

Asynchronous Javascript + XML (Ajax)

고급 웹 개발응용 실습
제주대학교 컴퓨터공학부

본 내용은 University of Washington, CSE 190 M (Web Programming), Spring 2007 수업에서 사용된 자료입니다.

Except where otherwise noted, the contents of this presentation are © Copyright 2007 Marty Stepp and are licensed under the Creative Commons Attribution 2.5 License.



Web data

- most interesting web pages revolve around data
 - examples: Google, IMDB, Digg, Facebook, YouTube, Rotten Tomatoes
 - can take many formats: text, HTML, XML, multimedia
- today we'll learn ways to connect to web applications that serve data
- we'll also learn the Ajax technique for retrieving and displaying data on our web pages

URLs and web servers

`http://server/path/file`

- usually when you type a URL in your browser:
 - your computer looks up the server's IP address using DNS
 - your browser connects to that IP address and requests the given file
 - the web server software (e.g. Apache) grabs that file from the server's local file system, and sends back its contents to you
- some URLs actually specify *programs* that the web server should run, and then send their output back to you as the result:

`http://science.slashdot.org/article.pl?sid=07/04/20/1651219`

- the above URL tells the server `science.slashdot.org` to run the program `article.pl` with certain parameters

Query strings

`http://www.google.com/search?q=colbert&ie=utf-8`

- query string: a way of encoding parameters into a URL

`http://server/path/program?query_string`

- a query string has the following format:

`field1=value1&field2=value2&field3=value3...`

- preceded by a ?
 - name=value pairs separated by &
- the above URL runs the program `search`, with parameter `q` set to `colbert` and the parameter `ie` set to `utf-8`
 - the program outputs the HTML search results

Web data example

- we have set up a program to retrieve student ASCII animations:
 - the program is called `get_ascii.php`
 - on server `faculty.washington.edu` in folder `/stepp/190m/`
 - accepts required parameter `name` specifying the student's UW NetID
 - accepts optional parameter `file` specifying the student's ASCII animation file name (if no value is passed, uses `asciianimation.txt`).
- what URL will request `essigw`'s animation with default file?
- what URL will request `amylocke`'s animation with file name `asciianimation.txt`?

What is Ajax?

- Ajax: Asynchronous Javascript + XML
- not a programming language; a way of using JS
- a way to download data from a server without reloading your page
- allows dynamically displaying data or updating the page without disturbing the user experience
- aids in the creation of rich, user-friendly web sites
 - the most excellent [CSE 14x Diff Tool](#)
 - other examples: [Google Suggest](#), [Facebook](#), [Flickr](#), [A9](#)

Web applications

- web application: a web site that mimics the look, feel, and overall user experience of a desktop application
 - web app presents a continuous user experience rather than disjoint pages
 - as much as possible, "feels" like a normal program to the user
- some of Google's web apps
 - [Gmail](#), [Google Maps](#), [Google Docs and Spreadsheets](#)
- many web apps use Ajax to battle these problems of web pages:
 - slowness / lack of UI responsiveness

- lack of user-friendliness
- jarring nature of "click-wait-refresh" pattern

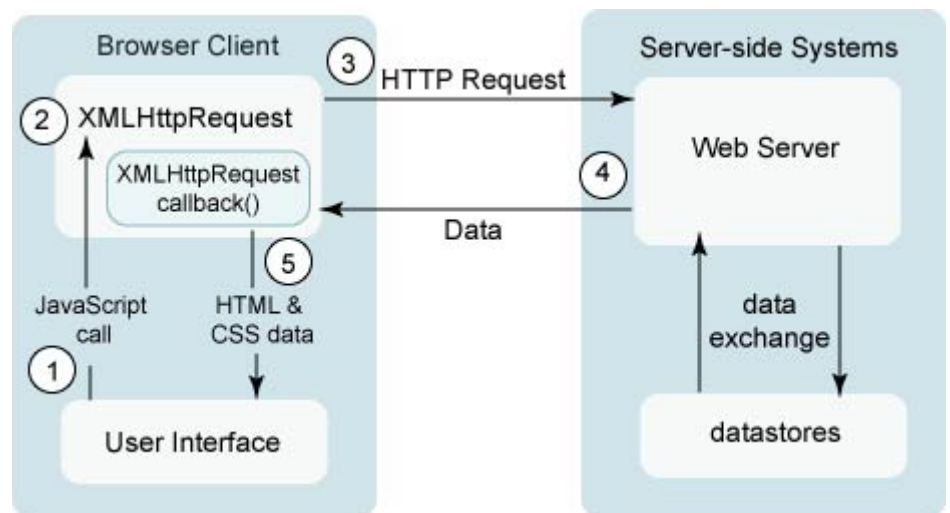
Quick Ajax example

Core Ajax concepts

- Javascript's XMLHttpRequest object can fetch files from a web server
 - supported in IE5+, Safari, Firefox, Opera (with minor compatibilities)
- it can do this asynchronously (in the background, transparent to user)
- contents of fetched file can be put into current web page using DOM
- result: user's web page updates dynamically without a page reload

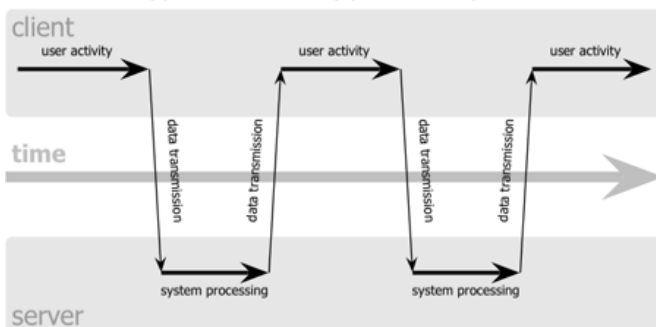
A typical Ajax request

- user clicks, invoking event handler
- that handler's JS code creates an XMLHttpRequest object
- XMLHttpRequest object requests a document from a web server
- server retrieves appropriate data, sends it back
- 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
- your callback event handler processes the data and displays it

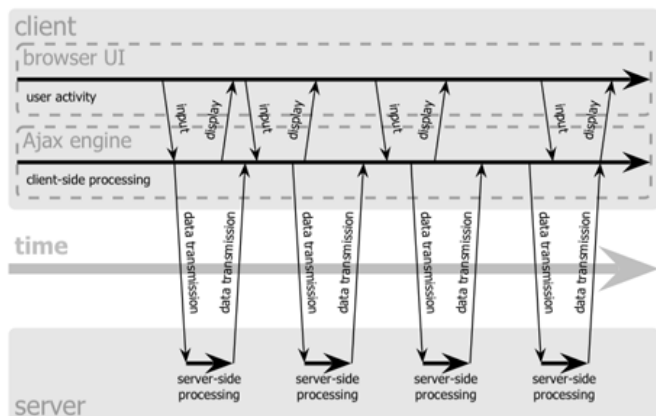


Asynchronous communication

classic web application model (synchronous)



Ajax web application model (asynchronous)



- synchronous: user must wait while new pages load
- asynchronous: user can keep interacting with page while data loads

Ajax communication flow

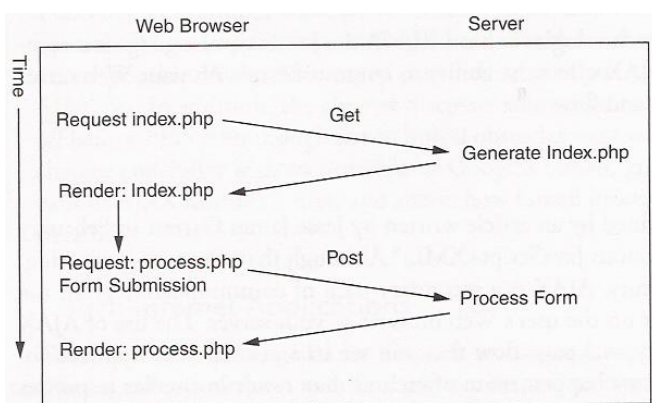


FIGURE 1-1
Web application request flow

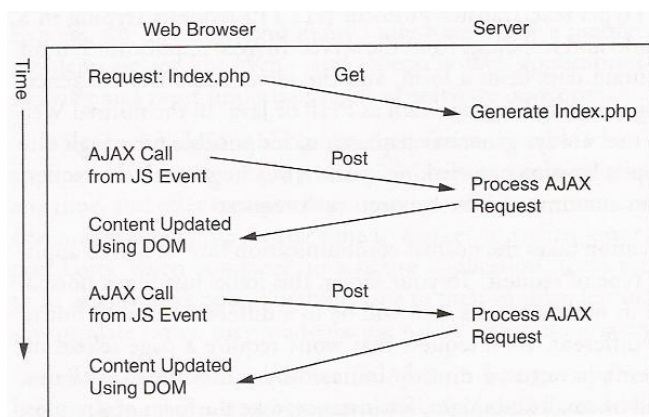


FIGURE 1-2
AJAX application request flow

- Ajax leads to more frequent, smaller communications between browser and server

The XMLHttpRequest object

- methods:
 - abort, getAllResponseHeaders, getResponseHeader, open, send, setRequestHeader
- properties:
 - onreadystatechange, readyState,.responseText, responseXML, status, statusText

Usage of XMLHttpRequest

```
// this code is in some onscreen control's event handler
var ajax = new XMLHttpRequest();
ajax.onreadystatechange = function();
ajax.open("GET", url, true);
ajax.send(null);
```

- attach an event handler to the onreadystatechange event
- handler will be called when request state changes, e.g. finishes
- function contains code to run when request is complete
- replace url with the file you want to download
- IE6 sucks and requires special ActiveXObject code instead

The readyState property

- holds the status of the XMLHttpRequest
- possible values for the readyState property:

State Description

0	not initialized
1	set up
2	sent
3	in progress
4	complete

- readyState changes → onreadystatechange handler runs
- usually we are only interested in readyState of 4 (complete)

Ajax XMLHttpRequest template

```
var ajax = new XMLHttpRequest();
ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
        do something with ajax.responseText;
    }
};
ajax.open("GET", url, true);
ajax.send(null);
```

- most Ajax code uses an anonymous function as the event handler
 - a function declared inside another, and not given a name
 - useful because it can access the surrounding local variables (e.g. ajax)

What if the request fails?

```
var ajax = new XMLHttpRequest();
ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
```

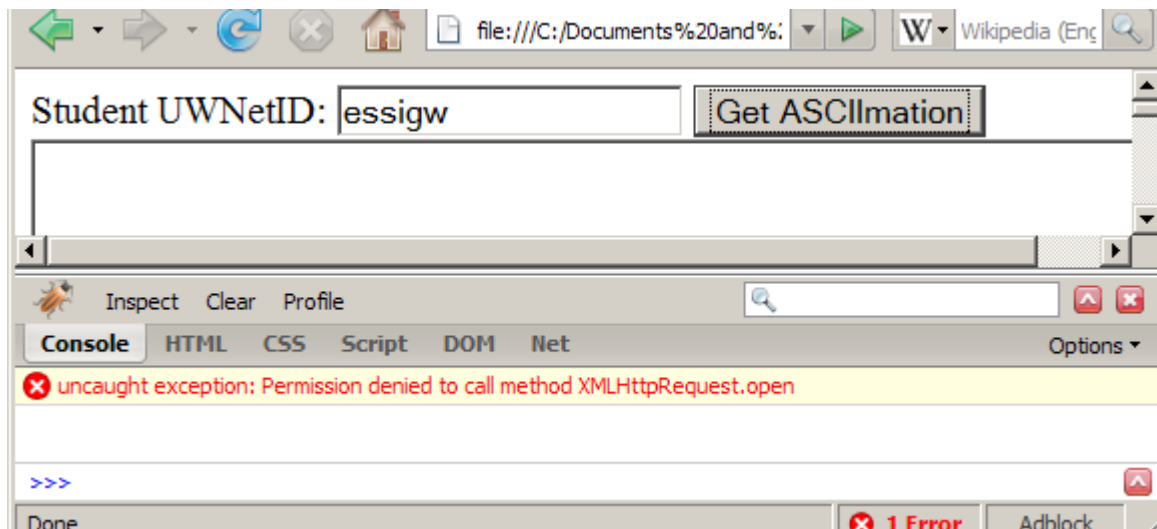
```

    if (ajax.status == 200) {
        do something with ajax.responseText;
    } else {
        code to handle the error;
    }
}
};
ajax.open("GET", url, true);
ajax.send(null);

```

- web servers return status codes for requests (200 means Success)
- you may wish to display a message or take action on a failed request

XMLHttpRequest security restrictions



- cannot be run from a web page stored on your hard drive
- can only be run on a web page stored on a web server
- can only fetch files from the same site that the page is on
 - `www.foo.com/a/b/c.html` can only fetch from `www.foo.com`

Practice problem: ASCIIimation viewer

- Edit the provided `get_ascii` files (HTML, JS) to fetch the student's ASCIIimation program from the server-side `get_ascii.php` program using Ajax.
- Add code to allow a custom file name.
- Add code so that the page gives a good error message if the URL is broken or the file is not found.