AJAX and REST API

Building Modern Web Applications - VSP2019

Karthik Pattabiraman
Kumseok Jung

What is AJAX?

1. **What is AJAX?**
2. XMLHttpRequest
3. Callbacks and Error Handling
4. JSON

What is AJAX?

- Stands for “**Asynchronous JavaScript and XML**”, but does not necessarily involve XML
- Mechanism for modern web applications to communicate with the server after page load
  - Without refreshing the current page
  - Request can be sent asynchronously without holding up the main JavaScript thread
- Complement of COMET (Server Push)

What is AJAX?: Brief History

- Introduced by Microsoft as part of the Outlook Web Access (OWA) in 1998
- Popularized by Google in Gmail, Maps in 2004-05
- Term AJAX coined around 2005 in an article
- Made part of the W3C standard in 2006
    - Supported by all major browsers today

What is AJAX?: Usage

- Interactivity
  - To enable content to be brought in from the server in response to user requests
- Performance
  - Load the most critical portions of a web-page first, and then load the rest asynchronously
- Security (this is doubtful)
  - Bring in only the code/data that is needed on demand to reduce the attack surface of the web application

XMLHttpRequest

1. What is AJAX?
2. **XMLHttpRequest**
3. Callbacks and Error Handling
4. JSON

XMLHttpRequest: Creating a Request

- `XMLHttpRequest`: Constructor function for supporting AJAX
- `open`: opens a new connection to the server using the specified method (GET or POST) and to the specified URL or resource


```
1  var req = new XMLHttpRequest();
2  req.open("GET", "/example.txt");
3
4
5
6
7
8
9
10
```

XMLHttpRequest: HTTP Methods

- Two popular methods to send HTTP Request
- GET: used to retrieve data from server by client (typically with no side effects), and does not send any additional data to the server
- POST: used to store data from HTML forms on the server (typically with side effects), and sends the form data to the server

XMLHttpRequest: Sending the Request

- `send`: sends the data to the server asynchronously and returns immediately. Takes a single parameter for the data to be sent (can be omitted for GET)


```
1  var req = new XMLHttpRequest();
2  req.open("GET", "/example.txt");
3  req.send(null);    // or simply - req.send();
4  // Returns here right after the send is complete
5
6
7
8
9
10
```

XMLHttpRequest: Registering Callbacks

- Because the `send` returns right away, the data may not be sent yet (as it's sent asynchronously). Also, we have no way of knowing when the server has responded.
- We need to setup a callback to handle the various events that can occur after a send as the `onreadystatechange` function

```
1  var req = new XMLHttpRequest();
2  req.open("GET", "/example.txt");
3  req.onreadystatechange = function() {
4     // triggered whenever ready state changes
5  }
6  req.send(null);    // or simply - req.send();
7  // returns here right after the send is complete
8
```

XMLHttpRequest: Registering Callbacks

- XMLHttpRequest Status Codes
  - UNSENT (0): open has not been called yet
  - OPENED (1): open has been called
  - HEADERS_RECEIVED(2): Headers have been received
  - LOADING(3): Response is being received
  - DONE(4) : Response is done
- Don’t use the direct numerical values in code

Callbacks and Error Handling

1. What is AJAX?
2. XMLHttpRequest
3. **Callbacks and Error Handling**
4. JSON

XMLHttpRequest 1: Old Method (deprecated)

- Check whether the request's state has changed to DONE
- Check if the status of the request is 200 (denotes success in the HTTP protocol)
- Check if response is of a specific type by examining the header
- If all three conditions match, then perform the action on message receipt (e.g., parse it)


```
1  req.onreadystatechange = function() {
2     if (x.readyState == 4 && x.status == 200){
3        // do something with req.responseText
4     }
5  }
```

XMLHttpRequest 2: New Method (recommended)

- Does away with the `onreadystatechange`
  - Triggers different events depending on response
  - Much cleaner but not all browsers support it (yet)
- Events triggered by the XHR2 Model
  - Load: Response was received (does not mean that it was error-free, so still need to check status)
  - Timeout: Request timed out
  - Abort: Request was aborted
  - Error: Some other error occurred

```
1  req.onload = function() {
2     if (req.status == 200){
3        // do something with req.responseText
4     }
5  }
```

Aborting Requests

- A request can be aborted after it is sent by calling the abort method on the request
- Request may have been already sent. If so, the response is discarded
- Triggers the Abort event handler of the request


```
1  req.onabort = function() {
2     console.log("Request aborted!");
3  }
4
5
```

Timeouts

- Can also specify timeouts in the request (though this is not supported by all browsers)
- Set timeout property in ms

```
1  req.timeout = 200;    // 200 ms timeout
2  req.ontimeout = function() {
3     console.log("Request timed out");
4  }
5
```

Errors

- These occur when there is a network level error (e.g., server is unreachable).
- Trigger the error event on the request
- NOT a substitute for checking status codes

```
1  req.onerror = function() {
2     console.log("Error occurred on request");
3  }
4
5
```

Class Activity

- Write the code for a request handler that issues an AJAX request every time you press the OK button, and cancels the last request every time you press the cancel button. Make sure the appropriate error messages are printed if the request times out (after 5 s), and also if the request has an error or is aborted
- Periodically display the list of messages that are “in-flight” from the client to the server

JSON

1. What is AJAX?
2. XMLHttpRequest
3. Callbacks and Error Handling
4. **JSON**

What is JSON?

- Serialization format for representing JavaScript Objects (JavaScript Object Notation)
- Useful for writing JavaScript objects to files, AJAX messages etc.
- Syntax is very similar to JavaScript itself
    - Not all object types are fully supported though

```
'{}'          // Empty object
'{"a":"Hello","b":42,"c":true,"d":null}'   // object
'[]'          // Empty Array
'"hello"'     // string
'2'           // number
'true'        // boolean
'null'        // null

```

How to handle JSON

- `JSON.parse(string)`: converts string to JavaScript (code/data)
- `JSON.stringify(object)`: converts object to JSON notation
- Header must be set to "application/json"

```
1  var req = new XMLHttpRequest();
2  req.open("GET", "http://example.com/api/");
3  req.onload = function() {
4     if (req.status === 200){
5        if (req.getResponseHeader("Content-Type")
6              .includes("application/json")){
7           var result = JSON.parse(req.responseText);
8           // result is now a JavaScript object
9        }
10    }
11 }
12 req.send();
```