CS1520 Recitation: Ajax

Jeongmin Lee
Plan for Today

- Introduction
- How does it work
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
Plan for Today

- Introduction
- How does it work
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
AJAX

- Asynchronous JavaScript And XML
AJAX

- Asynchronous JavaScript And XML
- Allows updating a webpage without reloading it.
AJAX

- **Asynchronous JavaScript And XML**
- Allows **updating** a webpage **without reloading** it.
- Only the part to be updated is reloaded with a request and a response.
• **Asynchronous JavaScript And XML**
• Allows **updating** a webpage **without reloading** it.
• Only the part to be updated is is reloaded with a **request** and a **response**.
• It is a combination of built-in **XMLHttpRequest** object (request to server) and Javascript and HTML **DOM** object (display/usage of data).
Plan for Today

- Introduction
- **How does it work**
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
How does it work?

- An event occurs in a web page (user load the page, or a button is clicked)

**Browser**
- An event occurs...
  - Create an XMLHttpRequest object
  - Send HttpRequest

**Server**
- Process HttpRequest
- Create a response and send data back to the browser

**Internet**
- Process the returned data using JavaScript
- Update page content
How does it work?

- An XMLHttpRequest object is created by JavaScript.
How does it work?

- The XMLHttpRequest object sends a request to a webserver.

  - An event occurs...
    - Create an XMLHttpRequest object
    - Send HttpRequest
  
  - Process the returned data using JavaScript
  - Update page content
How does it work?

- The server processes the request

**Browser**
- An event occurs...
  - Create an XMLHttpRequest object
  - Send HttpRequest

**Server**
- Process HttpRequest
- Create a response and send data back to the browser

**Internet**
- Process the returned data using JavaScript
- Update page content
How does it work?

- The server sends a response back to the webpage.
How does it work?

- The response is read by Javascript and proper action is performed by the JS.
Why Ajax?

- Without Ajax: reloading whole page in browser is needed
- With Ajax: not needed!
Plan for Today

- Introduction
- How does it work
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
The XMLHttpRequest object is used to exchange request and response data between a client and a server.

The actual thing you need to create first is the object:

```
variable = new XMLHttpRequest();
```
## Methods in XMLHttpRequest object

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new XMLHttpRequest()</td>
<td>Creates a new XMLHttpRequest object</td>
</tr>
<tr>
<td>abort()</td>
<td>Cancels the current request</td>
</tr>
<tr>
<td>getAllResponseHeaders()</td>
<td>Returns header information</td>
</tr>
<tr>
<td>getResponseHeader()</td>
<td>Returns specific header information</td>
</tr>
<tr>
<td>open(method,url,async,user,psw)</td>
<td>Specifies the request</td>
</tr>
</tbody>
</table>

- **method**: the request type GET or POST
- **url**: the file location
- **async**: true (asynchronous) or false (synchronous)
- **user**: optional user name
- **psw**: optional password
### Methods in XMLHttpRequest object

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>send()</code></td>
<td>Sends the request to the server. Used for GET requests</td>
</tr>
<tr>
<td><code>send(string)</code></td>
<td>Sends the request to the server. Used for POST requests</td>
</tr>
<tr>
<td><code>setRequestHeader()</code></td>
<td>Adds a label/value pair to the header to be sent</td>
</tr>
</tbody>
</table>
Plan for Today

- Introduction
- How does it work
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
Send a Request to Server

- The `open()` and `send()` methods are used to send a request object to server.

- ` xhttp.open("GET", "ajax_info.txt", true) `
Send a Request to Server

- `open(method, url, async)`
  - *Method*: the **type of request** (GET or POST)
Send a Request to Server

- `open(method, url, async)`
  - **Method**: the type of request (GET or POST)
  - **Url**: the server file location that is requested
Send a Request to Server

- `open(method, url, async)`
  - **Method**: the *type of request* (GET or POST)
  - **Url**: the server *file location* that is requested
  - **Async**: true (asynchronous) and false (synchronous)
  - **Asynchronous means you can fetch and reload only the part of the page you want to update.**
Send a Request to Server

- **send()**: sends the request to the server.
  - `send()` *without parameter*: GET
  - `send(“some parameter string”)`: POST
Plan for Today

- Introduction
- How does it work
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
Receive Response from Server

- After sending a request, your request object now has status properties.

- These properties represent the status of the request and you can get the information of whether it is processed well or not.
Receive Response from Server

- **readyState**: the status of the request
  - 0: request not initialized
  - 1: server connection established
  - 2: request received
  - 3: processing request
  - 4: request finished and response is ready
Receive Response from Server

- Status:
  - 200: “Ok” (no problem to get the object)
  - 403: “Forbidden” - not allowed to get the object
  - 404: “Page not found” - probably url is wrong
Receive Response from Server

- Response Type:
  - XMLHttpRequest.responseText
  - XMLHttpRequest.responseURL
  - XMLHttpRequest.responseXML
Receive Response from Server

- **onreadystatechange**: defines a function to be called when state property changes
Plan for Today

- Introduction
- How does it work
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
For example we have this HTML page:

```html
<html>
<body>
<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>
</div>
</body>
</html>
```
For example we have this HTML page

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

With this `loadDoc()` javascript function

```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", "ajax_info.txt", true);
    xhttp.send();
}
```
Let’s focus on loadDoc() function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

1. We created the XMLHttpRequest named `xhttp`
Let’s focus on `loadDoc()` function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

2. Defined a function for **onreadystatechange**, which triggered *only when the readyState changed*. 
Let’s focus on loadDoc() function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

3. The function checks `readyState == 4`.
Code 4 means that request finished and response is ready.
Let’s focus on loadDoc() function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

4. The function checks also `status == 200`. 200 meant “OK” on loading the http object.
Let’s focus on loadDoc() function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

5. Then, it gets ‘this.responseText’ and put the string in the “demo” DOM html object.
- Note that `this.responseText` gets the response data as string. When you want it as XML, you can use `responseXML`
Let’s focus on loadDoc() function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

6. Keep in mind that the loadDoc() function is called only when the readyState is changed.
Let’s focus on `loadDoc()` function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

So when the `loadDoc()` function is called, the `xhttp` request is made first and then the `xhttp.open(..)` and `xhttp.send()` are executed.
Let’s focus on `loadDoc()` function.

```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", "ajax_info.txt", true);
    xhttp.send();
}
```

7. **After** the request is made over `open()` and `send()` function, then when the **response is received** by client browser, the **anonymous function with onreadystatechange** is executed.
Plan for Today

- Introduction
- How does it work
- XMLHttpRequest Object
- Send Request
- Get Response
- Example: Get CD
One more example: Read XML file:

```javascript
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            myFunction(this);
        }
    }
    xhttp.open("GET", "cd_catalog.xml", true);
    xhttp.send();
}

function myFunction(xml) {
    var i;
    var xmlDoc = xml.responseXML;
    var table="<tr><th>Artist</th><th>Title</th></tr>";
    var x = xmlDoc.getElementsByTagName("CD");
    for (i = 0; i <x.length; i++) {
        table += "<tr><td>" + x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue + "</td><td>" + x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue + "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
}
```
1. loadDoc() function is in overall same as previous example.
2. But we have `myFunction(this)` to make it separate of processing XML file stuff.
3. We now open “cd_catalog.xml” file.
The cd_catalog.xml file looks like following. (http://www.w3schools.com/xml/cd_catalog.xml)

```
<CATALOG>
    <script/>
    <script/>
    <div>
        <a id="slick_getbyid_test"/>
    </div>
    <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    <CD>
        <TITLE>Hide your heart</TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>CBS Records</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1988</YEAR>
    </CD>
    ....
```
4. The `myFunction(xml)` function is on reading the xml file, and iterate over each CD object to create the HTML table.

```javascript
function myFunction(xml) {
    var i;
    var xmlDoc = xml.responseXML;
    var table = "<tr><th>Artist</th><th>Title</th></tr>";
    var x = xmlDoc.getElementsByTagName("CD");
    for (i = 0; i < x.length; i++) {
        table += "<tr><td>" +
            x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
            "</td><td>" +
            x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
            "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
}
```
function myFunction(xml) {
    var i;
    var xmlDoc = xml.responseXML;
    var table="<tr><th>Artist</th><th>Title</th></tr>";
    var x = xmlDoc.getElementsByTagName("CD");
    for (i = 0; i < x.length; i++) {
        table += "<tr><td>" +
        x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
        "</td><td>" +
        x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
        "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
}
Questions?