University of Glasgow

AJAX

Web Application Development 2

HYPE, MECHANICS, DEMOS

In 2006, an Amazon.co.uk search for ‘ajax’
under Books / Computer and Internet returned:

52 books

By 2020, an Amazon.co.uk search for 'ajax'
under Books / Computer and Internet returned:

739 books

AJAX origins

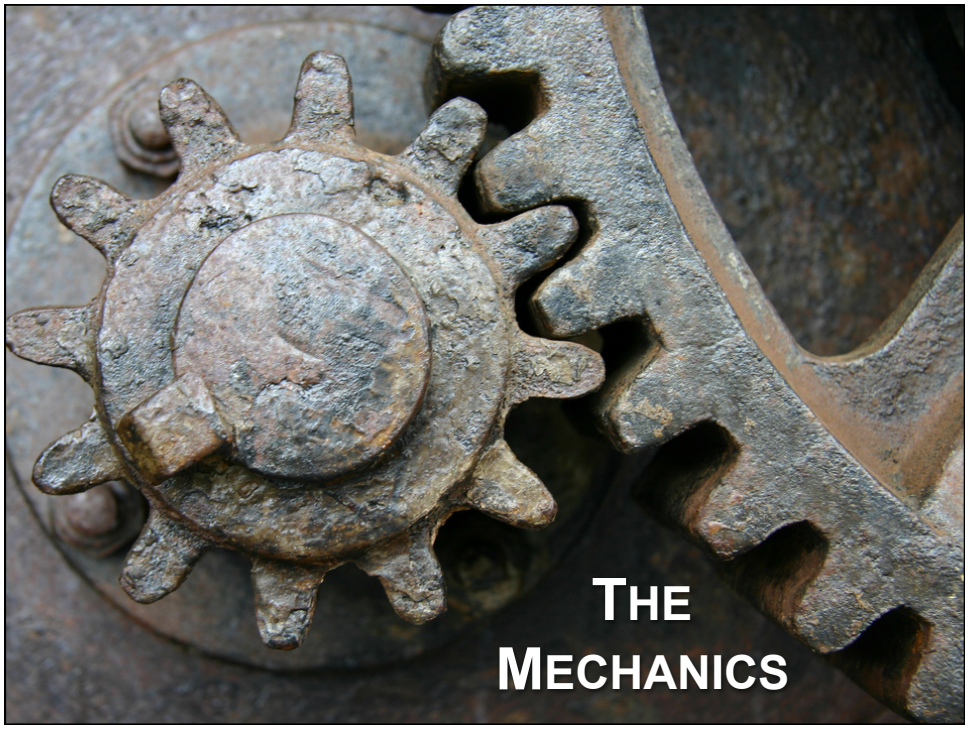
- A key technology set underlying web apps
  - **Asynchronous JavaScript** and **XML**
- Jesse James Garrett coined the term in his 2005 essay:
  - "*Ajax: A New Approach to Web Applications*"
- Critically, all of the components have existed in some form since the late 1990s
  - An example of where browsers introducing non-standard features has been a positive thing
- Generated an enormous amount of hype and energy in web development ever since

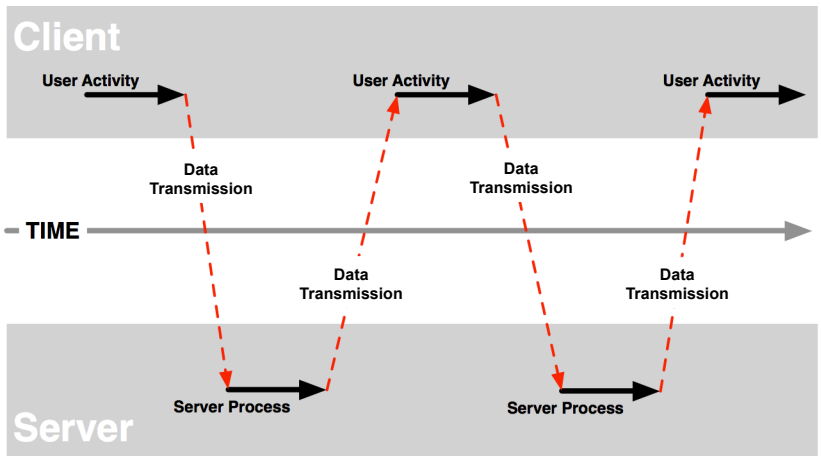
What does AJAX do?

- AJAX **eliminates the need to reload** a web page in order to get new content from the server
- This removes the **start-stop interaction** where a user has to wait for new pages to load
- An intermediate layer (**AJAX Engine**) is introduced into the communication chain between client and server
- Improves the interactive experience in web apps

Technology Set

- Garrett's original technology set:
  - Standards-based presentation using (X)HTML and CSS
  - Dynamic display and interaction using the Document Object Model
  - Data interchange and manipulation using XML and XSLT
  - Asynchronous data retrieval using XMLHttpRequest
  - and JavaScript binding everything together
- Can write in 'pure' JavaScript, jQuery or newer *fetch* API





Traditional Client/Server Synchronous Communication Model

AJAX Components

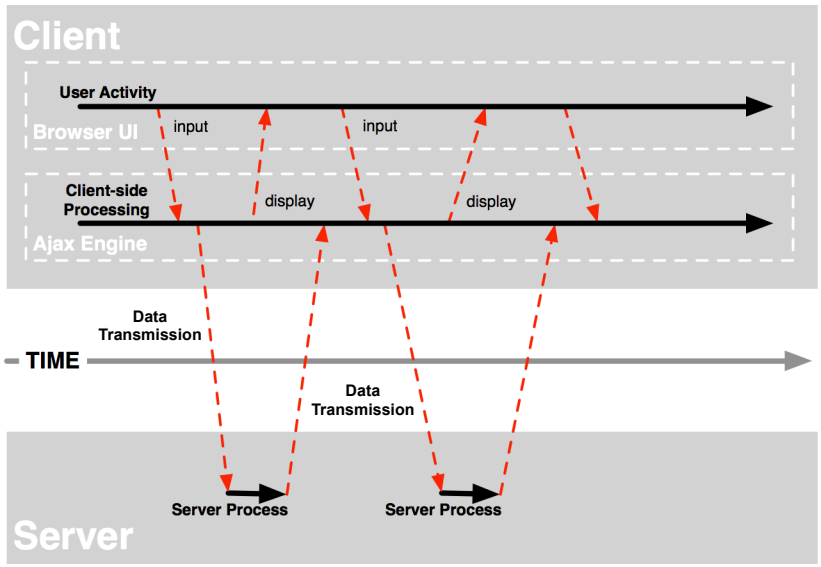
- JavaScript can manipulate the DOM of a webpage to **create, modify and remove content and style**
- JavaScript event handlers can be attached to events generated **by the user and browser**
- XML can **model data** and we can access it using the DOM. (AJAX can also transport JSON or plain text)
- So how does AJAX achieve asynchronous interaction and communication with the server?

XmlHttpRequest Object

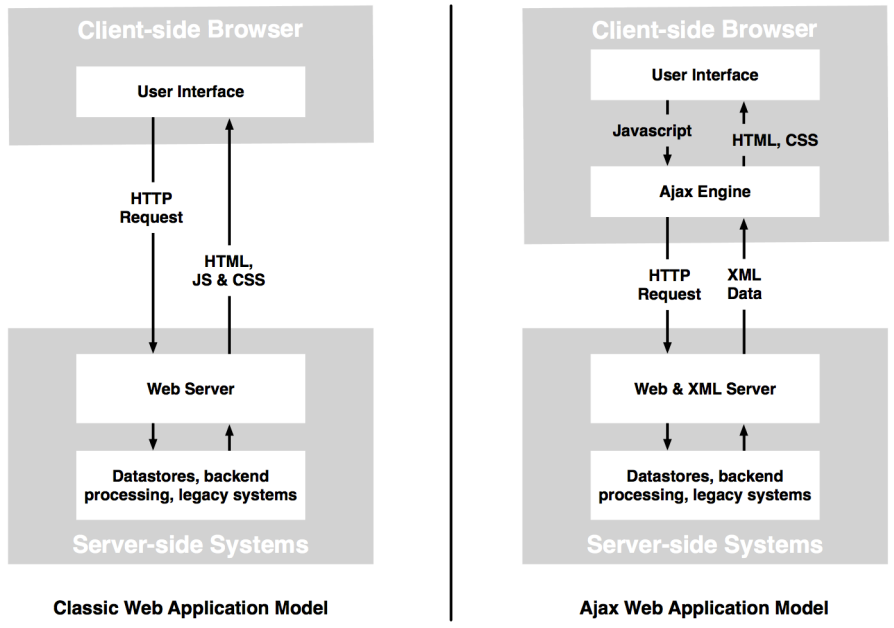
- The keystone of AJAX
- Introduced by Microsoft in Internet Explorer 5
- The XHR is an object that is part of the DOM and is built into most modern browsers
- It can communicate with the server by sending HTTP requests (much like normal client/server communication)

XmlHttpRequest Object

- Independent of <form> or <a> elements for generating HTTP GET/POST requests
- It does not block script execution after sending an HTTP request
- As with **content** and **style**, JavaScript can now programmatically manage **HTTP communication**



AJAX Client/Server Asynchronous Communication Model



Architectural Comparison

How AJAX works

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

XmlHttpRequest Properties

- **readyState** property
  - 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
- **onreadystatechange** property
  - accepts an EventListener value, specifying the method that the object will invoke whenever the readyState value changes
- **status** property
  - The status property represents the HTTP status code and is of type short (e.g. 200 = OK, 404 = Not Found)

XmlHttpRequest Properties

- **responseXML** property
  - represents the XML response data when the complete HTTP response has been received (when readyState is 4), and when the Content-Type header specifies the MIME (media) type as text/xml, application/xml, or ends in +xml
- **responseText** property
  - contains the text of the HTTP response received by the client
  - XML is not the only method to model data in Ajax applications. A popular alternative is JSON (JavaScript Object Notation)

responseXML

- Simple XML to model an address book entry

```
<Person>
    <firstName>John</firstName>
    <lastName>Smith</lastName>
    <age>25</age>
    <address>
        <streetAddress>21 2nd Street</streetAddress>
        <city>New York</city>
        <state>NY</state>
        <postalCode>10021</postalCode>
    </address>
    <phoneNumber type="home">212 555-1234</phoneNumber>
    <phoneNumber type=”mob”>0646 555-4567</phoneNumber>
    <companyName />
</Person>
```

responseText (JSON)

- Same data as previous, but in JSON:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {                                    // address object
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumbers": [                               // array of objects
        { "type": "home", "number": "212 555-1234" },
        { "type": ”mob", "number": ”0646 555-4567" }
    ],
    "companyName": null
}
```

XmlHttpRequest Methods

- **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
- **abort()**
  - Cancels the current request

XmlHttpRequest Methods (cont)

- **setRequestHeader()**
  - Adds a name/value pair to the header to be sent
  - Can be used with POST data to specify the type of data you want to send with the send() method
- **getAllResponseHeaders()**
  - Returns all header information of a resource such as length, server-type, content-type, last-modified
- **getResponseHeader()**
  - Returns specific header information such as “Last-Modified”

AJAX IN ACTION

AJAX Example – HTML header

```
<!DOCTYPE html>
<html>
  <head>
     <title>AJAX example</title>
      <script type="text/javascript">
            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",  "https://www.w3schools.com/js/
ajax_info.txt", true);
                  xhttp.send();
            }
      </script>
  </head>
```

AJAX Example – HTML Body

```
  <body>
    <div id="demo">
        <h2>Let AJAX change this text</h2>
        <button type="button" onclick="loadDoc()">Change Content</button>
    </div>
    <textarea rows="10" cols="50"> Enter text </textarea>
  </body>
</html>
```

- Contents of ajax-info.txt:

```
<h1>AJAX</h1>

AJAX is not a programming language.
<p>
AJAX is a technique for accessing web servers from a web page.
<p>
AJAX stands for Asynchronous JavaScript And XML.
```

Sending a Request

- To send a request to a server in the example, we used the open() and send() methods of the XMLHttpRequest object:
- 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
- Example of a 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();
```

Sending a Request (cont)

- 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();
```

- 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");
```

Responses from the server

- **responseText**: get the response data as a string

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

- **responseXML**: get the response data as XML data

```
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();
```

Processing XML example

cd_catalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
 <CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COMPANY>Columbia</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1985</YEAR>
 </CD>
 <CD>
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <COMPANY>CBS Records</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1988</YEAR>
 </CD>
 <CD>
  <TITLE>Greatest Hits</TITLE>
  <ARTIST>Dolly Parton</ARTIST>
  <COMPANY>RCA</COMPANY>
  <PRICE>9.90</PRICE>
  <YEAR>1982</YEAR>
 </CD>
…
```

Processing XML example

```
<html>  <head>  <title>AJAX example</title>
   <script type="text/javascript">
      function loadDoc() {
            var xhttp, xmlDoc, txt, x, i;
            xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
                  if (this.readyState == 4 && this.status == 200) {
                        xmlDoc = this.responseXML;
                        txt = "";
                        x = xmlDoc.getElementsByTagName("ARTIST");
                        for (i = 0; i < x.length; i++) {
                              txt = txt + x[i].childNodes[0].nodeValue + "<br>";
                        }
                        document.getElementById("demo").innerHTML = txt;
                  }
            };
            xhttp.open("GET", "https://www.w3schools.com/xml/cd_catalog.xml", true);
            xhttp.send();
      }
   </script> </head>
```

Processing XML example

```
<body>
   <div id="demo">
       <h2>Let AJAX change this text</h2>
       <button type="button" onclick="loadDoc()">Change Content</button>
    </div>
    <textarea rows="10" cols="50"> Enter text </textarea>
 </body>
</html>
```

Processing JSON example

myTutorials.json

```
[
{
"display": "JavaScript Tutorial",
"url": "http://www.w3schools.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "http://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "http://www.w3schools.com/css/default.asp"
}
]
```

Processing JSON example

```
<html> <body>   <div id="id01"></div>
<script>
      var xmlhttp = new XMLHttpRequest();
      var url = "https://www.w3schools.com/js/myTutorials.txt";
      xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                  var myArr = JSON.parse(this.responseText);
                  myFunction(myArr);
            }
      };
      xmlhttp.open("GET", url, true);     xmlhttp.send();

      function myFunction(arr) {
            var out = "";
            for(i = 0; i < arr.length; i++) {
                  out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';
            }
            document.getElementById("id01").innerHTML = out;
       }
</script></body></html>
```

Callback functions

- A callback function is a function passed as a parameter to another function
- If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task
- The main function call should contain the URL and which callback function to call when the response is ready

```
loadDoc("url-1", myFunction1);
loadDoc("url-2", myFunction2);

function loadDoc(url, cFunction) {
 var xhttp;
 xhttp=new XMLHttpRequest();
 xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   cFunction(this); }}
  xhttp.open("GET", url, true);
  xhttp.send();
}

function myFunction1(xhttp) {
 // action goes here
}
function myFunction2(xhttp) {
 // action goes here
}
```

Callback functions example

```
<html> <head> <title>AJAX example</title>  <script type="text/javascript">
    function loadDoc(url, cFunction) {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                cFunction(this);
            }
        };
        xhttp.open("GET", url, true);
        xhttp.send();
    }
    function myFn1(xhttp) {
        document.getElementById("demo1").innerHTML = xhttp.responseText;
    }
    function myFn2(xhttp) {
        document.getElementById("demo2").innerHTML = xhttp.responseText;
    }
</script></head>
```

Callback functions example

```
 <body>
  <div id="demo1”>
        <h2>Let AJAX change this text</h2>
        <button type="button" onclick="loadDoc('ajax.txt?t='+Math.random(),
myFn1)">AJAX Info</button>
        <button type="button" onclick="loadDoc('wad2.txt?t='+Math.random(),
myFn2)”>WAD2 Info</button>
  </div>

  <div id="demo2”>
  </div>

 </body>
</html>
```