introduction

Thursday, February 16, 2017 10:27 PM

how browsers work

- a browser requests a page
- client requests info from the server
- users sometimes request additional info

without AJAX

- make a request back to server
- receive entire page again
- lots of info you don't need

with AJAX

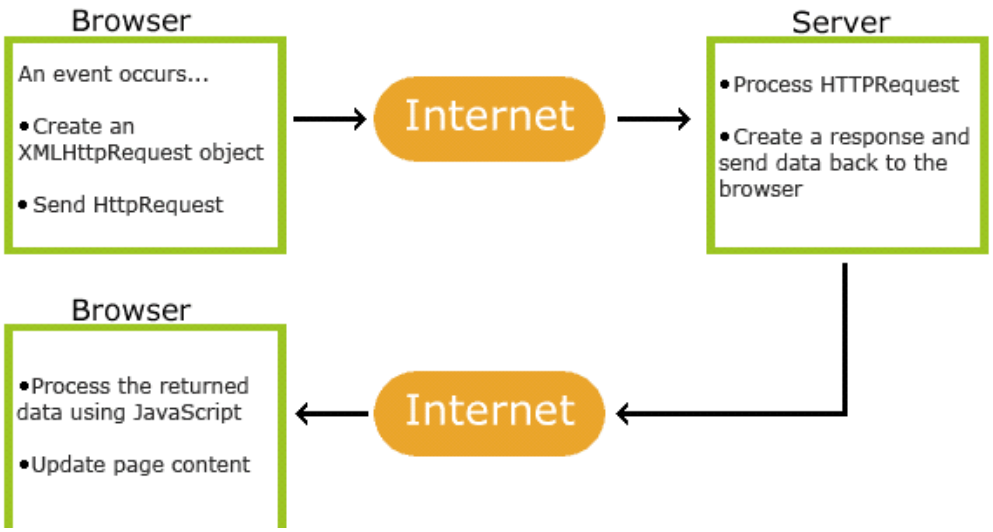
- server doesn't send entire page
- only sends pieces you need
- updates the thumbnails

how AJAX works

- asynchronous JavaScript and XML
- requests without reload
- JavaScript for heavy lifting
- talks to server through XHR

- J stand for JS and this done all things in AJAX
- A set of programming rules(API) is used by JS for communicating b/w Server and client which is XHR API.

How AJAX Works



what XHR isn't

- Not XML
- AJAX request can be in any format
  - Text File
  - HTML
  - JSON object

synchronous XMLhttpRequest

Thursday, February 16, 2017 10:49 PM

- Synchronous means all the request will be done once and then something will returned, or we can say synchronous request are not independent of others.
- A XMLHttp request or XHR object is needed to get information from server or sending information to it.
- Using that object we may be able to contant with the server and get information.

The XMLHttpRequest object is used to exchange data with a server.

Methods of XHR object

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)

- **By default if true/false is not passed in open() it will be taken as true(Asynchronous)**

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.

```
//creating XHR object
var request = new XMLHttpRequest();

//requesting connection, data.txt is a file,
// and we want to get the data from it using AJAX
request.open('GET', 'data.txt', false);
//false is used for synchronous request

request.send();//sending request

if (request.status===200)//if successfull(status code 200)
{
    console.log(request);
    document.writeln(request.responseText);
}
```

IN CONSOLE -

script.js:14

```
▶ XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 0, withCredentials: false,
upload: XMLHttpRequestUpload…}
```

- This is a XMLHttpRequest Object which have many properties,
  we can inspect to know what we really wanted.

```
▼ XMLHttpRequest
    onabort: null
    onerror: null
    onload: null
    onloadend: null
    onloadstart: null
    onprogress: null
    onreadystatechange: null
    readyState: 4
    response: "Hello World"
    responseText: "Hello World"
    responseType: ""
    responseXML: null
    status: 200
    statusText: "OK"
  ▶ upload: XMLHttpRequestUpload
    withCredentials: false
  ▶ __proto__: XMLHttpRequest
```

OUTPUT

Hello World

LIMITATION

- Synchronous means all the request will be done once and then something will returned.
  or we can say Synchronous request are not independent of others.
- So if we have 1000 request it will take time to request all and so nothing will be printed.

Asynchronous XMLhttpRequest

Thursday, February 16, 2017 11:26 PM

Asynchronous - True or False?

To send the request asynchronously, the async parameter of the open() method has to be set to true:

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

Sending asynchronous requests is a huge improvement for web developers. Many of the tasks performed on the server are very time consuming. Before AJAX, this operation could cause the application to hang or stop.

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 when the response ready

Async = true

When using async = true, specify a function to execute when the response is ready in the onreadystatechange event:

Example

```
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();
```

- XMLHttpRequest Object maintains
  - A property 'readyState' which store state status code for a request.
  - A event 'statechange' which will invoke when state of request changes.

```
▼XMLHttpRequest
  onabort: null
  onerror: null
  onload: null
  onloadend: null
  onloadstart: null
  onprogress: null
  onreadystatechange: null
  readyState: 4
  response: "Hello World"
  responseText: "Hello World"
  responseType: ""
  responseXML: null
  status: 200
  statusText: "OK"
▶ upload: XMLHttpRequestUpload
  withCredentials: false
▶ __proto__: XMLHttpRequest
```

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.

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")

The onreadystatechange function is called every time the readyState changes.

When readyState is 4 and status is 200, the response is ready:

- So now we can use this response and work with it, until then we can do some other things.

```
var request = new XMLHttpRequest();
request.open('GET', 'data.txt');
request.onreadystatechange = function() {
    if ((request.readyState===4) && (request.status===200))
        console.log(request);
        document.writeln(request.responseText);
    }
}
request.send();
```

CONSOLE -

```
script.js:5
▶ XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload,
responseURL: "http://localhost/ajax/01-03/workingfolder/ajax/data.txt"…}
```

OUTPUT-

Hello World

Supporting backward compatibility

Friday, February 17, 2017 12:35 AM

- Microsoft initially used 'activeX' technology to built AJAX like functionality first.
- So we just need to check if that browser have 'XMLHTTPRequest' object or it has Microsoft's ActiveXobject.
- All other code will remain the same.

```
var request;
if (window.XMLHttpRequest) {
    request = new XMLHttpRequest();
} else {
    request = new ActiveXObject("Microsoft.XMLHTTP");
}
request.open('GET', 'data.txt');
request.onreadystatechange = function() {
    if ((request.readyState===4) && (request.status===200)) {
        console.log(request);
        document.writeln(request.responseText);
    }
}
request.send();
```