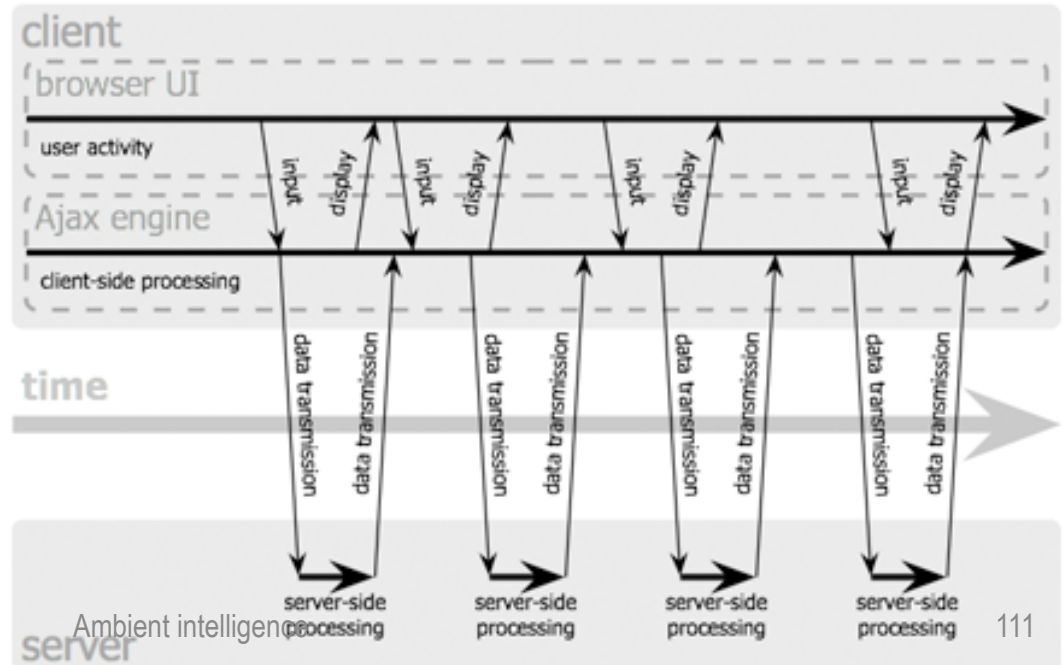


AJAX behavior

classic web application model (synchronous)



Ajax web application model (asynchronous)



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

Suggestions: [Joe](#), [Joseph](#), [John](#)

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



- These slides are distributed under a Creative Commons license “Attribution – NonCommercial – ShareAlike (CC BY-NC-SA) 3.0”
- You are free to:
 - Share — copy and redistribute the material in any medium or format
 - Adapt — remix, transform, and build upon the material
 - The licensor cannot revoke these freedoms as long as you follow the license terms
- Under the following terms:
 - Attribution — You must give [appropriate credit](#), provide a link to the license, and [indicate if changes were made](#). You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
 - NonCommercial — You may not use the material for [commercial purposes](#).
 - ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the [same license](#) as the original.
 - No additional restrictions — You may not apply legal terms or [technological measures](#) that legally restrict others from doing anything the license permits.
- <http://creativecommons.org/licenses/by-nc-sa/3.0/>

