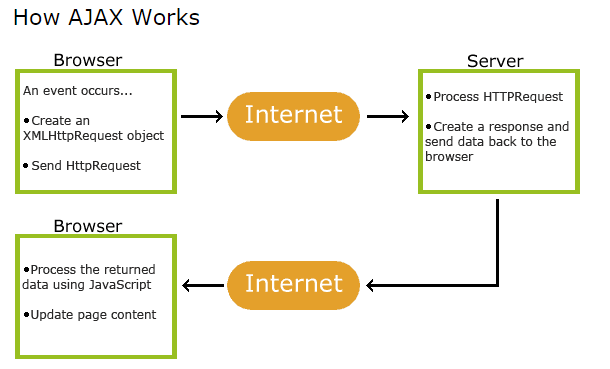
**AJAX** – Asynchronous Javascript and XML (**NOT** A PROGRAMMING LANGUAGE)

* Exchanging data with a server, and updating parts of a web page, without refreshing the browser
* A technique for creating fast and dynamic web pages, not really a technology in itself, just using existing technologies
* Term was coined in 2005 by Jesse James Garrett



* **Google Suggest** uses AJAX to load the dynamic web page as you type in the search engine. The letters you type are captured by the JavaScript on the page and sent off to a server to return list of suggestions.
  + Can be used to pre-fill customer’s name as the user enters it, to see if we have customer information already
    - [**http://www.w3schools.com/ajax/ajax\_aspphp.asp**](http://www.w3schools.com/ajax/ajax_aspphp.asp)
    - **http://www.w3schools.com/ajax/ajax\_php.asp**
* Can query database
  + [**http://www.w3schools.com/ajax/ajax\_database.asp**](http://www.w3schools.com/ajax/ajax_database.asp)
* Application and browser independent
* The XMLHttpRequest object is a key component to AJAX. This object is used to interact with the server behind the scenes
  + New versions of browsers have a built-in XMLHttpRequest object
  + Older versions use an ActiveX object
* PHP generates an XML file, which is then sent to the server

**JavaScript file:**

Var xmlHttp = createXmlHttpRequestObject();

Function createXmlHttpRequestObject(){

//only true if user is using IE

if(window.ActiveXObject){

xmlHttp = new ActiveXObject(“Microsoft.XMLHTTP”);

}

else{

xmlHttp = new XMLHttpRequest();

}

Return xmlHttp;

}

This example uses an onload function

<body onload=process()></body>

function process(){

//state 4 is if communication is over, 0 is if no request has been initialized

if( (xmlHttp.readyState == 0) ||(xmlHttp.readyState == 4) ){

var entered = encodeURIComponent(document.getElementByID().value);

xmlHttp.open(“GET”, “php\_fileName.php?val=”+entered, true);

//might be different for POST

//last argument is if the request is asynchronous or not

//open method creates the request

xmlHttp.onreadystatechange = handleServerResponse;

//handleServerResponse is what changes the web page (handles the request)

xmlHttp.send(null); //this is only null when using GET

}

Else{

//if object is busy, then wait one second and try again

setTimeout(‘process()’, 1000);

}

}

Function handleServerResponse(){

If(xmlHttp.readyState == 4){

//200 status means communication was successful

If(xmlHttp.status == 200){

xmlResponse = xmlHttp.responseXML;

xmlDocumentElement = xmlResponse.documentElement;

//get whatever the php file generated

message = xmlDocumentElement.firstChild.data;

document.getElementByID(“element”).innerHTML = “<> + message”;

//this is so there is time in between requests(so the screen doesn’t change with //every letter you type

setTimeout(“process()”, 1000);

}

}

}

* **When to use GET or POST?**
  + **GET** – displays form data entered in browser URL. For example, action\_page.php?firstname=Mickey&lastname=Mouse
  + Best suited for small amounts of non-sensitive data, since a URL is limited in how long it can be
  + **POST** – if the form is updating data, or includes sensitive information, offers better security since the data is not visible in browser.
* Each input field of a form must have a name attribute

Full-Calendar

* **Call-back** – method that is triggered on an event (when something happens)
  + For example, if the user clicks on a day:

dayClick: function() { alert(“Day clicked”) }

* **Method** – is only triggered when that method is called

DHTMLXScheduler –

<http://docs.dhtmlx.com/scheduler/>

This library may work better since it contains multi-resource view on calendar.

Loading Data

<http://docs.dhtmlx.com/scheduler/loading_data.html>

<http://docs.dhtmlx.com/scheduler/data_formats.html>

Can load data to calendar in 3 forms:

* JSON (similar to what we were doing for full-calendar)
* XML
* ICal

Date Format Spec:

<http://docs.dhtmlx.com/scheduler/settings_format.html>

Units View

<http://docs.dhtmlx.com/scheduler/units_view.html>

<http://docs.dhtmlx.com/scheduler/how_to_start.html>

<http://docs.dhtmlx.com/scheduler/guides.html>

customizations: <http://docs.dhtmlx.com/scheduler/custom_views.html>

<http://docs.dhtmlx.com/scheduler/configuration.html>

May want to customize the y-axis like in this example:

<http://docs.dhtmlx.com/scheduler/samples/02_customization/21_custom_hour_scale.html>

this mimics their current notebook

**JSON**

JavaScript Object Notation

A syntax for storing and transporting data, often used when data is sent to a server.

Data is in key/value pairs separated by commas.

Curly braces hold objects, square brackets hold arrays.

Key/value pair consists of a key, colon, then value

Alternative to XML

JSON evaluates to JavaScript objects

var objects = [

{“firstname”:”John”, “lastname”:”Smith”},

{“firstname”:”Sally”, “lastname”:”Jones”},

];

--One of the methods to pass data and schedule events on the dhtmlx scheduler

Can also modify objects created with JSON:

objects[0].firstname = “Harry”;

Can set a string (var x) to a JSON formatted string, and use the JSON.parse(x) to return the objects which the JSON string creates:

<http://www.w3schools.com/js/tryit.asp?filename=tryjs_json_parse>

--This is opposed to using an XML parser. JSON can be parsed by using a standard JavaScript function

--Note older browsers support the eval() function instead of parse(); however native JSON support is included in modern browsers (IE 8+)

In an AJAX application, JSON is faster than using XML.

When using XML, you need to fetch the XML document, loop through the XML DOM, then store the values in variables. With JSON, you fetch the string and call JSON.parse()