An Introduction to AJAX

By : I. Moamin Abughazaleh

How HTTP works?

Classical HTTP Process

1. The visitor requests a page
2. The server send the entire HTML, CSS and Javascript code at once to the client
3. So, the communication is synchronious

What is Javascript programming actually?

What is Javascript programming?

- It is programming the browsers.
- So, we are limited to the objects that the browser presents us

An Alternative for Managing requests - AJAX

- AJAX stands for **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is based on **XMLHttpRequest** object of Javascript - so the browser and
- XMLHttpRequest is a standard
  - http://www.w3.org/TR/XMLHttpRequest/
- It was introduced with IE-5.0 as an ActiveX object (1999)
- Later all the major browsers added XMLHttpRequest into their object bases.

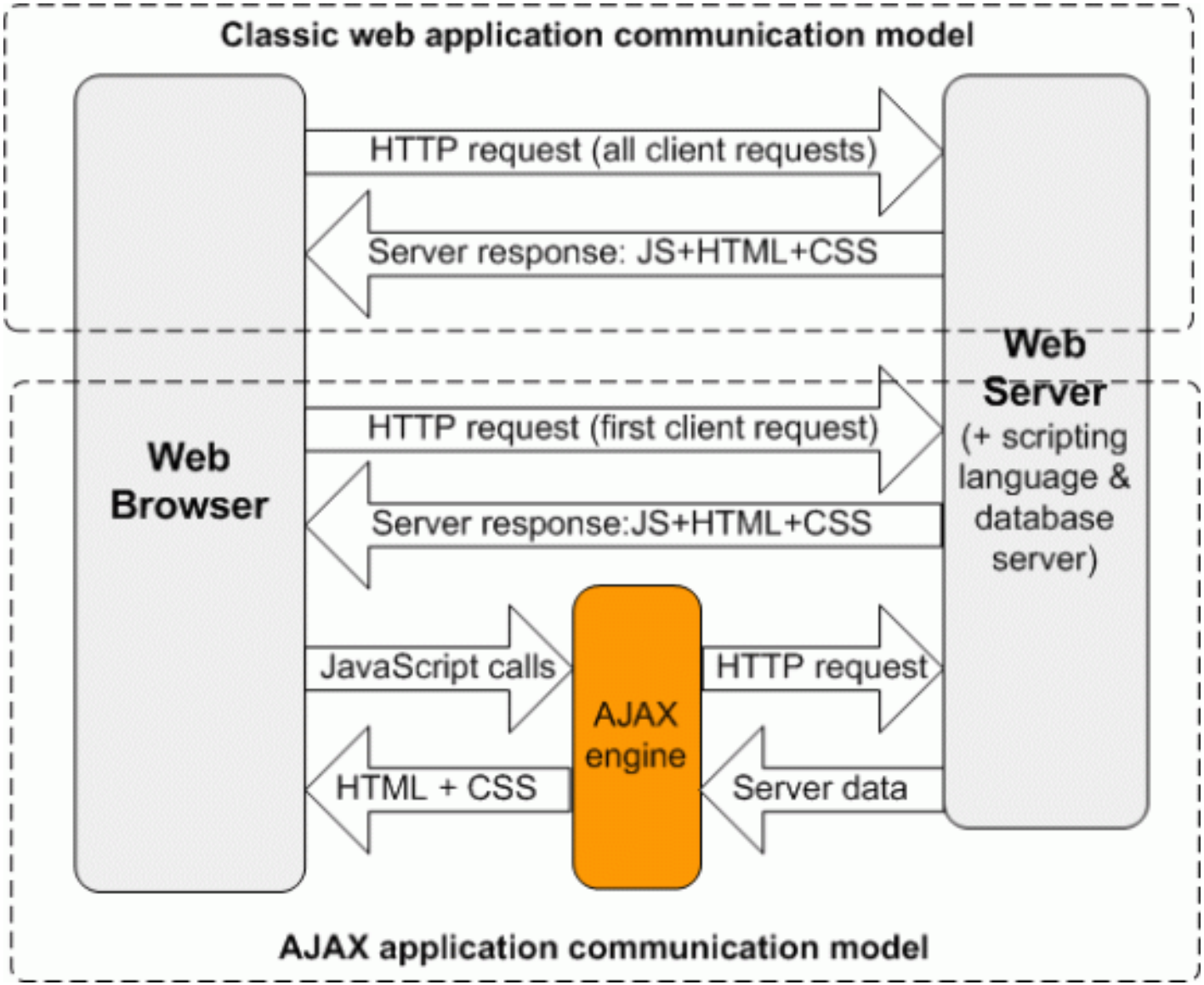
AJAX = Asynchronous JavaScript and XML

- It is a technique for creating better, faster, and more interactive web applications
- With **XMLHttpRequest** object JavaScript can trade data with a web server, without reloading the page
- AJAX uses “asynchronous data transfer” => allowing web pages to request small bits of information from the server instead of whole pages
- We can create desktop application like web applications using AJAX, this paradigm is also called “WEB 2.0” programming

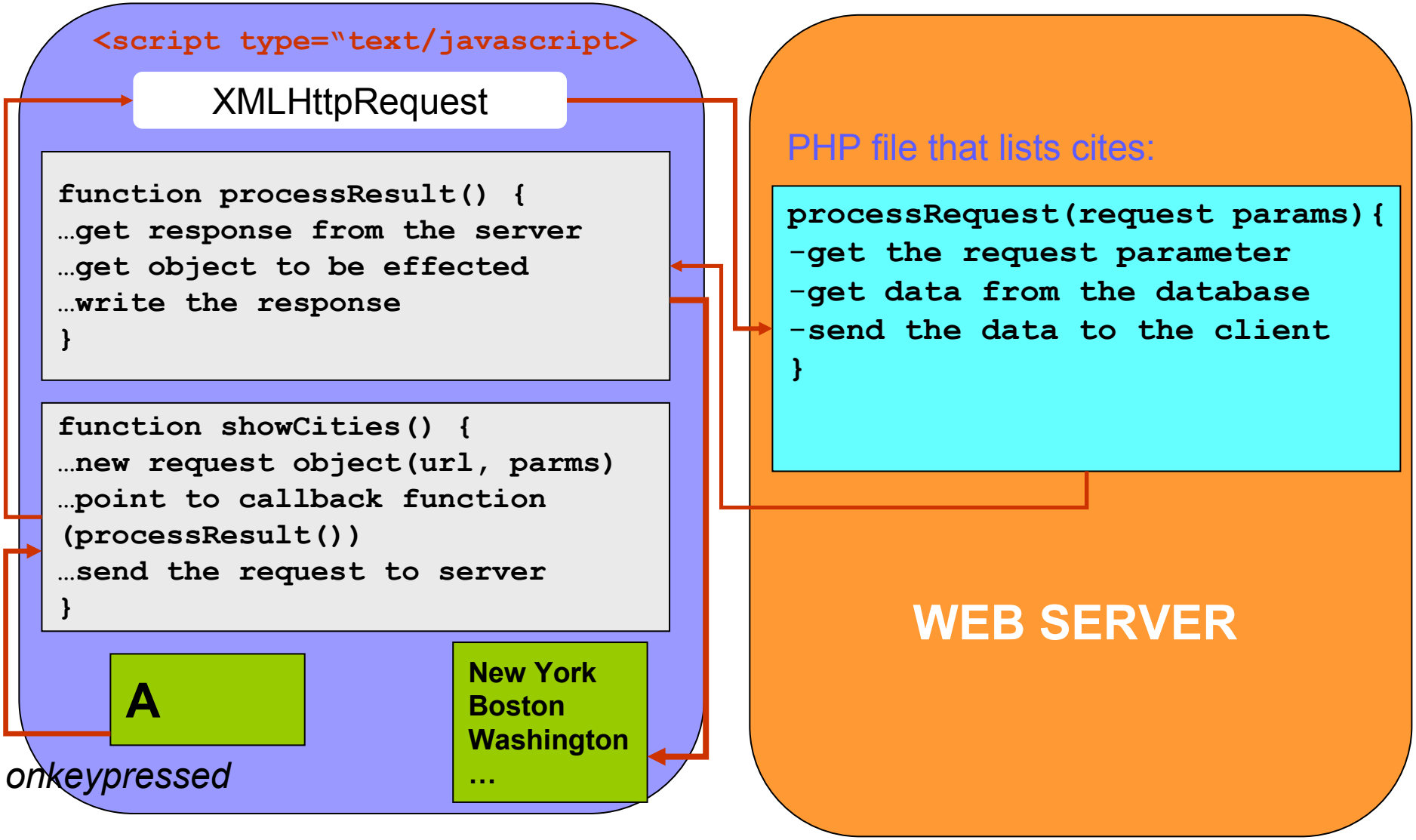
AJAX - Based on Web Standards

- AJAX is based on the following web standards:
  - XHTML and CSS → Presentation
  - DOM → Dynamic display of and interaction with data
  - XML and XSLT → Tranfering data back and forth
  - XMLHttpRequest → Asynchronous transfer of data
  - Javascript → Bring these technologies together
- AJAX applications are browser and platform independent
- The XMLHttpRequest object is supported in Internet Explorer 5.0+, Safari 1.2, Mozilla 1.0 / Firefox, Opera 8+, and Netscape 7.

How does it work?



How does it work? (cont'd)



Advantages

- Rich user experience
- Flexibility and accessibility
- Low bandwidth usage & fast responses
- Different platforms
- Increasing popularity with Web 2.0

Disadvantages

- No browser history or back button, bookmarking
- Still using JavaScript, is JS disabled on the client?
- Search engine optimization problem
- Works only in the current domain
- Hard to debug and write code

Who are using AJAX?

- AJAX was made popular in 2005 by Google (with Google Suggest).
- Facebook
- Gmail
- Google Maps
- and so on…

XMLHttpRequest Object

- Different browsers use different methods to create the XMLHttpRequest object
- Internet Explorer uses an **ActiveXObject**, most other browsers use **XMLHttpRequest**.
- So, in our javascript code we have to deal with the type of the browser when creating the request object

XMLHttpRequest Object(cont'd)

- XMLHttpRequest object is created with Javascript
- There are 3 ways to create this object:
  - For most of the browsers:
    - var request = new XMLHttpRequest();
  - For some IE versions:
    - var request = new ActiveXObject("Microsoft.XMLHTTP");
  - For other versions of IE:
    - var request = new ActiveXObject("Msxml2.XMLHTTP");
- For the best result we shoult create all three

XMLHttpRequest Object Methods

Method	Description
`abort()`	Stops the current request
`getAllResponseHeaders()`	Returns complete set of headers (labels and values) as a string
`getResponseHeader("headerLabel")`	Returns the string value of a single header label
`open("method", "URL"[, asyncFlag[, "userName"[, "password"]]])`	Assigns destination URL, method, and other optional attributes of a pending request
`send(content)`	Transmits the request, optionally with postable string or DOM object data
`setRequestHeader("label", "value")`	Assigns a label/value pair to the header to be sent with a request

XMLHttpRequest Object Properties

Property	Description
`onreadystatechange`	Event handler for an event that fires at every state change
`readyState`	Object status integer: 0 = uninitialized 1 = loading 2 = loaded 3 = interactive 4 = complete
`responseText`	String version of data returned from server process
`responseXML`	DOM-compatible document object of data returned from server process
`status`	Numeric code returned by server, such as 404 for "Not Found" or 200 for "OK"
`statusText`	String message accompanying the status code

The Code (cont'd)

```
<script type="text/javascript">
function getXMLHttpRequest() {
var request;
try {
// Firefox, Opera 8.0+, Safari
request = new XMLHttpRequest();
} catch (e1) {
// Internet Explorer
        try {
        request=new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e2) {
                try {
                request=new ActiveXObject("Microsoft.XMLHTTP");
                } catch (e3) {
                alert("Your browser does not support AJAX!");
                return false;
                }
        }
}
return request;
}
</script>
```

The Code (cont'd)

- After the creation of XMLHttpRequest object with the open method server request should be prepared:

  request.open(method, URL, async)

  - "method" can be 'GET' or 'POST'
  - "URL" determines where the request will be made
    - If GET => parameters will be added to the end of the URL
    - IF POST => parameters will be added in the package
  - "async" (true/false), specifies whether the request should be handled asynchronously or not – true is default

The Code (cont'd)

- After the creation of XMLHttpRequest it should be sent to the server with send() method
  - request.send(null);
    - Used if the method is GET
  - request.send([data]);
    - Used if the method is POST
- Example:

```
request.send('var1=' +value1+'&var2='+value2);
```

The Code (cont'd)

- To get the response message from the server we should determine which method will work when the request object's *onreadystatechange* event is fired.
  - `request.onreadystatechange = someFunction;`
  - ```
    function someFunction() {
     if(request.readyState == 4){
     var response = request.responseText;
     //here do whatever with the response
     }
    }
    ```
    ```

The Code (cont'd)

- onreadystatechange
  - Specifies a reference to an event handler for an event that fires at every state change
- readyState
  - Returns the state of the object as follows:
    - 0 = uninitialized – open() has not yet been called.
    - 1 = open – send() has not yet been called.
    - 2 = sent – send() has been called, headers and status are available.
    - 3 = receiving – Downloading, responseText holds partial data (although this functionality is not available in IE [3])
    - 4 = loaded – Done.
- responseText
  - Returns the response as a string.

The Code - Using GET

```
var url = "get_data.php";
var params = "lorem=ipsum&name=binny";
http.open("GET", url+"?"+params, true);
http.onreadystatechange = function() {
//Call a function when the state changes.
        if(http.readyState == 4 && http.status == 200) {
                alert(http.responseText);
        }
}
http.send(null);
```

The Code - Using POST

```
var url = "get_data.php";
var params = "lorem=ipsum&name=binny";
http.open("POST", url, true);

//Send the proper header information along with the request
http.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
http.setRequestHeader("Content-length", params.length);
http.setRequestHeader("Connection", "close");

http.onreadystatechange = function() {
//Call a function when the state changes.
        if(http.readyState == 4 && http.status == 200) {
                alert(http.responseText);
        }
}
http.send(params);
```

we choose post

We have to set the content header

Not NULL, and parameters aren't seperated with "?"

EXAMPLES