Univerzitet Singidunum

Web programming

www.singidunum.ac.rs

AJAX
Asynchronous JavaScript + XML

“a fundamental shift in what’s possible on the Web”

The AJAX Name

- Ajax: A New Approach to Web Applications
- Jesse James Garrett
- February 18, 2005
- Adaptive Path
  - http://www.adaptivepath.com/publications/essays/archives/000385.php

The *Name* Tipped the Scales

- The whole Ajax idea has tipped, thanks in large part to the fact that there is now a name attached to it that is quite easy for everyone to get.

Motivation

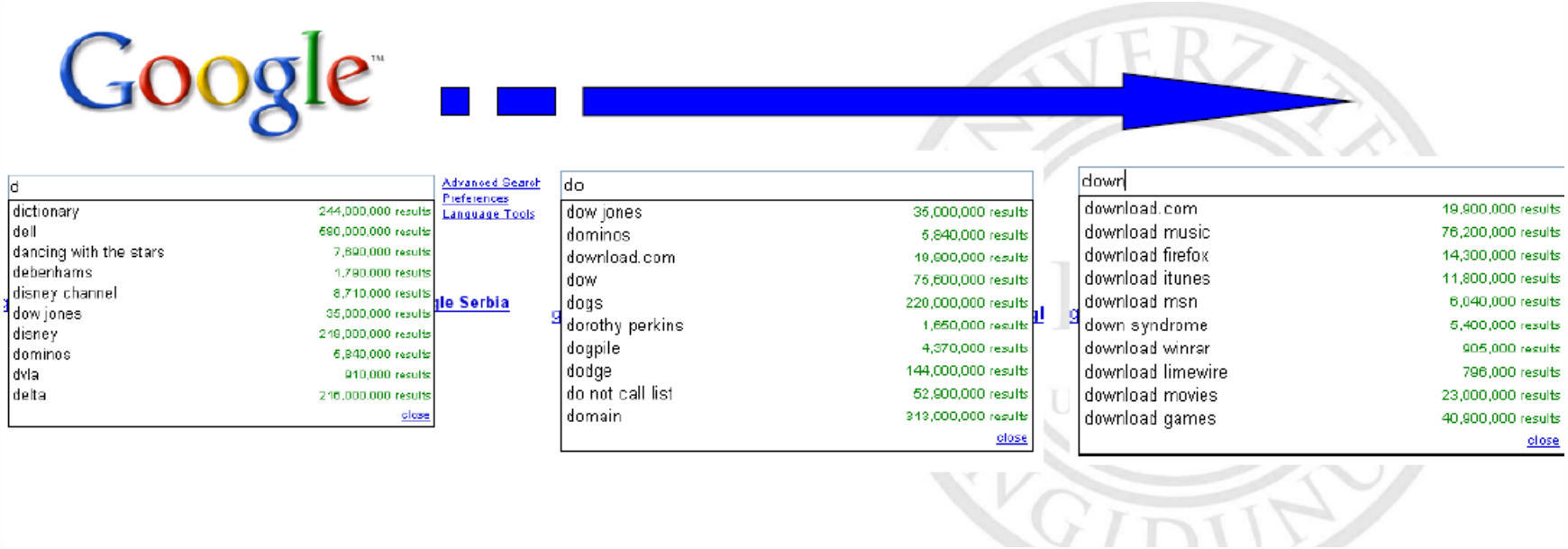
- On one hand
  - desktop applications have a richness and responsiveness that has seemed out of reach on the Web.
- On the other
  - simplicity has enabled the Web's rapid proliferation
- Result – a *gap* between
  - the experiences we can provide on the Web and
  - the experiences users can get from a desktop application.

The Gap is Closing

- *Google Suggest*
  - Suggested terms update as you type, almost instantly.
- *Google Maps*
  - Zoom in.
  - Use your cursor to grab the map and scroll around a bit.
  - Everything happens almost instantly, with no waiting for pages to reload.

Google Suggest

- When typing words in search field of Google, JavaScript sends entered characters, and gets from the server list of the possible destinations, offering them to the user interactively with further typing



Google Suggest

- Standard client-server communication – request to the server for a web page
- Server interprets the request, translates server code, sends back new HTML code representing new page
- AJAX – first page is loaded in standard way, without using AJAX, on user request. Further changes of the content in the page are displayed based on the user request, with contact to the server, but without sending the complete page code from the server
- AJAX – faster, better and user oriented web pages
- AJAX – one of the technologies that brought term WEB 2.0, and changed standard approaches to web pages

Classic Web Application Model

- Most user actions in the interface trigger an HTTP request back to a web server.
- The server does some processing and then returns an HTML page to the client. Processing includes
  - retrieving data
  - crunching numbers
  - talking to various legacy systems.
- This model is adapted from the Web's original use as a hypertext medium.
- *But what makes the Web good for hypertext doesn't necessarily make it good for software applications.*

What is the Problem?

- This approach makes a lot of technical sense.
- *But* it doesn't make for a great user experience.
  - While the server is doing its thing, what's the user doing?
    - Waiting.
    - And at every step in a task, the user waits some more.

What is the Problem?

- Obviously, if we were designing the Web from scratch for applications, we wouldn't make users wait around.
- Once an interface is loaded, why should the user interaction come to a halt every time the application needs something from the server?
- In fact, why should the user see the application go to the server at all?

What is the Answer?

- Eliminate the start-stop-start-stop nature of interaction on the Web by introducing an intermediary between the user and the server.
  - The Ajax engine!
- It seems like adding a layer to the application would make it less responsive
  - but the opposite is true.
- Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine
  - written in JavaScript and usually tucked away in a hidden frame.
- This engine is responsible for
  - rendering the interface the user sees
  - communicating with the server on the user's behalf.
- The Ajax engine allows the user's interaction with the application to happen *asynchronously*
  - i.e., independent of communication with the server.
- The user is never staring at a blank browser window and an hourglass icon, waiting around for the server to do something.

What is the Answer?

- Every user action that normally would generate an HTTP request takes the form of a JavaScript call to the *Ajax engine* instead.
- The engine handles the response to a user action that doesn't require a trip back to the server
  - simple data validation
  - editing data in memory
  - some navigation
- If the engine needs something from the server in order to respond it makes those requests asynchronously, usually using XML, without stalling a user's interaction with the application
  - submitting data for processing
  - loading additional interface code
  - retrieving new data
- The server returns the requested data in the form of XML documents.
- The XML documents may then be used by the JavaScript technology to update or modify the Document Object Model (DOM) of the HTML page

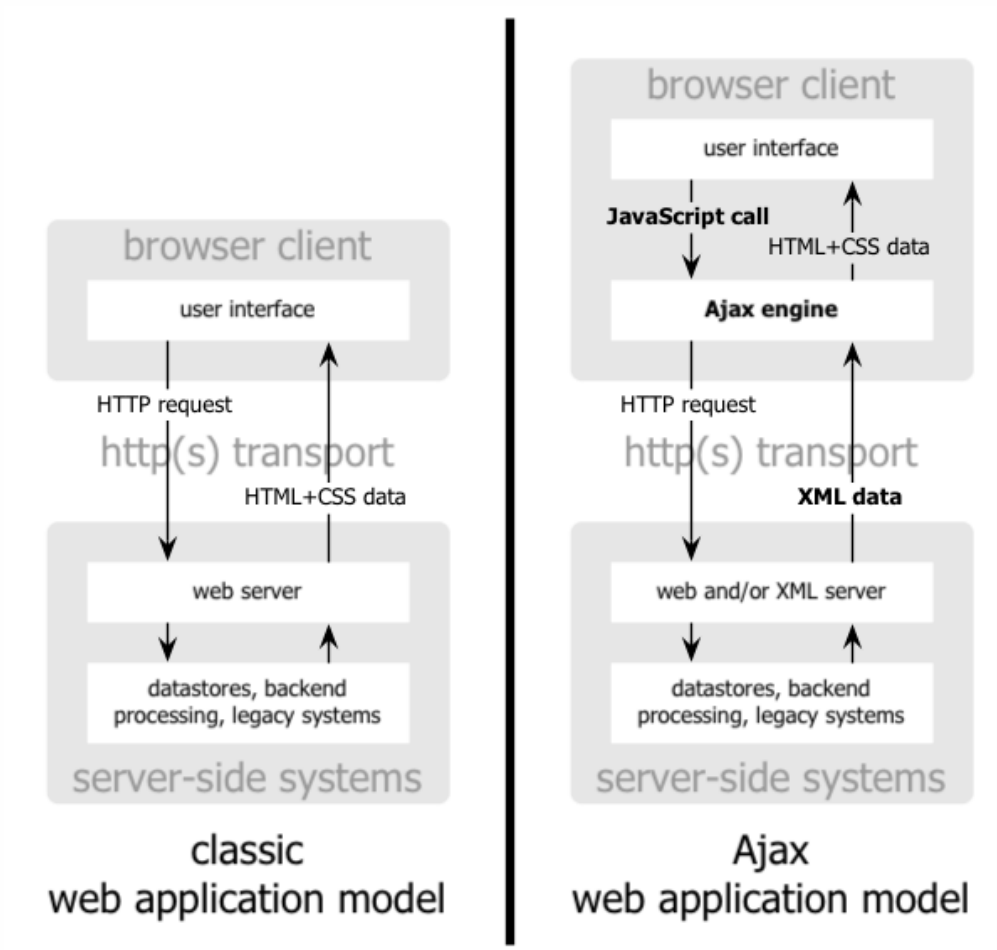
What is Ajax?

- Is Ajax a *technology platform* or is it an *architectural style*?
  - It's both. Ajax is a set of technologies being used together in a particular way.
- Where can I download it?
  - Ajax isn't something you can download.
  - It's an approach, a way of thinking about the architecture of web applications using certain technologies.

Defining AJAX

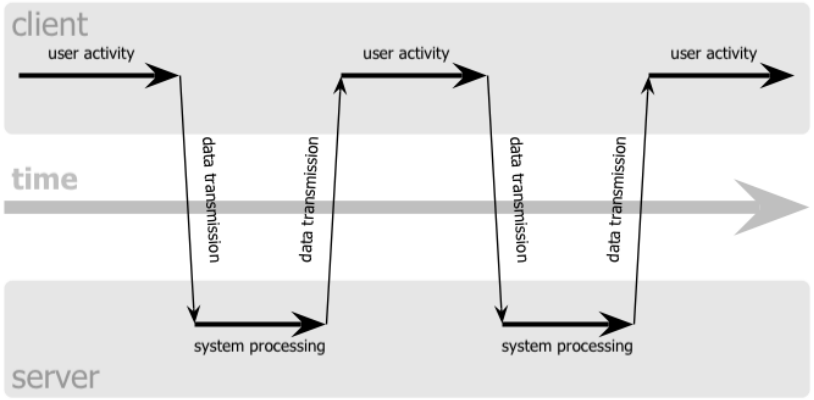
- Ajax isn't a *single* technology.
- It's really several independent technologies coming together in new ways.
- Ajax incorporates:
  - *standards based presentation* using XHTML and CSS
  - *dynamic display and interaction* using the Document Object Model
  - *data interchange and manipulation* using XML and XSLT
  - *asynchronous data retrieval* using XMLHttpRequest
  - *JavaScript* binding everything together

Classic Model *v* AJAX

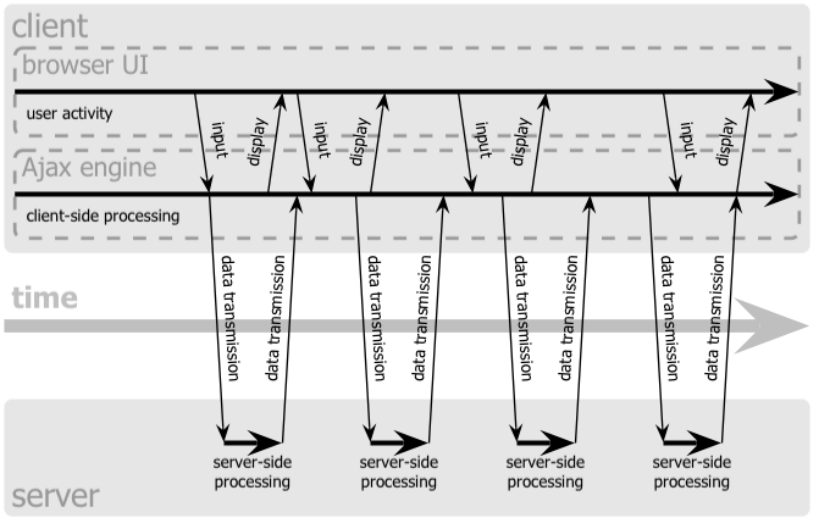


Classic Model *v* Ajax

classic web application model (synchronous)



Ajax web application model (asynchronous)



Extensible HyperText Markup Language - XHTML

- A markup language that has the same expressive possibilities as HTML, but a stricter syntax.
- HTML is an application of SGML - a very flexible markup language.
- XHTML is an application of XML - a more restrictive subset of SGML.
- XHTML documents allow for automated processing to be performed using a standard XML library
  - because they need to be well-formed (syntactically correct).
- HTML, on the other hand, requires a relatively complex, lenient, and generally custom parser.
- The need for a more strict version of HTML was felt primarily because WWW content now needs to be delivered to many devices (like mobile devices) apart from traditional computers
  - where extra resources cannot be devoted to support the additional complexity of HTML syntax.

Cascading Style Sheets (CSS)

- A **stylesheet language** is a computer language used to describe the style of elements in a document marked up using a markup language.
- **CSS** is a stylesheet language.
- Its most common application is to style web pages written in HTML and XHTML
  - but the language can be applied to any kind of XML document.
- The CSS specifications are maintained by the World Wide Web Consortium.

Cascading Style Sheets (CSS)

- CSS is used by both the authors and readers of web pages to define
  - colors
  - fonts
  - layout
  - other **aspects** of document presentation.
- It is designed primarily to enable the separation of document structure (written, for example, in XHTML) from document presentation (written in CSS).
- This separation can
  - improve content accessibility
  - provide more flexibility and control in the specification of presentational characteristics
  - reduce complexity and repetition in the structural content.
- CSS can also allow the same markup page to be presented in different styles for different rendering methods, for example
  - on-screen
  - in print
  - by voice (when read out by a speech-based browser or screen reader)
  - on braille-based, tactile devices.

Document Object Model (DOM)

- A description of how a HTML or XML document is represented in an object-oriented fashion.
- DOM provides an API to access and modify the content, structure and style of the document.
- Using DOM, the document is accessed in a tree form
  - this is also the data structure that most XML parsers (e.g., Xerces) have been developed to make use of.
- Such an implementation requires that the entire content of a document be parsed and stored in memory.
- DOM is best used for applications where the document elements have to be accessed and manipulated in an unpredictable sequence and repeatedly.

Scripting Languages

- Languages initially designed for "scripting" the operations of a computer.
  - Early script languages were often called *batch languages* or *job control languages*.
- **Scripting** refers to the idea of connecting diverse pre-existing components to accomplish a new related task.
- Common properties of scripting languages
  - they favor rapid development over efficiency of execution
  - they are often implemented with interpreters rather than compilers
  - they are strong at communication with program components written in other languages.

Scripting Languages

- Many scripting languages emerged as tools for executing one-off tasks
  - particularly in system administration.
- Can be thought of as "glue" that puts several components together. For example,
  - they are widely used for creating graphical user interfaces
  - executing a series of commands that might otherwise have to be entered interactively through keyboard at the command prompt.
- The operating system usually offers some type of scripting language by default
  - widely known as a *shell script* language.
- The boundary between scripting languages and regular programming languages tends to be vague
  - and is blurring ever more with the emergence of new languages.
- In some scripting languages, an experienced programmer can accomplish a good deal of optimization if they choose.

Prototype-based Programming

- A style and subset of object-oriented programming in which
  - classes are *not present*
  - behavior reuse (known as inheritance in class-based languages) is done by *cloning* existing objects which serve as prototypes for the new ones
  - also known as *class-less*, *prototype-oriented*, or *instance-based* programming.

Prototype-based Programming

- Instead of data-containing instances and code-containing classes, prototype-based languages have only *objects*.
- A prototype system starts with at least one atomic object loaded
  - new objects are created by cloning existing ones
  - cloning an object creates an entirely new one that starts with the same default behaviors as its original.
- New objects contain a pointer to the object that created them
  - as opposed to having a pointer to a class of which it is an instance.
- Objects are largely empty, and only start growing in memory when changed.
  - This is different from class-based, object-oriented languages, where each instance of a class usually sets aside a known amount of memory.
- Additional data can be added to any object at any point at runtime.
  - Since objects grow as needed, anything can be added to them.
  - Every object tends to be different from every other
    - not only in the data themselves, but in what data are being recorded.
  - Not only data but also methods can be added or changed.
    - For this reason most prototype-based languages refer to both data and methods as "slots".

Prototype-based Programming

- Proponents of statically typed programming languages claim that *correctness*, *safety*, *efficiency* and *predictability* are more important than the increase in flexibility gained through the ability to modify code at run-time.
- A good example of this is the extensive use of JavaScript to implement Mozilla Firefox's user interface and its extensions.
  - The JavaScript running in the browser has higher security access than the JavaScript objects embedded in web pages, but often has to interact with untrusted objects.
  - In a non-statically-typed language, it can be quite difficult to guarantee that you have the object you think you have and that the method you're calling does what you think it does.
  - Calling a method that was replaced can cause untrusted code to run at a higher security level.
  - This has resulted in many security bugs.

JavaScript

- An object-based scripting programming language based on the concept of prototypes.
- Best known for its use in websites
  - but is also used to enable scripting access to objects embedded in other applications.
- Despite the name, JavaScript is only distantly related to the Java programming language.
  - The main similarity is their common debt to the C programming language.
  - JavaScript has far more in common with the *Self* programming language.

JavaScript

- JavaScript engines embedded in a web browser allow JavaScript to connect to both the server side and the client side of web applications.
- Connections are made through DOM interfaces.
- One major use of web-based JavaScript is to write functions that are embedded in HTML pages.
- Functions interact with the DOM of the page to perform tasks not possible in static HTML alone, such as
  - opening a new window
  - checking input values
  - changing images as the mouse cursor moves over
  - etc.

JavaScript

- JavaScript interpreters are embedded in a number of tools outside of the web.
  - Adobe Acrobat and Adobe Reader support JavaScript in PDF files.
  - The Mozilla platform, which underlies several common web browsers, uses JavaScript to implement the user interface and transaction logic of its various products.
  - JavaScript interpreters are also embedded in proprietary applications that lack scriptable interfaces.
    - Dashboard Widgets in Apple's Mac OS X v10.4 are implemented using JavaScript.
    - Microsoft's Active Scripting technology supports JavaScript-compatible JScript as an operating system scripting language.
    - JScript.NET is similar to JScript, but has further object oriented programming features.
- Each of these applications provides its own object model which provides access to the host environment
  - *with the core JavaScript language remaining mostly the same in each application*.

XMLHttpRequest

- **XMLHTTP** is a set of APIs that can be used to transfer and manipulate XML data to and from a web server using HTTP
- Used to establish an independent connection channel between Client-Side and Server-Side.
- Used by
  - JavaScript
  - Jscript
  - VBScript
  - other web browser scripting languages.

AJAX - The XMLHttpRequest Object

- The keystone of AJAX is the XMLHttpRequest object.
- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a web server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.
- All modern browsers (Chrome, Firefox, IE7+, Edge, Safari, Opera) have a built-in XMLHttpRequest object.

AJAX - The XMLHttpRequest Object

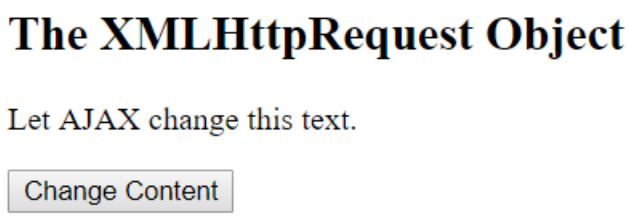
- Syntax for creating an XMLHttpRequest object:

  variable = new XMLHttpRequest();

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

AJAX - The XMLHttpRequest Object

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<p id="demo">Let AJAX change this text.</p>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```



AJAX - The XMLHttpRequest Object

- Access Across Domains
- For security reasons, modern browsers do not allow access across domains.
- This means that both the web page and the XML file it tries to load, must be located on the same server.
- If you want to use the example in this presentation on one of your own web pages, the XML files you load must be located on your own server.

AJAX - The XMLHttpRequest Object

- Older Browsers (IE5 and IE6)
- Old versions of Internet Explorer (5/6) use an ActiveX object instead of the XMLHttpRequest object:

  variable = new ActiveXObject("Microsoft.XMLHTTP");
- To handle IE5 and IE6, check if the browser supports the XMLHttpRequest object, or else create an ActiveX object:

```
if (window.XMLHttpRequest) {
    // code for modern browsers
    xmlhttp = new XMLHttpRequest();
 } else {
    // code for old IE browsers
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

AJAX - The XMLHttpRequest Object

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

AJAX - The XMLHttpRequest Object

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

AJAX - Send a Request To a Server

- The XMLHttpRequest object is used to exchange data with a server.
- To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

AJAX - Send a Request To a Server

Method	Description
open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

AJAX - Send a Request To a Server

- GET or POST?
- GET is simpler and faster than POST, and can be used in most cases.
- However, always use POST requests when:
  - A cached file is not an option (update a file or database on the server).
  - Sending a large amount of data to the server (POST has no size limitations).
  - Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

AJAX - Send a Request To a Server

- A simple GET request:

```
xhttp.open("GET", "demo_get.asp", true);
xhttp.send();
```

- In the example above, you may get a cached result. To avoid this, add a unique ID to the URL:

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();
```

AJAX - Send a Request To a Server

- If you want to send information with the GET method, add the information to the URL:

```
xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
xhttp.send();
```

AJAX - Send a Request To a Server

- A simple POST request:

```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

- To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

AJAX - Send a Request To a Server

- The url - A File On a Server
- The url parameter of the open() method, is an address to a file on a server:

```
xhttp.open("GET", "ajax_test.asp", true);
```

- The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

AJAX - Send a Request To a Server

- Asynchronous - True or False?
- Server requests should be sent asynchronously.
- The async parameter of the open() method should be set to true:

```
xhttp.open("GET", "ajax_test.asp", true);
```

- By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:
  - execute other scripts while waiting for server response
  - deal with the response after the response is ready

Ajax – Send a Request To a Server

- The onreadystatechange Property
- With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.
- The function is defined in the onreadystatechange property of the XMLHttpResponse object:

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Ajax – Send a Request To a Server

- Synchronous XMLHttpRequest (async = false) is not recommended because the JavaScript will stop executing until the server response is ready. If the server is busy or slow, the application will hang or stop.
- Synchronous XMLHttpRequest is in the process of being removed from the web standard, but this process can take many years.
- Modern developer tools are encouraged to warn about using synchronous requests and may throw an InvalidAccessError exception when it occurs.

Ajax – Server response

- The onreadystatechange Property
- The **readyState** property holds the status of the XMLHttpRequest.
- The **onreadystatechange** property defines a function to be executed when the readyState changes.
- The **status** property and the statusText property holds the status of the XMLHttpRequest object.

Ajax – Server response

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Ajax – Server response

- The onreadystatechange function is called every time the readyState changes.
- When readyState is 4 and status is 200, the response is ready:

```
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML =
            this.responseText;
        }
    };
    xhttp.open("GET", "ajax_info.txt", true);
    xhttp.send();
}
```

Ajax – Server response

Server Response Properties

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Server Response Methods

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

Ajax – Server response

- The responseText Property
- The responseText property returns the server response as a JavaScript string, and you can use it accordingly

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

Ajax – Server response

- The responseXML Property
- The XML HttpRequest object has an in-built XML parser.
- The responseXML property returns the server response as an XML DOM object.
- Using this property you can parse the response as an XML DOM object

Ajax – Server response

```
xmlDoc = xhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
  txt += x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("demo").innerHTML = txt;
xhttp.open("GET", "cd_catalog.xml", true);
xhttp.send();
```

Bob Dylan
Bonnie Tyler
Dolly Parton
Gary Moore
Eros Ramazzotti
Bee Gees
Dr.Hook
Rod Stewart
Andrea Bocelli
Percy Sledge
Savage Rose
Many
Kenny Rogers
Will Smith

Ajax – Server response

```
▼<CATALOG>
  ▼<CD>
      <TITLE>Empire Burlesque</TITLE>
      <ARTIST>Bob Dylan</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>Columbia</COMPANY>
      <PRICE>10.90</PRICE>
      <YEAR>1985</YEAR>
    </CD>
  ▼<CD>
      <TITLE>Hide your heart</TITLE>
      <ARTIST>Bonnie Tyler</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>CBS Records</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1988</YEAR>
    </CD>
```

Ajax – Server response

- **The getAllResponseHeaders() Method**
- The getAllResponseHeaders() method returns all header information from the server response.

- **The getResponseHeader() Method**
- The getResponseHeader() method returns specific header information from the server response.

Ajax – Server response

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getAllResponseHeaders();
  }
};
```

The XMLHttpRequest Object

The getAllResponseHeaders() function returns all the header information of a resource, like length, server-type, content-type, last-modified, etc:

date: Sun, 06 May 2018 21:54:45 GMT content-encoding: gzip vary: Accept-Encoding last-modified: Tue, 18 Jul 2017 16:14:29 GMT server: ECS (fcn/40FB) x-powered-by: ASP.NET etag: "15bfdeee0ffd21:0" x-frame-options: SAMEORIGIN x-cache: HIT content-type: text/plain status: 200 cache-control: public,max-age=14400,public accept-ranges: bytes content-length: 245

Ajax – Server response

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getResponseHeader("Last-Modified");
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

The XMLHttpRequest Object

The getResponseHeader() function is used to return specific header information from a resource, like length, server-type, content-type, last-modified, etc:

Last modified: Tue, 18 Jul 2017 16:14:29 GMT

Is “AJAX” really new?

- Many techniques that are used in Ajax architectures have been available to developers targeting Internet Explorer on the Windows platform for many years.
- Until recently, the technology was known as *web remoting* or *remote scripting*.
- Web developers have also used a combination of plug-ins, Java applets, and hidden frames to *emulate* this interaction model for some time.
- What has changed recently is that the inclusion of support for JavaScript’s *XMLHttpRequest* object has became ubiquitous in the mainstream browsers across all platforms.
  - Although this object is not specified in the formal JavaScript technology specification, all of today's mainstream browsers support it.
  - The subtle differences with the JavaScript technology and CSS support among current generation browsers such as Firefox, Internet Explorer, and Safari are manageable.
  - *However*, if you are required to support older browsers, AJAX may not be the answer for you.

Ajax Application Characteristics

- The client contains page-specific control logic embedded as JavaScript technology.
- The page interacts with the JavaScript technology based on events such as
  - the document being loaded
  - a mouse click
  - focus changes
  - a timer.
- AJAX interactions allow for a *clear separation* of presentation logic from the data.
- An HTML page can pull in bite-size pieces of data as needed rather than reloading the whole page every time a change needs to be displayed.

Ajax Application Characteristics

- AJAX requires a different server-side architecture to support this interaction model.
- Traditionally, server-side web applications have focused on generating HTML documents for every client event resulting in a call to the server.
- The clients would then refresh and re-render the complete HTML page for each response.
- Rich web applications focus on a client fetching an HTML document that acts as a *template* or *container*
  - Ajax engine injects content into container using XML data retrieved from a server-side component.

AJAX

- What happens when the aforementioned technologies interact?
- Javascript has access to the DOM of its webpage to
  - read information
  - change the DOM interactively.
- So, a web page can be built on the fly, and altered once it has been constructed.
- Build a page at load-time has always been possible in Javascript
  - But, the effect of changing it later has historically been quite erratic.
  - That's all pretty much sorted out now, which means that a DOM can be radically changed by Javascript, *even after it's been built*.

AJAX

- So a web page can contain (or link to) some Javascript.
  - Script can change its appearance.
  - This is still basically *static*, though:
    - the entire behaviour of the Javascript+HTML is already determined.
  - A page might be able to generate lots of pretty graphs or pictures or things, but it will do the same thing each time (pseudorandomness permitting).

AJAX

- To make web page truly *dynamic*, Javascript requires some external source of *input*.
- Traditionally, this has come from the user, through forms.
  - A user could
    - Click
    - move the mouse
    - fill in values in forms
  - Javascript can
    - take this input
    - perform some calculations
    - then change the DOM as a result.
  - For example, in a page containing a graph, different values from the user change the graph.

AJAX

- There's one other source of data available to a Javascript program:
  - *a network connection back to its web server*.
- This is where XML comes in:
  - both requests and information can be encoded as XML
  - sent to and from the program and the server.
- A web page can be dynamically updated based on this XML data.
- *In effect, web pages can be rebuilt on the fly*.

AJAX

- XML-based network communication is nothing new.
  - Both Flash and Java applets have had nice formalized models for bidirectional XML-based communication for some time.
  - Now, these kinds of rich interactions are available to *web pages*, based on standard and widely available technologies. Welcome to Ajax.
- In an Ajax page, an "Ajax engine" is loaded in the form of a Javascript program.
- This Ajax engine then builds and modifies the DOM of a web page based on XML interactions with its server.
- No new HTML pages are requested from the server.

AJAX

- This is a big shift from traditional web applications.
  - Usually, the HTML is built on the web-server and pushed out to the browser.
  - For most web sites, chances are that the new web page will be at least 90% the same as the last one
    - the structure and formatting is the same, and just some of the links and data will have changed.
  - But the whole lot has to be built, by the server, each time, and then transmitted.
  - If you're generating millions of pages a day, this is quite a lot of work
    - which is why there are all those server farms for big websites.

AJAX

- Under Ajax, the HTML (or the DOM, really) is being built on the client side, by the Ajax engine.
- Only the "interesting bits" - the things that have changed - need to be built by the server and sent over the network.
- This saves on a lot of resource usage
  - CPU time
  - network bandwidth.
- Again, this is nothing new, but now it's available in normal, standards-compliant web pages
  - rather than through applets or plugins.

AJAX

- This has some big implications for web application development.
- It promotes a clean separation between the interface and the application.
- However, providing cleaner interfaces between data and interface raises some other issues.
- Scrap screen-scraping to extract data from web pages
  - Allow interested parties to build their own Ajax engines to use various sources of XML-formatted data.
  - Don't like the layout of a website? You can build your own instead.
    - Given the correct licensing of *their* data (and this will be a big problem).
    - This just goes further down the path that Amazon has been pushing, of open web services and easy data availability.

AJAX

- Using Ajax also exposes the DOM-rendering Ajax engine to the client.
  - *In addition to exposing the data*.
- Javascript is distributed as source code which contains all the details of how the web page is built.
- This shouldn't be a problem, but it might be seen as one by a lot of businesses.
  - All the interesting stuff is still happening on the server
  - An Ajax engine should be seen as a courtesy by the company, available for free, so that the company's data (their webpage) can be viewed.
- *How will the better availability of data and code, combined with intellectual property issues play out?*

AJAX

- Ajax places larger demands upon the web client
  - something that is usually seen as quite a thin client.
- Handheld and low-power devices may not be able to support Ajax engines well at the moment
  - but in the long term this type of approach has many benefits in this sector.
- Ajax makes for richer clients with more ability to leverage the specifics of the interface
  - E.g., small screens on phones
  - Reduces the amount of bandwidth used
  - These are both issues that have plagued internet use on small-format devices so far.

AJAX – Some Predictions

- New development tools for "Ajax engine" development:
  - better libraries for XML and DOM support in Javascript
  - toolkits built on these that come closer to the server-side templating engines such as PHP
    - Short for "PHP: Hypertext Preprocessor", an open-source, reflective programming language used mainly for developing server-side applications and dynamic web content, and more recently, other software.
- Migration tools to help turn server-side templates into client-side templates
- Faster scripting in browsers - this is already happening
- Renewed interest in networking platforms that are good at dealing with lots of little connections, perfect for Ajax servers, e.g.
  - *Twisted*, an event-driven networking framework written in Python and licensed under the MIT license.
- Greater data-reuse from sites that feed XML
- A move away from the limitations of Javascript, towards a better language or set of languages: the .NET framework and CLR show what this direction could offer.
- *Architectural styles are derived from technology and drive further development.*
  - *Which comes first, the chicken or the egg?*

What is AJAX good for?

- What kinds of applications is Ajax best suited for?
  - We don't know the full range yet.
  - Because this is a relatively new approach, our understanding of where Ajax can best be applied is still in its infancy.
  - Sometimes the traditional web application model is the most appropriate solution to a problem.

Some Example Applications

- **Real-Time Form Data Validation**
  - Form data such as user IDs, serial numbers, postal codes, or even special coupon codes that require server-side validation can be validated in a form before the user submits a form.
- **Autocompletion**
  - A specific portion of form data such as an email address, name, or city name may be autocompleted as the user types.
- **Master Details Operations**
  - Based on a client event, an HTML page can fetch more detailed information on data such as a product listing that enables the client to view the individual product information without refreshing the page.
- **Sophisticated User Interface Controls**
  - Controls such as tree controls, menus, and progress bars may be provided that do not require page refreshes.
- **Refreshing Data on the Page**
  - HTML pages may poll data from a server for up-to-date data such as scores, stock quotes, weather, or application-specific data.
- **Server-side Notifications**
  - An HTML page may simulate a server-side push by polling the server for event notifications that may notify the client with a message, refresh page data, or redirect the client to another page.

Who is Using AJAX Now?

- Google is making a huge investment in developing the Ajax approach.
- All of the major products Google has introduced over the last year are Ajax applications.
  - Orkut
  - Gmail
  - Google Groups
  - Google Suggest
  - Google Maps

Developing with AJAX

- Are Ajax applications easier to develop than traditional web applications?
  - Not necessarily.
  - Ajax applications inevitably involve running complex JavaScript code on the client.
  - Making that complex code efficient and bug-free is not a task to be taken lightly.
  - Better development tools and frameworks will be needed.

Developing with Caution

- Do Ajax applications always deliver a better experience than traditional web applications?
  - Not necessarily.
  - Ajax gives interaction designers more flexibility.
  - However, the more power we have, the more caution we must use in exercising it.
  - We must be careful to use Ajax to enhance the user experience of our applications, *not degrade it*.

Potential Drawbacks

- **Complexity**
  - Server-side developers will need to understand that presentation logic will be required in the HTML client pages as well as in the server-side logic to generate the XML content needed by the client HTML pages.
  - HTML page developers must have JavaScript technology skills.
  - Creating AJAX-enabled applications should become easier as new frameworks are created and existing frameworks evolve to support the interaction model.
- **Standardization of the XMLHttpRequest Object**
  - The XMLHttpRequest object is not yet part of the JavaScript technology specification, which means that the behavior may vary depending on the client.
- **JavaScript Technology Implementations**
  - AJAX interactions depend heavily on JavaScript technology, which has subtle differences depending on the client.
    - See **QuirksMode.org** for more details on browser-specific differences.
- **Debugging**
  - AJAX applications are also difficult to debug because the processing logic is embedded both in the client and on the server.
- **Viewable Source**
  - The client-side JavaScript technology may be viewed simply by selecting View Source from an AJAX-enabled HTML page.
  - A poorly designed AJAX-based application could open itself up to hackers or plagiarism.

Example Dangers

- **Not giving immediate visual cues for clicking widgets.**
  - If something I'm clicking on is triggering Ajax actions, you have to give me a visual cue that something is going on.
  - An example of this is GMail loading button that is in the top right.
    - Whenever I do something in GMail, a little red box in the top right indicates that the page is loading, to make up for the fact that Ajax doesn't trigger the normal web UI for new page loading.

Example Dangers

- **Breaking the back button**
  - The back button is a great feature of standard web site user interfaces.
  - Unfortunately, the back button doesn't mesh very well with Javascript.
  - Keeping back button functionality is a major reason not to go with a pure Javascript web app.

Example Dangers

- **Not using links I can pass to friends or bookmark**
  - Another great feature of websites is that I can pass URLs to other people and they can see the same thing that I'm seeing.
  - I can also bookmark an index into my site navigation and come back to it later.
  - Javascript, and thus Ajax applications, can cause huge problems for this model of use.
  - Since the Javascript is dynamically generating the page instead of the server, the URL is cut out of the loop and can no longer be used as an index into navigation.
  - This is a very unfortunate feature to lose, many Ajax webapps thoughtfully include specially constructed permalinks for this exact reason.

Example Dangers

- **Too much code makes the browser slow**
  - Ajax introduces a way to make much more interesting javascript applications.
  - Unfortunately interesting often means more code running.
  - More code running means more work for the browser.
  - Thus, for some javascript intensive websites, especially poorly coded ones, you need to have a powerful CPU to keep the functionality zippy.
  - The CPU problem has actually been a limit on javascript functionality in the past.
  - Just because computers have gotten faster doesn't mean the problem has disappeared.

Example Dangers

- **Inventing new UI conventions**
  - A major mistake that is easy to make with Ajax is: 'click on this non obvious thing to drive this other non obvious result'.
  - Users who use an application for a while may learn, for example, that if you click and hold down the mouse on a certain widget that you can then drag it and permanently move it to some other place.
  - But since that's not in the common user experience, you increase the time and difficulty of learning your application
    - *a major negative for any application*.

Example Dangers

- **Asynchronously performing batch operations**
  - With Ajax you can make edits to a lot of form fields happen immediately.
  - But that can cause a lot of problems.
  - For example
    - I check off a lot of check boxes that are each sent asynchronously to the server.
    - I lose my ability to keep track of the overall state of checkbox changes.
    - The flood of checkbox change indications will be annoying and disconcerting.