AJAX

CSC 342 - Web Technologies

Asynchronous JavaScript And XML (AJAX)

- AJAX allows web pages to be updated asynchronously by making HTTP requests in the background
- AJAX facilitates updating parts of a web page without reloading the whole page
- AJAX is a method that relies on the XMLHttpRequest object (to make HTTP requests) and the JavaScript DOM

Typical AJAX Procedure

1. A DOM event is fired
2. An `XMLHttpRequest` object is created as part of the event handler
3. The `XMLHttpRequest` object sends an HTTP request to a server
4. The server processes the request and sends a response back to the browser
5. The response is read by JavaScript and the DOM is updated accordingly

The `XMLHttpRequest` Object

- Methods:
  - `new XMLHttpRequest()`
  - `abort()`
  - `getAllResponseHeaders()`
  - `getResponseHeader()`
  - `open(method, url, async, user, password)`
  - `send()`
  - `send(string)`
  - `setRequestHeader(header, value)`
- Properties:
  - `onreadystatechange`
  - `readyState`
  - `responseText`

XMLHttpRequest Example

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var node = document.getElementById("one");
    node.innerHTML = this.responseText;
  }
};
xhttp.open("GET", "file.txt", true);
xhttp.send();
```

HTTP Requests

- The `open()` and `send()` methods are used to make an HTTP request
- `open(method, url, async, user, password)`
  - *method*: type of request (GET or POST)
  - *url*: the URL of the resource
  - *async*: true (asynchronous) or false (synchronous)
  - *user*: optional user name
  - *password*: optional password
- An asynchronous request does not have to wait for the server's response

The send Methods

- The send() method is intended for GET requests or POST requests with no URL query data

```
var xhttp = new XMLHttpRequest();
xhttp.open("GET", "file.txt", true);
xhttp.send();
```

- The send(string) method is used to POST URL encoded data

```
var xhttp = new XMLHttpRequest();
xhttp.open("POST", "login.php", true);
xhttp.setRequestHeader("Content-type",
    "application/x-www-form-urlencoded");
xhttp.send("username=Bob&password=swordfish");
```

The `setRequestHeader` Method

- The `setRequestHeader` is used to add an HTTP request header
- `setRequestHeader(header, value)`
  - *header*: specifies the HTTP header name
  - *value*: specifies the HTTP header value

The `onreadystatechange` Property

- The `onreadystatechange` defines a function to be called when the `readyState` changes
- The `readyState` property contains the status of the `XMLHttpRequest`
  - 0: request not initialized
  - 1: server connection established
  - 2: request recieved
  - 3: processing request
  - 4: request finished and response is ready
- The `status` property contains the HTTP status code
- The `statusText` property contains the HTTP status text

Retrieving the HTTP Response Data

- The `responseText` property contains the response data as a string
- The `responseXML` property contains the response as an XML DOM object
- The `getResponseHeader(header name)` method returns the value of a specific HTTP header
- The `getAllResponseHeaders()` method returns all of the HTTP response headers

`XMLHttpRequest` Events

- `readystatechange`: the `readyState` property changes
- `loadstart`: progress has begun
- `progress`: in progress
- `error`: progression failed
- `abort`: progression is terminated
- `timeout`: progression is terminated due to preset time expiring
- `load`: progression is successful
- `loadend`: progress has stopped

Abstracting XMLHttpRequests

- `XMLHttpRequest` code typically follows a standard pattern
- That pattern can be abstracted into a helper function to avoid writing the repetitive `XMLHttpRequest` handling code:

```
function getURL(url, callback) {
  var req = new XMLHttpRequest();
  req.open("GET", url, true);
  req.addEventListener("load", function() {
    if (req.status == 200) {
      callback(req.responseText);
    }
  });
  req.send();
}
```