Reminders

- Use the weekly QR code emailed to you. You must do this during class time.
- Quiz 7 – No Quiz on Week 7. Material will be covered in the midterm.
- Reading:
  - Eloquent JavaScript Chapter 14 DOM
  - MDN Introduction to the DOM Section
    - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
  - MDN Learn Forms Section
    - https://developer.mozilla.org/en-US/docs/Learn/Forms
- Midterm Exam will be available by October 12 (covers weeks 1-7)
- Lab 3 – Thursday, October 12, 2023. Due: October 24 or 26?

ITMD 441/541
Web Application Foundations
Week 8

FALL 2023 – OCTOBER 9, 2023

Weekly’s Agenda

- Finish Week 7 DOM slides
- DOM demo
- JSON
- Introduction to AJAX

EXERCISE 1

Use the getElementById method to find the <p> element, and change its text to "Hello".

```
<p id="demo"></p>

<script>

    document.getElementById("demo").innerHTML    = "Hello";

</script>
```

EXERCISE 2

Use the getElementsByTagName method to find the *first* <p> element, and change its text to "Hello".

```
<p id=“demo”></p>

<script>

  document.getElementsByTagName("p")[0].innerHTML  = “Hello”;

</script>
```

EXERCISE 3

Change the text of the first element that has the class name “Test”.

```
<p class="test"></p>
<p class="test"></p>


<script>

        document.getElementsByClassName("test")[0].innerHTML  = “Hello”;

</script>
```

EXERCISE 4

Change the text color of the <p> element to "red".

```
<p id="demo"></p>

<script>

    document.getElementById("demo").style.color = "red";

</script>
```

EXERCISE 5

Use the eventListener to assign an onclick event to the <button> element.

```
<button id="demo">Click me1</button>


<script>
    document.getElementById("demo").addEventListener(“click”, myFunction);
</script>
```

JSON

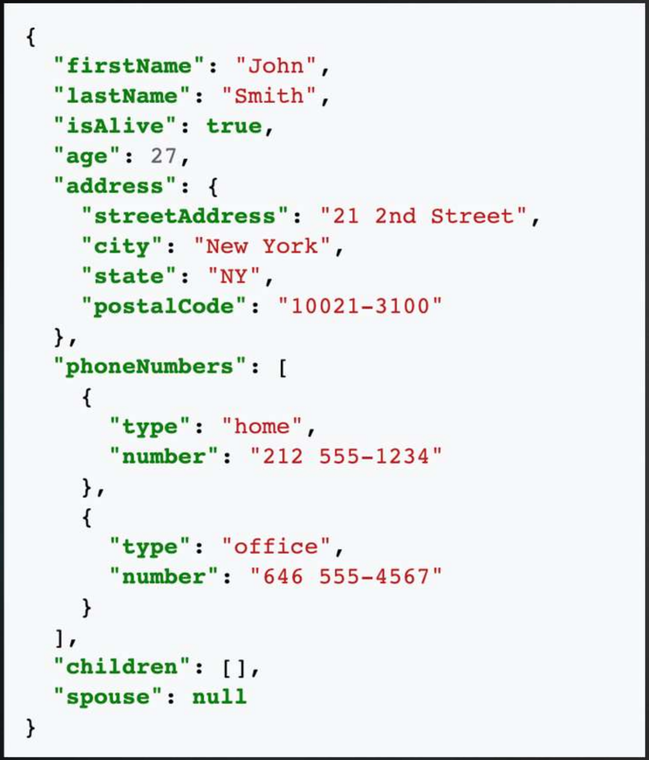
JSON

- **JSON** is **JavaScript Object Notation**
- Lightweight data exchange format that is easily for humans to read or write
- Plain text format that is language independent but was based on a subset of JavaScript language syntax
- Usable in most languages so it is an ideal data-interchange format
- JSON is built on two universal data structures that map to almost all programming languages
  - Collection of name/value pairs
  - Ordered list of values
- Standardized by ECMA International as ECMA-404
  - https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

JSON

- Collection of name/value pairs
  - Realized in languages as object, record, struct, dictionary, hash table, keyed list, or associative array
  - In JSON and JavaScript this is an ***Object***
    - Unordered set of name/value pairs
- Ordered list of values
  - Realized in languages as array, vector, list, or sequence
  - In JSON and JavaScript this is an ***Array***
    - Ordered collection of values
- https://www.json.org/json-en.html
- **SHOW VIDEO**

JSON



```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

- Property names must be double quoted
- String values must be double quoted
- Trailing commas are forbidden
- Numbers can not have leading zeros
- If a number has a decimal point, it must be followed by at least one digit
- NaN and Infinity are not supported
- Values can be:
  - **object**
  - **array**
  - **string**
  - **number**
  - boolean value of **true** or **false**
  - **null**

JSON Example that could represent a person https://en.wikipedia.org/wiki/JSON

JSON

- JavaScript provides a JSON object which contains methods for parsing JSON text to JavaScript objects and converting JavaScript objects to JSON text.
- These methods are available on the JSON object
- `JSON.parse(text)`
  - The parse method will convert the JSON formatted text string that is passed to it to the corresponding JavaScript objects
- `JSON.stringify(value)`
  - The stringify method will convert JavaScript objects to the corresponding JSON formatted text
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

Reminders

- Use the weekly QR code emailed to you. You must do this during class time.
- Week 7 covered on midterm. Week 8 covered with Week 9.
- Reading:
  - Eloquent JavaScript Chapter 14 DOM
  - MDN Introduction to the DOM Section
    - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
  - MDN Learn Forms Section
    - https://developer.mozilla.org/en-US/docs/Learn/Forms
- Midterm Exam will be available today (covers weeks 1-7)
- Lab 3 – Thursday, October 12, 2023. **Due: Sunday, October 29 (NEW DATE)**

Weekly's Agenda

- JSON demo (Intro, Syntax, and JSON.parse())
  - https://www.w3schools.com/js/js_json_intro.asp
- Discuss Midterm
- Discuss Lab 3. New DUE DATE: Sunday, October 29
- Introduction to AJAX delay until next week

EXERCISE JS 1

```
var obj = { color: "red", quantity: 5, instock: true };
```

Command to change obj color to green

```
obj.color = “green";
```

EXERCISE JS 2

```
let x = 200%3;
x += 13
```

What is the value of x?

x is 15

200 divided by 3 is 66 with a reminder of 2
Add 13 to 2 and it is 15

EXERCISE JS 3

```
let x = 10;
if (x > 10) {
    x=x-7;
    console.log(x);
} else if (x > 5) {
    console.log(x+3);
}
```

What is output to the log?

x is 13

EXERCISE JS 4

```
total = 0;
for(let i=0; i<6; i++){
    total = total + i;
}
console.log(total);
```

What is the output?

15

EXERCISE JS 5

Given the following JavaScript array:
myArray = [‘red’, ‘blue’, ‘green’, ‘black’, ‘gray’, ‘orange’];

What is the value of myArray[6]?

Index out of range error

AJAX

AJAX

- AJAX – **A**synchronous **J**avaScript **A**nd **X**ML
- AJAX is not a programming or markup language it is a combination of technologies that were already in use.
  - HTML (or XHTML) and CSS
  - DOM
  - XML or JSON
  - XMLHttpRequest Object
  - JavaScript
- The term was coined by Jesse James Garrett in 2005 when he described the use of these technologies together in an article named *AJAX: A New Approach to Web Applications*

AJAX

- Brief History of AJAX
- It all starts with Microsoft and Internet Explorer
- In 1996 IE introduced the iframe tag which allowed for loading of content asynchronously.
- In 1998 the Outlook Web Access team came up with the concepts and created an ActiveX control XMLHTTP which was released in IE 5 in March 1999
- Mozilla developed an interface that modeled the XMLHTTP object as closely as possible and created a JavaScript object called XMLHttpRequest for their Gecko engine.
  - First available in v0.6 in December 2000 but not fully functional until V1.0 in June 2002
- XMLHttpRequest became a de facto standard in other major web browsers

AJAX

- W3C published a draft of the XMLHttpRequest object specification in April 2006.
- Microsoft adopted the XMLHttpRequest object and added it to IE7 in October 2006.
- This allowed for using the object without platform specific code
- The XMLHttpRequest specification is now a whatwg living standard
- The newer Fetch API specification has been introduced to provide the same functionality with a simplified promise-based API

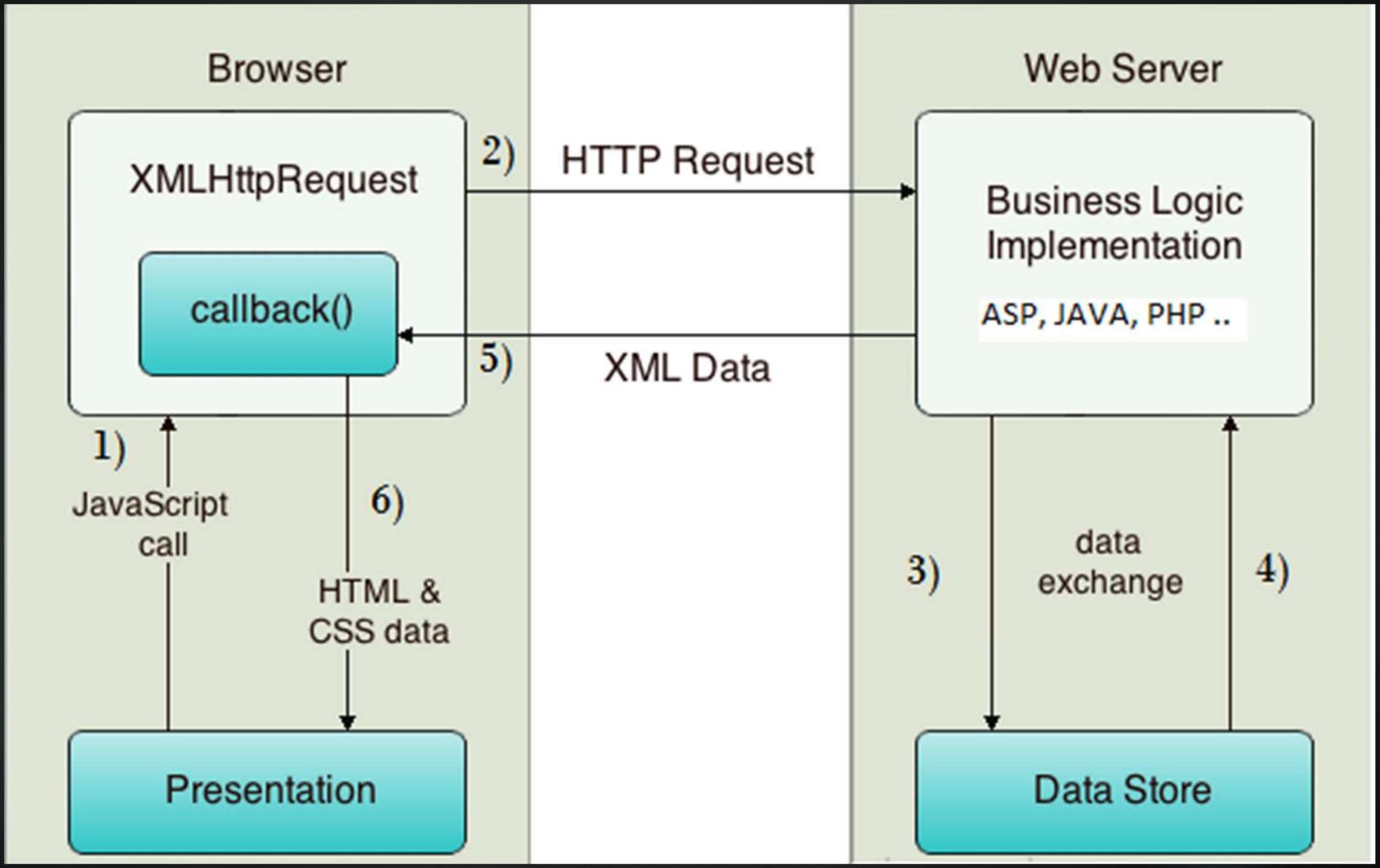
AJAX

- Requests are sent to the server and responses are received asynchronously in the background and processed by JavaScript without a full page load.
- This allows for smaller incremental updates to the UI
- The application/webpage will feel faster and more responsive to the user's interactions
- The X in AJAX stands for XML but it is not required anymore
- JSON is the preferred option now since it has advantages.
  - JSON is part of JavaScript and more lightweight
- There can be various response types including plain text and html
- XML responses require using XLST to process

AJAX Basic Steps

- Event occurs in page that triggers AJAX request (click or something)
- XMLHttpRequest object is created and configured
- The XMLHttpRequest object sends an asynchronous request to the server
- Server processes the request in back-end code
- Server sends a response back to the XMLHttpRequest object including the results as response text. Check readyState and status to see if success.
- The XMLHttpRequest object uses the configured callback function to run and process the results if successful, or proper error response if not
- JavaScript in the callback function updates the HTML, DOM, and CSS of the page as needed

AJAX Flow


Browser
Web Server
XMLHttpRequest
callback()
2)
HTTP Request
Business Logic Implementation
ASP, JAVA, PHP ..
5)
XML Data
1)
JavaScript call
6)
HTML & CSS data
3)
data exchange
4)
Presentation
Data Store

AJAX

- XMLHttpRequest (XHR)
  - http://en.wikipedia.org/wiki/XMLHttpRequest
  - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
  - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
- All modern browsers and IE 7 + support the native XMLHttpRequest object.
- For IE < 7 support you would need to use the active x version. (not a problem anymore, if interested just for information there are resources online that talk about this)
- Important Properties and Methods if xhr === XMLHttpRequest
  - xhr.onreadystatechange
  - xhr.readyState
  - xhr.status
  - xhr.responseText
  - xhr.open()
  - xhr.send();
- Ready State is an unsigned int
  - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState
  - Can use constant as comparison instead of the number.
  - XMLHttpRequest.DONE same as 4

AJAX Same-Origin Policy

- Script code must come from same server, domain, subdomain, protocol, port as the ajax call
- http://en.wikipedia.org/wiki/Same_origin_policy
- CORS will allow for cross-origin requests
  - https://enable-cors.org/
  - https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
- Basic CORS support can be achieved as long as the cross-origin resource providing the data sets the proper header
  - Access-Control-Allow-Origin: *

AJAX Basic Example

```
var myRequest = new XMLHttpRequest();

myRequest.onreadystatechange = function(){
  if (myRequest.readyState === 4) {
    if (myRequest.status === 200)
      var myArray = JSON.parse(myRequest.responseText);
      parseData(myArray);
    }
  }
};

myRequest.open('GET', 'http://example.com/scripts/data.php', true);
myRequest.send();

function parseData(arr) {
  console.log(arr);
}
```

AJAX Basic Example 2

```
var myRequest = new XMLHttpRequest();


myRequest.onreadystatechange = function(){
  if (myRequest.readyState === XMLHttpRequest.DONE) {
    if (myRequest.status === 200)
      var myArray = JSON.parse(myRequest.responseText);
      parseData(myArray);
    }
  }
};


myRequest.open('GET', 'http://example.com/scripts/data.php', true);
myRequest.send();


function parseData(arr) {
  console.log(arr);
}
```

AJAX Basic Example 3

```
var myRequest = new XMLHttpRequest();

myRequest.onreadystatechange = function(){
  if (myRequest.readyState === 4 && myRequest.status === 200) {
      var myArray = JSON.parse(myRequest.responseText);
      parseData(myArray);
  }
};

myRequest.open('GET', 'http://example.com/scripts/data.php', true);
myRequest.send();

function parseData(arr) {
  console.log(arr);
}
```

AJAX Events and Monitoring

- The XMLHttpRequest object allows us to listen for events that occur when the request is being processed, including progress, errors, and more.
- Must add before calling open()
- Doesn't work on file:// protocol
- Events:
  - progress
  - load
  - error
  - abort
  - loadend – happens at the end of all three (load, error, abort) events. Can not tell which event happened though. Useful for things that need to happen no matter which event happens.
- An event object is the parameter of the event handler function
- See section on monitoring progress https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest

AJAX Example using event for load

```
var myRequest = new XMLHttpRequest();

myRequest.addEventListener(‘load’, parseData);

myRequest.open('GET', 'http://example.com/scripts/data.php', true);
myRequest.send();

function parseData(evt) {
  console.log(evt);
  var myArray = JSON.parse(evt.target.responseText);
  // code to process the array and modify the DOM
}
```

Fetch API

- Fetch API is a promise based api for doing AJAX requests.
- No support in IE but other modern browsers do support
- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- https://developers.google.com/web/updates/2015/03/introduction-to-fetch

```
fetch('http://example.com/scripts/data.php')
.then(function(response) {
    return response.json();
})
.then(function(myJson){
    console.log(JSON.stringify(myJson));
});
```

AJAX Libraries

- There is Ajax support built-in to most JavaScript libraries including jQuery
- jQuery ajax support is based on $.ajax(); function
- See jQuery docs for API reference
- http://api.jquery.com/
- Using libraries can simplify your use of Ajax
- New native Fetch API is a more modern way to do ajax
- Popular AJAX library is Axios
  - Promise based HTTP client for the browser and node.js
  - https://www.npmjs.com/package/axios

AJAX POST & Data

POST

- Send data to a server typically with a POST request
- Datatype of the Body of the request is indicated by the Content-Type header
- HTML Forms typically submit using a POST request
    - When sending by form the form tags enctype attribute will determine the content type
    - application/x-www-form-urlencoded
        - Keys and values have an = between them
        - Key-value pairs are separated with an &
        - Non-alpha characters are percent encoded so not usable for binary data
    - multipart/form-data
        - Each value is sent as a block of data in the body with a delimiter between them
        - Use when binary data needs to be sent
    - text/plain
- If using AJAX to send a request the body can be of any data type you want
    - application/json is one example for JSON data
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

POST with XMLHttpRequest

- You can use the XMLHttpRequest object's setRequestHeader() method to set headers on the request.
- This is how you would set the Content-Type header to let the server know what the data type of the request body is.
- The XMLHttpRequest object's send() method can take one parameter.
- That parameter is the body of the request
- https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/send

POST with Fetch

- Fetch defaults to a GET request so to send a POST request you need to use the optional init parameter to the fetch() method.
- The options object is the second parameter to the fetch() method and it is an object
- In that object you can set the body, headers, and request method, among other things
- Headers are set by creating a new Headers object and appending headers to it before setting it to the headers key in the options object.
- The body key can be a list of data types but is typically a string of JSON or form-urlencoded data
- https://developer.mozilla.org/en-US/docs/Web/API/fetch

Demo

- Demo showing 4 different AJAX POST requests

AJAX Resources

- https://en.wikipedia.org/wiki/Ajax_(programming)
- https://en.wikipedia.org/wiki/XMLHttpRequest
- https://web.archive.org/web/20150910072359/http://adaptivepath.org/ideas/ajax-new-approach-web-applications/
- https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX
- https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
- https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

Assignments

Reading/Assignments

- Midterm Exam will be assigned Oct 12 and will be due in one week by end of day Wednesday October 18 (covers weeks 1-7)
- Reading:
  - MDN About Ajax
    - https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX
    - https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started
    - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
    - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- Lab 3