Recitation Week 12

TA: PRANUT JAIN

SLIDES ADAPTED FROM: HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/WEB/GUIDE/AJAX/GETTING_STARTED

HTTP request methods

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource.

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

HEAD

The HEAD method asks for a response identical to that of a GET request, but without the response body.

POST

The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.

HTTP request methods

PUT

The PUT method replaces all current representations of the target resource with the request payload.

DELETE

The DELETE method deletes the specified resource.

OPTIONS

The OPTIONS method is used to describe the communication options for the target resource.

HTTP request methods

An HTTP method is safe if it doesn't alter the state of the server. In other words, a method is safe if it leads to a read-only operation. Several common HTTP methods are safe: GET, HEAD, or OPTIONS.

An HTTP method is idempotent if an identical request can be made once or several times in a row with the same effect while leaving the server in the same state. The GET, HEAD, PUT, and DELETE methods are idempotent, but not the POST method.

What's AJAX?

AJAX stands for **A**synchronous **J**avaScript **A**nd **X**ML

It can send and receive information in various formats, including JSON, XML, HTML, and text files.

AJAX's most appealing characteristic is its "asynchronous" nature, which means it can communicate with the server, exchange data, and update the page without having to refresh the page.

How to make an HTTP request

XMLHttpRequest object used to make request to the server

Its predecessor originated in Internet Explorer as an ActiveX object called XMLHTTP.

```
// Old compatibility code, no longer needed.
if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 6 and older
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

How to make an HTTP request

After making a request, you will receive a response back. At this stage, you need to tell the XMLHttp request object which JavaScript function will handle the response, by setting the onreadystatechange property of the object:

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

For the function, shouldn’t it include ()?

How to make an HTTP request

Another alternative to the function reference?

```
httpRequest.onreadystatechange = function(){
    // Process the server response here.
};
```

How to make an HTTP request

After declaring what happens when you receive the response, you need to actually make the request, by calling the open() and send() methods of the HTTP request object, like this:

```
httpRequest.open('GET', 'http://www.example.org/some.file', true);
httpRequest.send();
```

The first parameter of the call to open() is the HTTP request method – GET, POST, HEAD, or another method supported by your server.

The second parameter is the URL you're sending the request to.

The optional third parameter sets whether the request is asynchronous.

How to make an HTTP request

The parameter to the send() method can be any data you want to send to the server if POST-ing the request. Form data should be sent in a format that the server can parse, like a query string:

"name=value&anothername="+encodeURIComponent(myVar)+"&so=on"

or other formats, like multipart/form-data, JSON, XML, and so on.

Note that if you want to POST data, you may have to set the MIME type of the request. For example, use the following before calling send() for form data sent as a query string:

```
httpRequest.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

Handling the server response

First, the function needs to check the request's state. If the state has the value of XMLHttpRequest.DONE (corresponding to 4), that means that the full server response was received and it's OK for you to continue processing it.

```
if (httpRequest.readyState === XMLHttpRequest.DONE) {
    // Everything is good, the response was received.
} else {
    // Not ready yet.
}
```

Other possible states:

https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState

Handling the server response

Next, check the HTTP response status codes of the HTTP response. The possible codes are listed at the W3C.

```
if (httpRequest.status === 200) {
    // Perfect!
} else {
    // There was a problem with the request.
    // For example, the response may have a 404 (Not Found)
    // or 500 (Internal Server Error) response code.
}
```

Handling the server response

Two options to access that data:

httpRequest.responseText – returns the server response as a string of text

httpRequest.responseXML – returns the response as an XMLDocument object you can traverse with JavaScript DOM functions

If you used a **synchronous** request you don't need to specify a function, but this is highly discouraged as it makes for an awful user experience.

httpRequest.responseText Example

Week12-ex1.html

Week12-ex2.html