Controller/server communication

CPEN320

Controller's role in Model, View, Controller

- Controller's job to fetch model for the view
  - May have other server communication needs as well (e.g. authentication services)
- Browser is already talking to a web server, ask it for the model
- Early approach: have the browser do a HTTP request for the model
  - First people at Microsoft liked XML so the DOM extension got called: **`XMLHttpRequest`**
- Allowed JavaScript to do a HTTP request without inserting DOM elements
- Widely used and called **AJAX** - **A**synchronous **J**avaScript **a**nd **X**ML
- Since it is using an HTTP request it can carry XML or anything else
  - More often used with JSON

XMLHttpRequest

Sending a Request

```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open("GET", url);
xhr.send();
```

Any HTTP method (GET, POST, etc.) possible.

Responses/errors come in as events

XMLHttpRequest: status codes?

Sending a Request

```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open("GET", url);
xhr.send();
```

Any HTTP method (GET, POST, etc.) possible.

Responses/errors come in as events

XMLHttpRequest: status codes?

Sending a Request

```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open("GET", url);
xhr.send();
```

Any HTTP method (GET, POST, etc.) possible.

Responses/errors come in as events

200 OK
request succeeded, requested object later in this message

301 Moved Permanently
requested object moved, new location specified later in this message (Location:)

400 Bad Request
request message not understood by server

404 Not Found
requested document not found on this server

505 HTTP Version Not Supported

XMLHttpRequest

Sending a Request

```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open("GET", url);
xhr.send();
```

Any HTTP method (GET, POST, etc.) possible.

Responses/errors come in as events

Event handling

```
function xhrHandler(event) {
  // this === xhr
  if (this.readyState != 4) { // DONE
      return;
  }
  if (this.status != 200) { // OK
      return; // Handle error ...
  }
  ...
  let text = this.responseText;
  ...
```

XMLHttpRequest event processing

- Event handler gets called at various stages in the processing of the request

0	UNSENT	open() has not been called yet.
1	OPENED	send() has been called.
2	HEADERS_RECEIVED	send() has been called, and headers and status are available.
3	LOADING	Downloading; responseText holds partial data.
4	**DONE**	**The operation is complete**.

- Response available as:

  raw text - `responseText`

  XML document - `reponseXML`

- Can set request headers and read response headers

XMLHttpRequest

Sending a Request

```
xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open("GET", url);
xhr.send();
```

Any HTTP method (GET, POST, etc.) possible.

Responses/errors come in as events

Event handling

```
function xhrHandler(event) {
  // this === xhr
  if (this.readyState != 4) { // DONE?
      return;
  }
  if (this.status != 200) { // OK?
      return; // Handle error ...
  }
  ...
  let text = this.responseText;
  ...
```

Traditional AJAX uses patterns

- Response is HTML

```
elem.innerHTML = xhr.responseText;
```

- Response is JavaScript

```
eval(xhr.responseText);
```

Neither of the above are the modern JavaScript framework way:

- Response is model data (JSON frequently used here)

```
JSON.parse(xhr.responseText);
```

Fetching resources with XMLHttpRequest via REST

- Can encode model selection information in request in:

URL path:

```
xhr.open("GET","userModel/78237489/fullname");
```

Query params:

```
xhr.open("GET","userModel?id=78237489&type=fullname");
```

Request body:

```
xhr.open("POST", url);
xhr.setRequestHeader("Content-type",
        "application/x-www-form-urlencoded");
xhr.send("id=78237489&type=fullname");
```

Other Transports: HTML5 WebSockets

- Rather than running over HTTP, HTML5 brings sockets to the browser
  - TCP connection from JavaScript to backend Web Server - Bidirectional pipes
- Event-based interface like XMLHttpRequest:

```
let socket = new WebSocket("ws://www.example.com/socketserver");
socket.onopen = function (event) {
  socket.send(JSON.stringify(request));
};

socket.onmessage = function (event) {
  JSON.parse(event.data);
};
```

Trending approach: GraphQL

- Standard protocol for backends from Facebook
  - Like REST, server exports resources that can be fetched by the web app
  - Unlike REST
    - GraphQL is a **query language** for APIs and a **runtime** for executing those queries by using a type system you define for the data.
    - Exports a "schema" describing the resources and supported queries.
    - Client specifies what properties of the resource it is interested in retrieving.
    - Unlike REST, which uses multiple endpoints to retrieve different data, GraphQL typically exposes a **single** endpoint.
- Gaining in popularity particularly compared to REST
  - Gives a program accessible backend - **Application Programming Interface (API)**

Questions?