Prof. Dr. Claudia Müller-Birn
Institute for Computer Science, Networked Information Systems

Freie Universität Berlin

Web application development (part 2)

01-17-2012
Netzprogrammierung
(Algorithmen und Programmierung V)

Our topics today

JavaScript … 5

- Some basics
- The Document Object Model (DOM)
- JavaScript Frameworks (e.g., jQuery, Processing.js)

Asynchronous Java and XML (AJAX) … 31

- Synchronous vs. asynchronous interaction pattern
- Web application request flow vs. AJAX request flow
- Core AJAX concepts (esp. XMLHttpRequest)
- Discussion of a simple AJAX example

Web sockets … 47

Web data and browser as platform

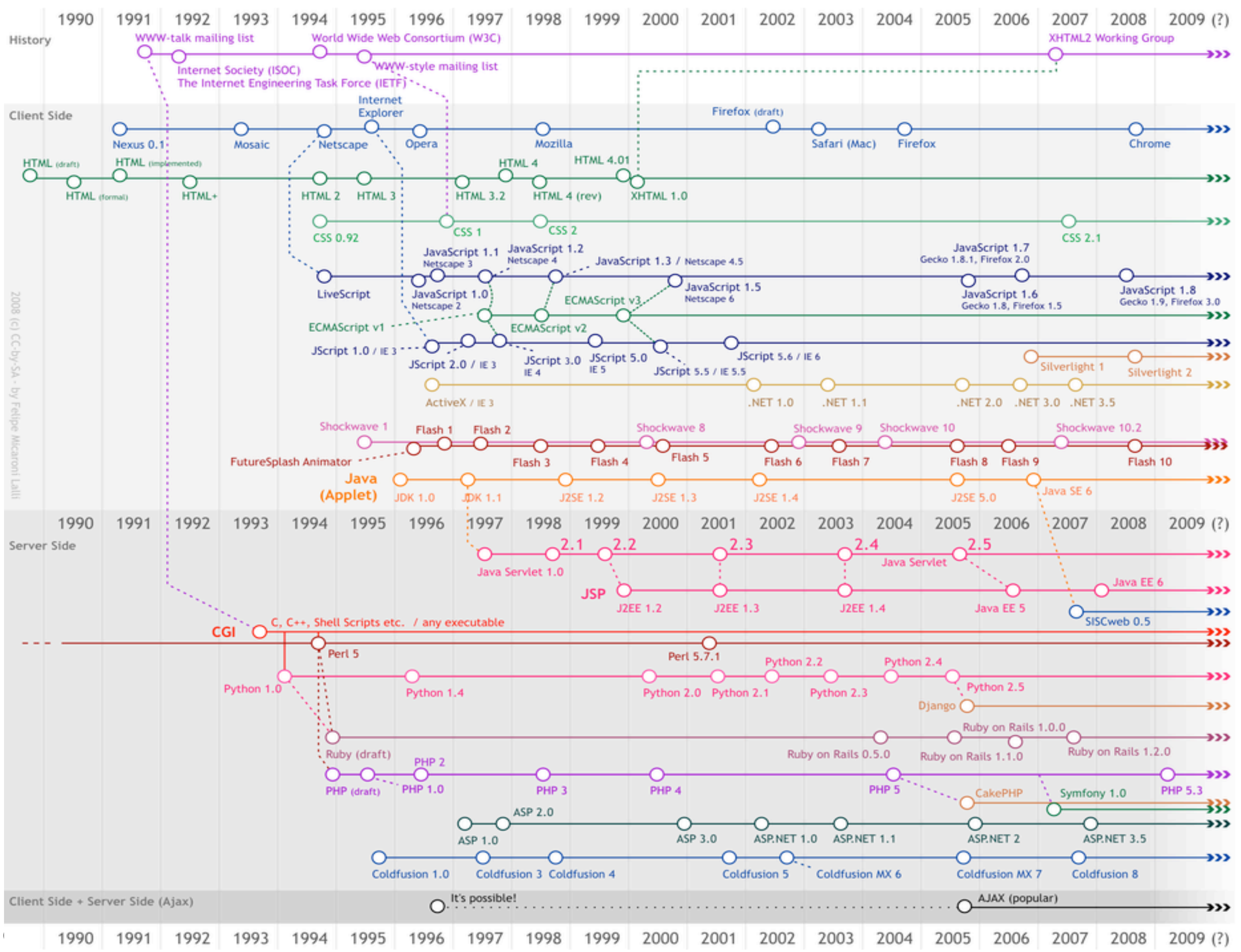
Most interesting web pages (e.g., Google, IMDB, Digg, Facebook, YouTube, Rotten Tomatoes) revolve around data. They include many formats: text, HTML, XML, multimedia.

How can we connect to web applications that serve data?

Runtime environments are critical for running applications. Popular computer environments are Windows, MacOS, Linux, and Java. Popular Web-based environments are JavaScript, Flash, and Java applets or Web-oriented environments are Silverlight and AIR.

JavaScript is a scripting language supported by most browsers. It accesses to the current document's DOM is the most important part of DHTML. JavaScript has code, functions, and interacts with the user through the browser.

Modern implementations allow sophisticated desktop-like applications, such as Google Docs.



http://en.wikipedia.org/wiki/File:Web_development_timeline.png

Freie Universität Berlin

Web application development (part 2)

JavaScript

What is JavaScript?

JavaScript is an implementation of the ECMAScript language standard. ECMA-262 is the official JavaScript standard. (http://www.ecma-international.org/publications/standards/Ecma-262.htm)

JavaScript was invented by *Brendan Eich* at Netscape (with Navigator 2.0), and has appeared in all browsers since 1996.

The official standardization was adopted by the ECMA organization (an industry standardization association) in 1997.

The ECMA standard was approved as an international ISO (ISO/IEC 16262) standard in 1998.

The development is still in progress.

Side note: JavaScript is not based on or related to Java.

What can JavaScript do?

JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages

JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element

JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer

Lexical structure

- JavaScript programs are written using the Unicode character set. It is a case-sensitive language.
- In JavaScript, you can usually omit the semicolon between two statements if those statements are written on separate lines.
- JavaScript does not treat every line break as a semicolon: it usually treats line breaks as semicolons only if it can't parse the code without the semicolons.

Examples

```
var a
a
=
3
console.log(a)
```

```
var y = x + f
(a+b).toString()
```

```
return
true;
```

```
x
++
y
```

Interpretation

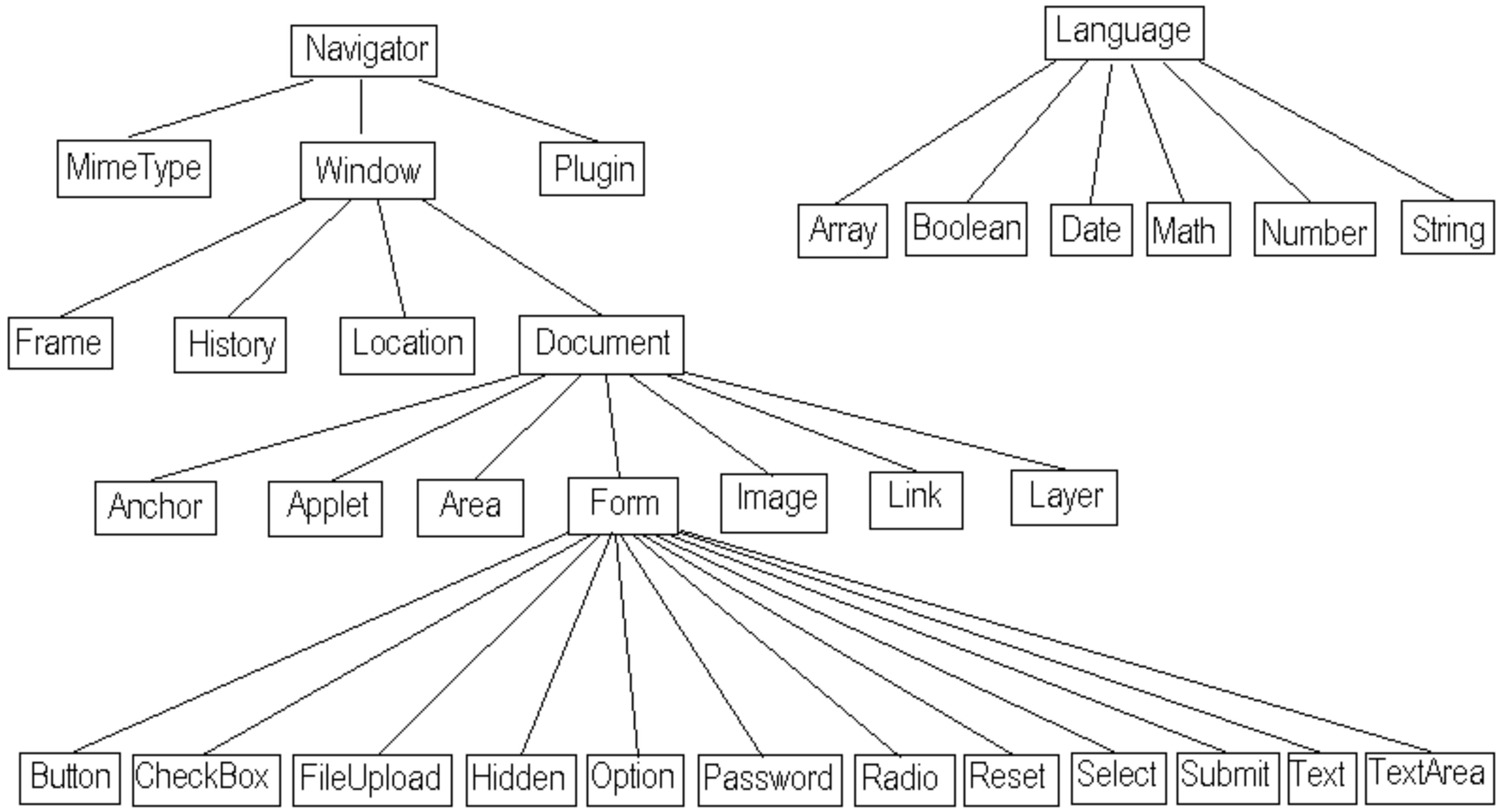
```
var a; a = 3;
console.log(a);
```

```
var y = x + f(a
+b).toString();
```

```
return; true;
```

```
x; ++y;
```

JavaScript object hierarchy



http://www.comptechdoc.org/independent/web/cgi/javamanual/javaobjheir.html

JavaScript basics

Tutorial at
http://www.w3schools.com/js/

Declaration

- Explicit: var i = 12; // no 'var' in declaration
- Implicit: i = 12;

Variable scope

- Global: Declared outside functions and any variable that is *implicitly* defined
- Local: *Explicit* declarations inside functions

Control and looping

- 'if', 'switch' statement
- while, and do-while loops (break and continue keywords)

Not object-oriented but object-based

```
function Student(studentName, studentAge) {
            this.name = studentName;
            this.age = studentAge; }

var someStudent = new Student("Michael", 21);
```

JavaScript output

The document objects allow printing directly into the browser page (amongst other things)

The window object is implied.

Example: Writing in text or HTML with script or without line-break

```
document.write(“I am <B>BOLD</B>”);
document.writeln(“I am <U>underlined</U>”);
```

What is the window object?

- It is part of the Browser Object Model (BOM)
- BOM is a a collection of objects that represent the browser and the computer screen
- Objects are accessible through the global objects window and window.screen
- The window object is global to the extent that it represents the very host environment of all other JavaScript objects

Browser Window Object

window object is a JavaScript representation of a browser window

- closed - A boolean value that indicates whether the window is closed
- defaultStatus - This is the default message that is loaded into the status bar when the window loads
- Example: window.defaultStatus = “This is the status bar”;

Selected *build in functions*

alert("message”) - string passed to the alert function is displayed in an alert dialog box.

window.close() - function will close the current window or the named window.

confirm("message") - string passed to the confirm function is displayed in the confirm dialog box.

focus() - function will give the focus to the window.

open("URLname","Windowname",["options"]) - new window is opened with the name specified by the second parameter.

Form object

The Form object is a property of the document object. This corresponds to an HTML input form constructed with the FORM tag. A form can be submitted by calling the JavaScript submit method or clicking the form submit button.

Form objects can be accessed by

window.document.myForm OR window.document.forms[0]

Selected properties

action - This specifies the URL and CGI script file name the form is to be submitted to. It allows reading or changing the ACTION attribute of the HTML FORM tag.

target - The name of the frame or window the form submission response is sent to by the server. Corresponds to the FORM TARGET attribute.

length - The number of fields in the elements array, i.e. the length of the elements array.

method - This is a read or write string. It has the value "GET" or "POST".

Freie Universität Berlin

Form object – objects and methods

Form *Objects (one example)*

- text - A GUI text field object. Methods are blur(), focus(), and select()
- Attributes

  defaultValue - The text default value of the text field

  name - The name of the text field

  type - Type is "text”

  value - The text that is entered and appears in the text field. It is sent to the server when the form is submitted
- Example “accessing form field values”: window.document.myForm.firstname.value

Form object methods

- reset() - Used to reset the form elements to their default values
- Example window.document.myForm.reset();
- submit() - Submits the form as though the submit button were pressed by the user
- Example window.document.myForm.submit();

Event handlers

Events are actions that occur usually as a result of something the user does. For example, clicking a button is an event, as is changing a text field or moving the mouse over a hyperlink.

Other events are click, change, focus, load, mouseover, mouseout, reset, submit, select …

You can use event handlers, such as onChange and onClick, to make your script react to events.

Examples

```
<input type="button" onClick="javascript:doButton()>
<select onChange="javascript:doChange()">
<a onClick="javascript:doSomething()"> </a>
<form onSubmit="javascript:validate()">
```

Freie Universität Berlin

JavaScript principle

Contents	Behavior	Presentation
HTML	*JavaScript* (event handling)	*CSS*
`<head>` `<script src="myJS.js">` `<link rel="stylesheet" type="text/css" href="myCSS.css">` ...	myJS.js	myCSS.css

From HTML to DOM

HTML is a representation for hypermedia documents

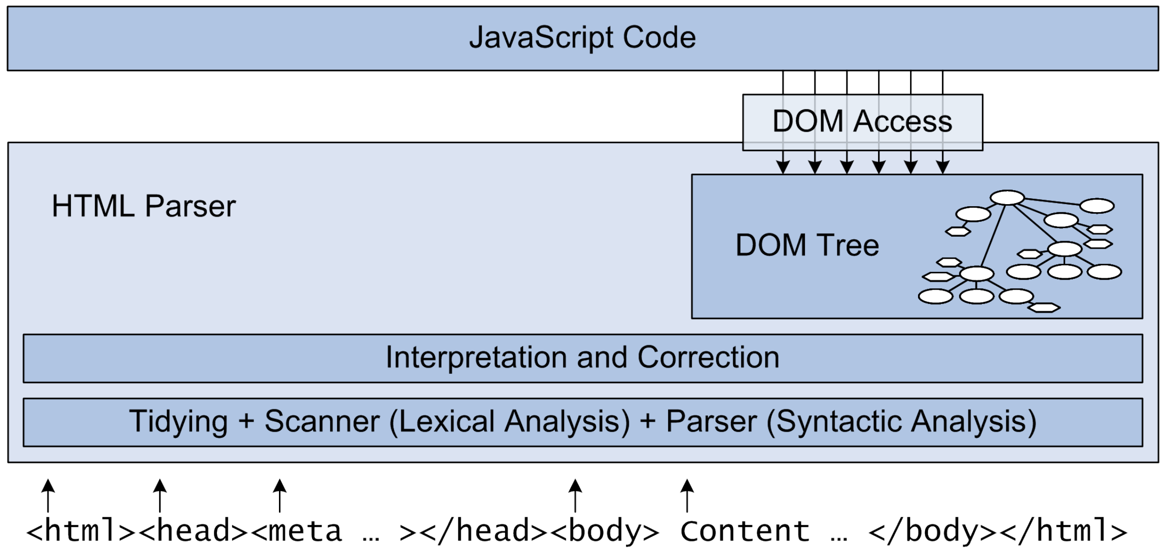
- A representation is required to store and transmit the document
- HTML uses markup for representing the document structure

Browsers must render HTML documents (i.e., apply CSS and execute JavaScript)

1. GET HTML from server and receive as text/html document
2. Parse document and deal with any errors by “fixing them”
3. Interpret document as if it had been error-free
4. GET all additional resources (CSS, images, JavaScript, …)
5. Build internal model (DOM) based on error-free interpretation
6. Apply CSS rules to determine styling of document (e.g., margins and font sizes)
7. Render into visual structure
8. Start executing JavaScript code
9. Listen for events (keyboard, mouse, timer) and execute code

W3C Document Object Model (DOM)

It describes a tree structure of all HTML elements, including attributes and the text they contain. (http://www.w3.org/DOM/DOMTR)


JavaScript Code
DOM Access
HTML Parser
DOM Tree
Interpretation and Correction
Tidying + Scanner (Lexical Analysis) + Parser (Syntactic Analysis)
<html><head><meta … ></head><body> Content … </body></html>

DOM is under constant revision

- DOM0 was invented by Netscape (backing the LiveScript/JavaScript)
- DOM1 was the first DOM version produced by the W3C
- DOM2 is the currently available stable version of DOM
- DOM3 is highly modularized and still under development

DOM representation of an HTML page

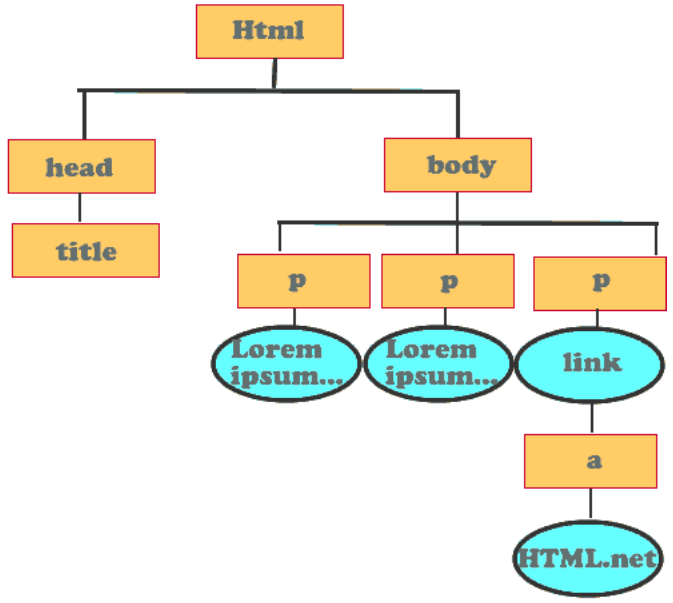
Each HTML document loaded into a browser window becomes a Document object

- Provides access to all HTML elements in a page, from within a script.
- Is also part of the Window object, and can be accessed through the window.document property.

DOM describes how all elements in an HTML page are related to the topmost structure: the document itself.

Example

```
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <p>Lorem ipsum ...</p>
    <p>Lorem ipsum ...</p>
    <p>link <a href="http://html.net">HTML.net</a></p
  </body>
</html>
```



Freie Universität Berlin

Accessing elements

By name

    document.getElementsByTagName(“td”)[indexOfColumn]

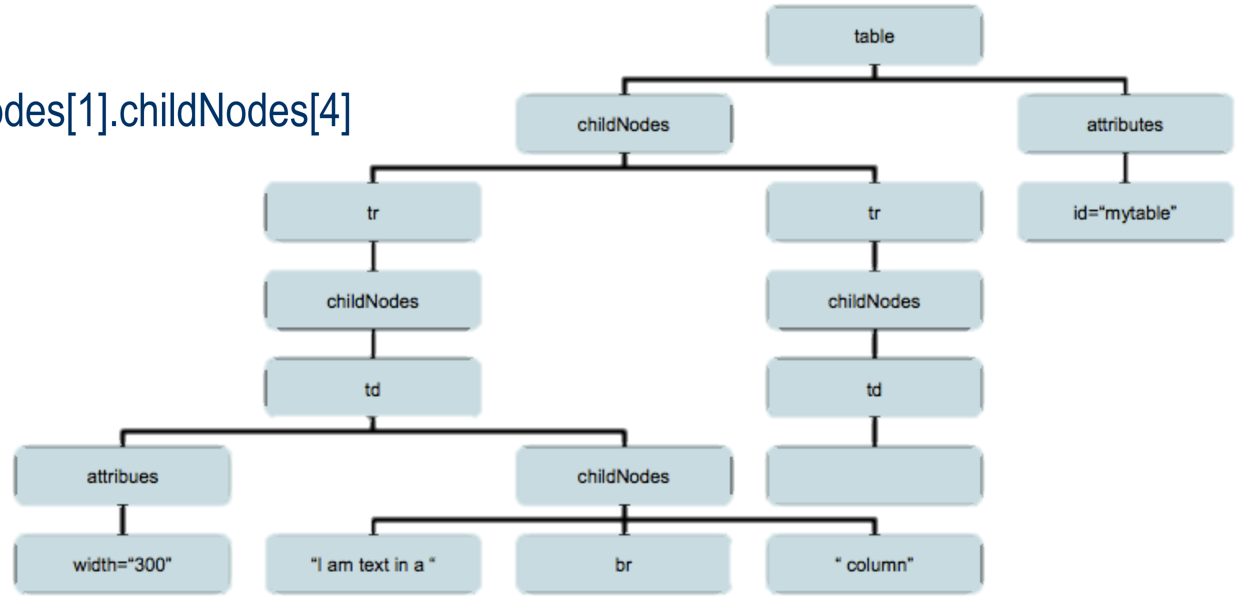
By ID

    document.getElementById(“mytable”)

Walk the DOM Tree

    document.childNodes[0].childNodes[1].childNodes[4]

```
<table id=“mytable”>
  <tr>
      <td width=“300”>
      I am text in a <BR> column
      </td>
  </tr>
<tr>
    <td> ... </td>
</tr>
<table>
```



JavaScript

JavaScript Frameworks

JavaScript Frameworks

Different needs produce different frameworks

- Dojo (http://dojotoolkit.org/)
- Ext JS (http://www.sencha.com/products/extjs/)
- jQuery (http://jquery.com/)
- Processing.js (http://processingjs.org/)

More comprehensive list is on Wikipedia:
http://en.wikipedia.org/wiki/List_of_JavaScript_libraries

There is no such thing as the “best JavaScript framework”

- For any given project, decide on the support you need
- Evaluate frameworks for the support they provide
- Evaluate for *functional requirements* (“is there a collapse/expand folder view?”)
- Evaluate for *non-functional requirements* (“is the framework openly licensed”)

JavaScript Frameworks

jQuery

jQuery basics

jQuery is a JavaScript Library following the mantra: *Find stuff and do stuff with it!*

It focuses on simplicity.

URL: http://jquery.com

Elements in the document are not found with

document.getElementById() or document.getElementByTag()

But with

$(‘p’) or $(‘#id’)

jQuery uses CSS Selectors to find elements and returns them as an array of elements.

JavaScript Frameworks

Processing.js

What is Processing.js?

Processing.js is the sister project of the popular Processing visual programming language, designed for the web. Processing.js makes your *data visualizations, digital art, interactive animations, educational graphs, video games,* etc. work using web standards and without any plug-ins.

It uses HTML5's `<canvas>` element and converts the Processing code to JavaScript and runs it. How can I use it?

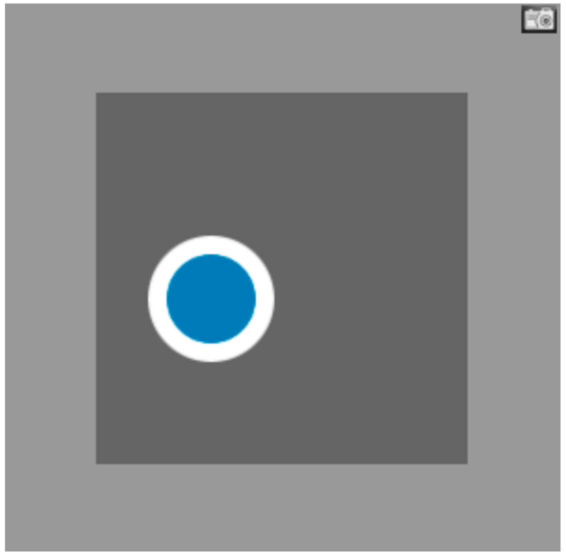
1. To use it, download Processing.js here: http://processingjs.org/download/
2. Make your Processing *.pde files as you normally would, for example in the “Visual Analytics” seminar
3. Create a web page that includes Processing.js as well as a `<canvas>` with info about where to get your sketch file (you can specify multiple *.pde files, separating them with spaces):

```
<script src="processing-1.0.0.min.js"></script>
<canvas data-processing-sources="hello-web.pde"></canvas>
```

4. Load your web page, and it will parse, translate, and run your sketch in the browser.

Example…

```
// Global variables
float radius = 50.0;
int X, Y;
int nX, nY;
int delay = 16;

// Setup the Processing Canvas
void setup(){
  size( 200, 200 );
  strokeWeight( 10 );
  frameRate( 15 );
  X = width / 2;
  Y = height / 2;
  nX = X;
  nY = Y;
  }

// Main draw loop
void draw(){

  radius = radius + sin( frameCount / 4 );

  // Track circle to new destination
  X+=(nX-X)/delay;
  Y+=(nY-Y)/delay;

  // Fill canvas grey
  background( 100 );

  // Set fill-color to blue
  fill( 0, 121, 184 );

  // Set stroke-color white
  stroke(255);

  // Draw circle
  ellipse( X, Y, radius, radius );
  }

// Set circle's next destination
void mouseMoved(){
  nX = mouseX;
  nY = mouseY;
  }
```



http://sketchpad.cc/

JavaScript

Check the following out – interesting stuff…

CoffeeScript

CoffeeScript is a programming language that transcompiles to JavaScript.

The language adds syntactic sugar to enhance JavaScript's brevity and readability, as well as adding more sophisticated features like array comprehension and pattern matching.

CoffeeScript compiles predictably to JavaScript and programs can be written with less code (typically 1/3 fewer lines) with no effect on runtime performance.

http://en.wikipedia.org/wiki/CoffeeScript

URL: http://coffeescript.org/

Dart is a Web programming language developed by Google. The goal of Dart is *"ultimately to replace JavaScript as the lingua franca of web development on the open web platform."*

http://en.wikipedia.org/wiki/Dart_(programming_language)

Dart is a new class-based programming language for creating structured web applications. Developed with the goals of simplicity, efficiency, and scalability, the Dart language combines powerful new language features with familiar language constructs into a clear, readable syntax.

http://www.dartlang.org/docs/technical-overview/index.html

URL: http://www.dartlang.org

Web application development (part 2)

Asynchronous Java and XML (AJAX)

Freie Universität Berlin

AJAX basics

Asynchronous Java and XML (AJAX) is a technology which allows client-side JavaScript to make requests to a server without causing a reload of the current page in the browser.

Using AJAX and standard DOM methods, client-side JavaScript can request, receive, and visualize information in the context of a single Web page.

The advantage of AJAX over more traditional Web pages is that they better resemble the behavior of desktop applications, providing enhanced features and usability for the user.

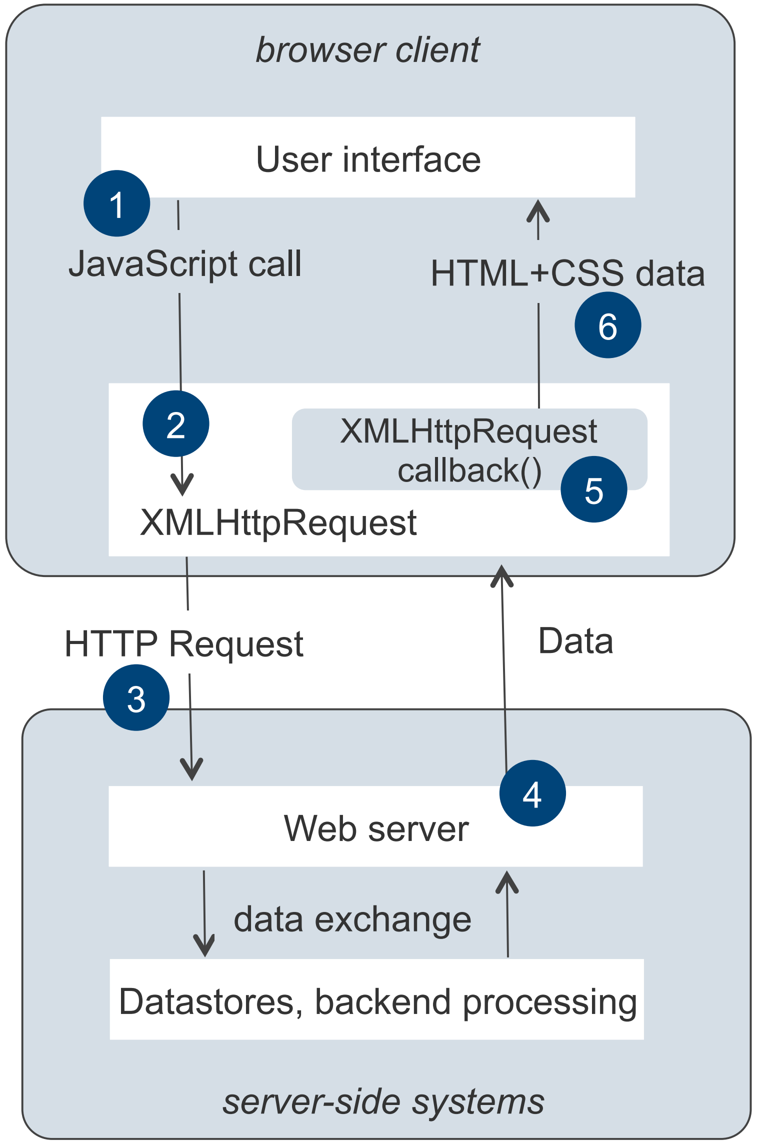
AJAX is not a technology per se, but a web development method and its aim is to increase the interactivity of a website.

The basic idea is, when you work with a web page, the web page itself can be interacting with the server, but without reloading the (whole) page you are on. The result is an uninterrupted user experience, with full server-side capabilities.

Freie Universität Berlin

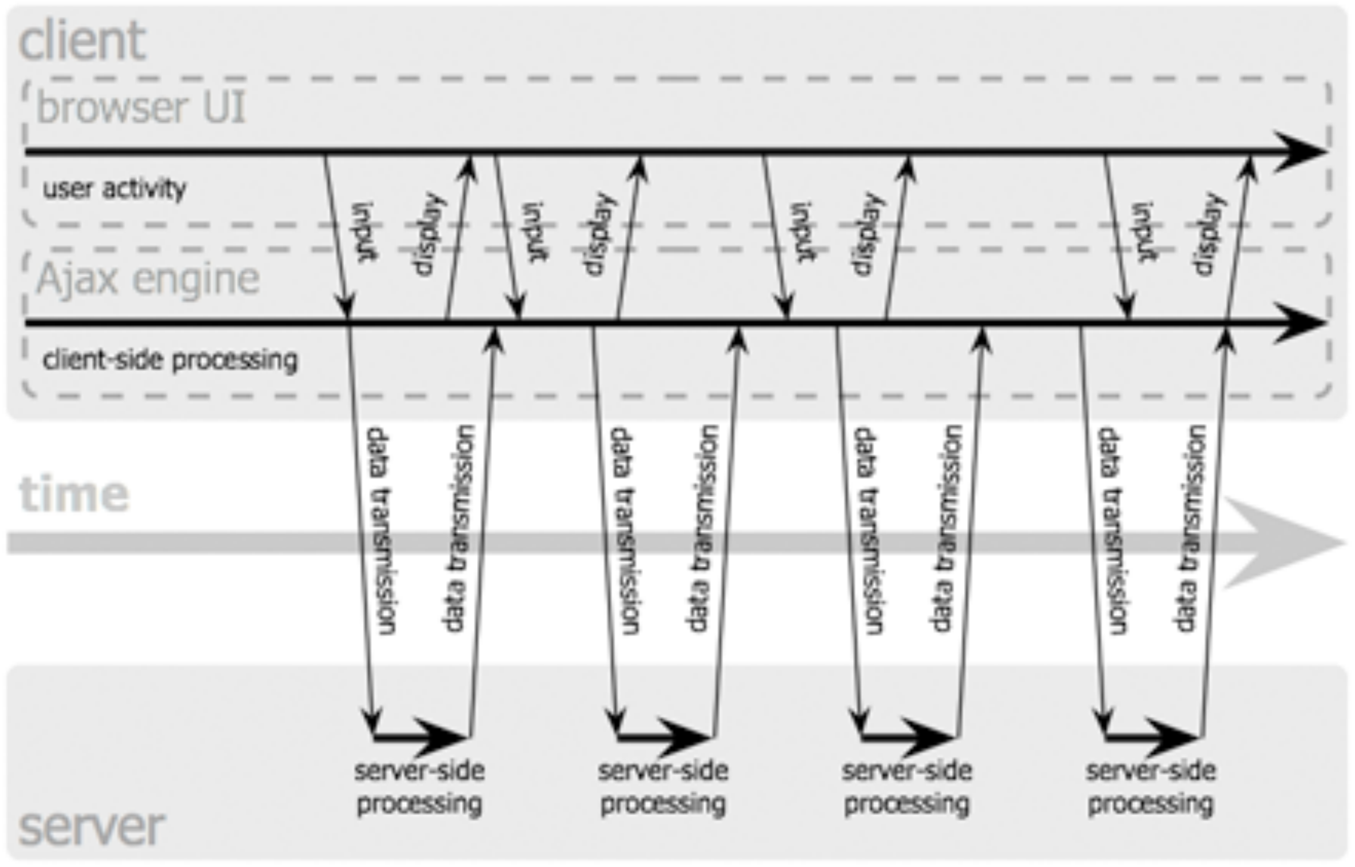
A typical Ajax request

1. User clicks, invoking event handler
2. Handler's JS code creates an XMLHttpRequest object
3. XMLHttpRequest object requests a document from a web server
4. Server retrieves appropriate data, sends it back
5. XMLHttpRequest fires event to say that the data has arrived (this is often called a callback; you can attach a handler to be notified when the data has arrived)
6. Your callback event handler processes the data and displays it



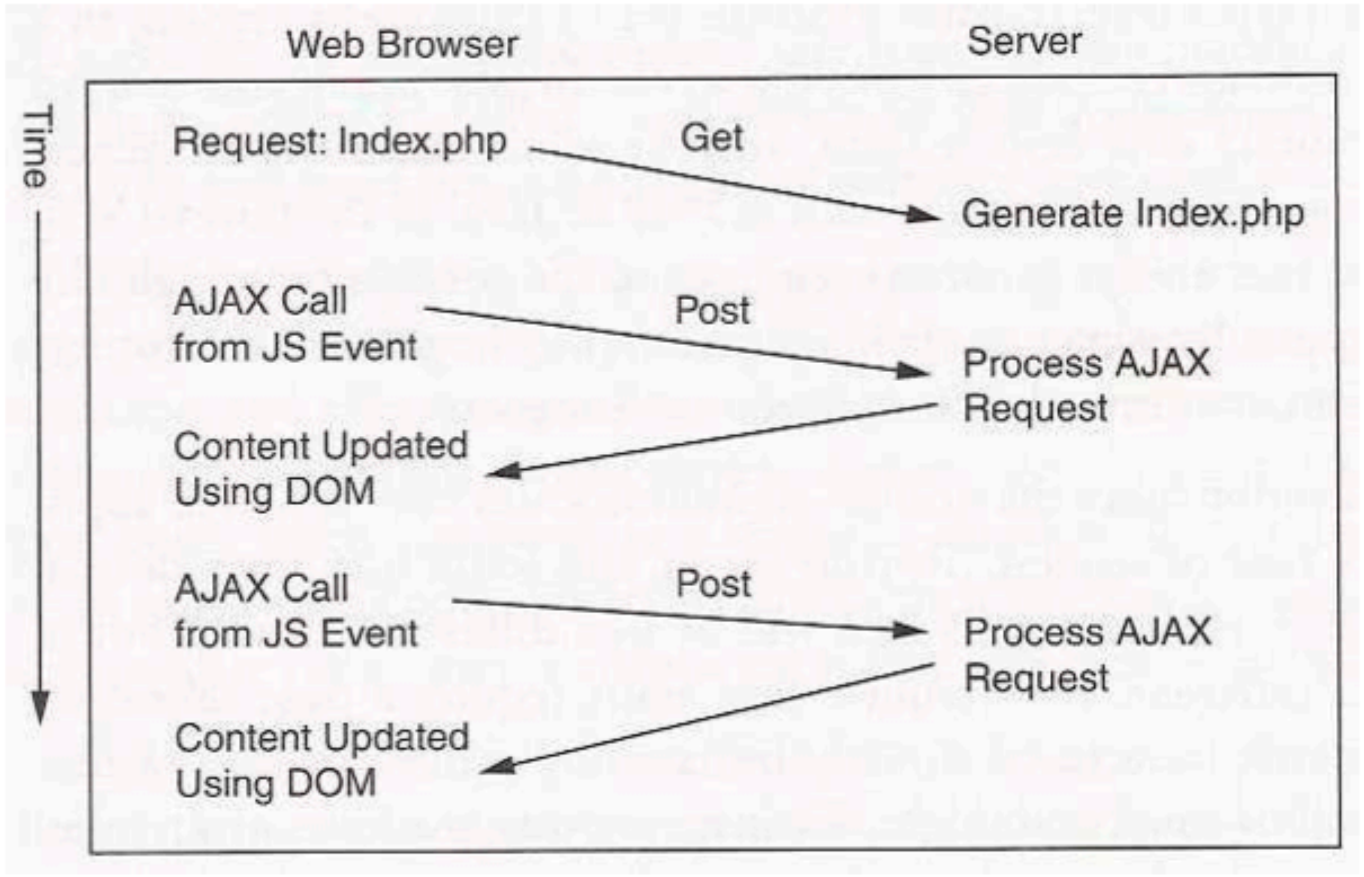
Freie Universität Berlin

Synchronous vs. asynchronous interaction pattern



Jesse James Garrett / adaptivepath.com

Web application request flow vs. Ajax request flow



AJAX is not a technology…

It rather incorporates several technologies.

Standards-based presentation using **XHTML and CSS**

Dynamic display and interaction using the **Document Object Model**

Data interchange and manipulation using **XML and XSLT**

Asynchronous data retrieval using **XMLHttpRequest**

- Used to submit an HTTP request to the server to exchange data
- Occurs within the JavaScript of the page

JavaScript that handles all the client-side functionality, such as capturing events

Core AJAX concepts

JavaScript includes an XMLHttpRequest object that can fetch files from a web server.

XMLHttpRequest can load HTTP requests

- *Synchronously*: The browser will wait for the response. This is a simpler model for the developer, but the resulting blocking is not acceptable.
- *Asynchronously*: The browser will not wait, but the object will call an event handler when the loading status changes. This is what you would use in practice

The contents of the fetched file can be put into current web page using the DOM.

XMLHttpRequest Methods

abort()	Stops the current request
getAllResponseHeaders()	Returns all headers (name and value) as a string
	Returns the value of the specified header
getResponseHeader(<headerName>)	Opens a connection and retrieves response from the specified URL.
open(“GET”/“POST”, “URL” [, true/false [, <username>,<password>]])	Can also specify optional values method (GET/POST), username and password for secured sites
send(content)	Transmits request (can include postable string or DOM object data)
setRequestHeader(<name>, <value>)	Assigns values to the specified header

XMLHttpRequest Properties

Onreadystatechange	Event handler for an event that fires at every state change
readyState	holds the status of the XMLHttpRequest, if the readyState changes → onreadystatechange handler runs; possible state values: 0 = uninitialized, 1 = loading, 2 = loaded, 3 = interactive, 4 = complete (readyState)
responseText	Data returned from server in string form
responseXML	DOM-compatible document object of data returned
status	HTTP status code (i.e., 200, 404, 500, etc.)
statusText	String message associated with the status code

Freie Universität Berlin

Simple AJAX example

```
<script type="text/javascript" language="javascript">
        function sndReq() {
                var resObject = new XMLHttpRequest();

                resObject.open('GET', 'laender.php?wo=1',true);
                resObject.onreadystatechange= handleResponse;
                resObject.send(null); }

        function handleResponse() {
                if(resObject.readyState == 4 {
                        document.getElementById(„hs“).innerHTML=
                        resObject.responseText; } }
</script>
<span style="cursor: pointer; text-decoration: underline” onclick="sndReq()”>
        Einen Request absetzen.
</span>
<span id=“hs“>
</span>
```

Simple AJAX example *(cont.)*

```
<script type="text/javascript" language="javascript">
        function sndReq() {
                var resObject = new XMLHttpRequest();

                resObject.open('GET', 'laender.php?wo=1',true);
                resObject.onreadystatechange= handleResponse;
                resObject.send(null); }

        function handleResponse() {
                if(resObject.readyState == 4 {
                        document.getElementById(„hs“).innerHTML=
                        resObject.responseText; } }
</script>
<span style="cursor: pointer; text-decoration: underline” onclick="sndReq()”>
        Einen Request absetzen.
</span>
<span id=“hs“>
</span>
```

Simple AJAX example *(cont.)*

```
<script type="text/javascript" language="javascript">
        function sndReq() {
                var resObject = new XMLHttpRequest();

                resObject.open('GET', 'laender.php?wo=1',true);
                resObject.onreadystatechange= handleResponse;
                resObject.send(null); }

        function handleResponse() {
                if(resObject.readyState == 4 {
                        document.getElementById(„hs“).innerHTML=
                        resObject.responseText; } }
</script>
<span style="cursor: pointer; text-decoration: underline” onclick="sndReq()”>
        Einen Request absetzen.
</span>
<span id=“hs“>
</span>
```

Simple AJAX example *(cont.)*

```
<script type="text/javascript" language="javascript">
        function sndReq() {
                var resObject = new XMLHttpRequest();

                resObject.open('GET', 'laender.php?wo=1',true);
                resObject.onreadystatechange= handleResponse;
                resObject.send(null); }

        function handleResponse() {
                if(resObject.readyState == 4 {
                        document.getElementById(„hs“).innerHTML=
                        resObject.responseText; } }
</script>
<span style="cursor: pointer; text-decoration: underline” onclick="sndReq()”>
        Einen Request absetzen.
</span>
<span id=“hs“>
</span>
```

Simple AJAX example *(cont.)*

```
<script type="text/javascript" language="javascript">
        function sndReq() {
                var resObject = new XMLHttpRequest();

                resObject.open('GET', 'laender.php?wo=1',true);
                resObject.onreadystatechange= handleResponse;
                resObject.send(null); }

        function handleResponse() {
                if(resObject.readyState == 4 {
                        document.getElementById(„hs“).innerHTML=
                        resObject.responseText; } }
</script>
<span style="cursor: pointer; text-decoration: underline” onclick="sndReq()”>
        Einen Request absetzen.
</span>
<span id=“hs“>
</span>
```

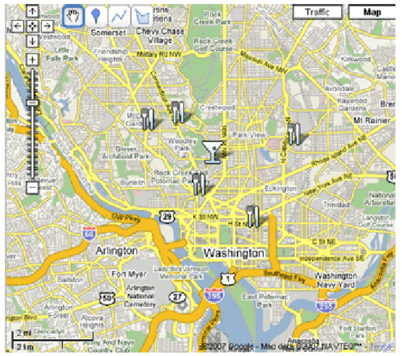
```
if (resObject.status == 200) {
do something with resObject.responseText;
} else {
code to handle the error;
}
```

AJAX examples

This is a very simple idea, but it can have very nice results, such as:

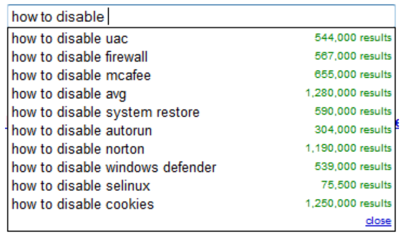
Google Maps (and A9 Maps)

- An image displaying a map of a specified region is inserted into a web page
- The user clicks the scroll left button
- The JavaScript module loads a new image and inserts it into the current document



Google Suggest (and Amazon Suggest):

- The user types in part of a keyword
- The JavaScript module sends the partial keyword to the server, and gets back a list of matches, which are shown to the user



how to disable

how to disable uac	544,000 results
how to disable firewall	567,000 results
how to disable mcafee	655,000 results
how to disable avg	1,280,000 results
how to disable system restore	590,000 results
how to disable autorun	304,000 results
how to disable norton	1,190,000 results
how to disable windows defender	539,000 results
how to disable selinux	75,500 results
how to disable cookies	1,250,000 results

close

Using AJAX - Pros and Cons

PROS

[+] Enhanced User Experience
[+] Reduced Bandwidth Usage
[+] Increased User Productivity
[+] Increased Compatibility
[+] Faster & Enhanced Application Development

CONS

[-] Learning Curve For Developers
[-] Effects On End Users
[-] Not Search Engine Friendly
[-] Dependency Of Browser Settings

Web application development (part 2)

Web Sockets

Need for polling/long-polling solutions

Comet is a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it. It is an umbrella term, encompassing multiple techniques for achieving this interaction. Comet applications attempt to eliminate the limitations of the page-by-page web model and traditional polling by offering real-time interaction, using a persistent or long-lasting HTTP connection between the server and the client. (http://en.wikipedia.org/wiki/Comet_(programming))

With *polling*, the browser sends HTTP requests at regular intervals and immediately receives a response. This technique was the first attempt for the browser to deliver real-time information.

With *long-polling*, the browser sends a request to the server and the server keeps the request open for a set period. If a notification is received within that period, a response containing the message is sent to the client. If a notification is not received within the set time period, the server sends a response to terminate the open request.

Websockets basics

The WebSocket specification (http://dev.w3.org/html5/websockets/) introduced the WebSocket JavaScript interface, which defines a full-duplex single socket connection which messages can be sent between client and server.

The WebSocket standard simplifies much of the complexity around bi-directional web communication and connection management.

Web Sockets provide an enormous reduction in unnecessary network traffic and latency compared to the unscalable polling and long-polling solutions that were used to simulate a full-duplex connection by maintaining two connections.

More information

http://www.infoq.com/presentations/WebSockets-The-Web-Communication-Revolution

http://websocket.org/

Web application development (part 2)

Summary

What have we discussed today?

We discussed the client side scripting (run in the browser) based on JavaScript (another example is VB script).

Motivated by the JavaScript principle we talked about the Document Object Model; why it is useful and how can it be used?

We did not talked about client side application programs that are programs that run as an independent program in the client machine. The typical example are Java Applets.

Anyway, we introduced the Asynchronous Java and XML (AJAX) which allows client-side JavaScript to make requests to a server without causing a reload of the current page in the browser. We discussed the interaction patterns and the request flows and compared it to a typical web application. You should know the steps of a typical Ajax request.

We did not stop with AJAX. We also shortly introduced Websockets, a promising new approach for full-duplex single socket connection.

References

Marty Stepp (2007): Web Programming. Computer Science & Engineering, University of Washington. http://www.cs.washington.edu/education/courses/cse190m/

Erik Wilde and Dilan Mahendran (2011): Web Architecture and Information Management. School of Information, UC Berkeley. http://dret.net/lectures/web-spring11/

Gregory V. Wilson (2005) CSC309: Web Programming. Computer Science - University of Toronto. http://www.cdf.toronto.edu/~csc309h/summer

Additional resources:

http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

http://www.comptechdoc.org/independent/web/cgi/javamanual

http://www.quirksmode.org/dom/intro.html

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

http://www.html.net/tutorials/javascript/lesson14.php