JSON

JSON - JAVASCRIPT OBJECT NOTATION

- A set of key/value pairs
- Keys must be stored in quotes
- Values can be Number, String, Boolean, Array, Object or null
- Can be validated at JSONlint.com

```
{
  "firstName": "Jane",
  "lastName": "Smith",
  "address": {
    "streetAddress": "425 2nd Street",
    "city": "San Francisco",
    "state": "CA",
    "postalCode": 94107
  },
  "phoneNumbers": [
    "212 732-1234",
    "646 123-4567" ]
}
```

MOVING JSON TO THE DOM

Start by storing JSON in a variable:

```
var myProfile = {
  "firstName": "Heather",
  "lastName": "Tovey",
  "cats": ["Hops", "Barley"]
};
```

Then, you can access the data stored inside and use it while creating new nodes:

```
var p = document.createElement('p');
p.innerHTML = 'My name is ' + myProfile.firstName + ' ' + myProfile.lastName + '. ';
p.innerHTML += 'My cats are ' + myProfile.cats.join(' and ') + '.';

var body = document.getElementsByTagName('body')[0];
body.appendChild(p);
```

ACTIVITY: JSON

In this exercise, you'll practice using JSON to dynamically build up HTML. The starter webpage already has the videos described in JSON and functions that turn them into an interactive list.

1. Download json_starter.zip from Brightspace. Unzip the files and ensure that index.html works in the browser.
2. Read through the code in `script.js` to make sure you understand how it works.
3. Add an additional video to the JSON, so that you have 4 videos listed.
4. When you click on the video, it shows the video player on the side. Make it so that it also shows an `h2` with the video title above the `iframe`.
5. Add an "author" property to each video in the JSON and display that next to the title.

Bonus: Add a "favorite" property to each video which is either true or false, and use that to decide whether to put a round red border around the video in the list.

COOKIES

COOKIES

As web developers, we can:

- store small amounts of information in a special place on the user's local disk using a cookie
- use these cookies to access data about the user past the first visit

LIMITATIONS OF COOKIES

- You must use document.cookie to write and read cookies. Reading a cookie takes a lot of code.
- The browser limits the amount of cookies it stores and the size they can be.
- Cookies are shared between both the browser and the server so if your server needs a lot of cookies, that leaves you with little to work with.
- Cookies can expire.

WEB STORAGE

WEB STORAGE

- Solves the problems cookies have
- 2 components
  - Session Storage
  - Local Storage
- Stays within the browser and is never transmitted to the server. Storage for JS developers.
- Provides more storage space than cookies.
- Never expires. Remains until you or the user deletes it.

LOCAL STORAGE

- Data stored in key/value pairs.
- Use the localStorage object to set, get, and remove data.

SETTING DATA

```
localStorage.setItem("username", "Janessa");
```

```
localStorage.userName = "Janessa";
```

Why use setItem?

```
localStorage.user name = "Janessa"; // invalid
localStorage.setItem("user name", "Janessa"); //valid
```

GETTING DATA

```
var name = localStorage.getItem("userName");
```

```
var name = localStorage.userName;
```

REMOVING DATA

Remove a key

```
localStorage.removeItem("userName");
```

```
localStorage.userName = null;
```

Remove all keys and values

```
localStorage.clear();
```

IMPORTANT NOTES

- Web storage only stores strings. Keys and their values must be strings.
- Anything that isn't a string will be converted into a string (numbers or objects).

```
localStorage.age = 35;
var age = localStorage.age;
typeof age; // string
```

```
var janeDoe = {
  firstName: "Jane",
  lastName: "Doe",
  age: 35
};

localStorage.person = janeDoe; // nope!
localStorage.person = JSON.stringify(janeDoe); // YEP! Serialize the object.

var savedPerson = JSON.parse(localStorage.person); // Deserialize it.
```

ACTIVITY: CAT WALK, PART 2

- Go back to your cat walk code.
- Modify the code so that it uses localStorage to store the current location of the cat.
- When the page loads, check if the information is stored in localStorage, and if so, set the cat to that location.
- Now modify it to remember the direction the cat is walking in, and remember that upon page load.

Remember: localStorage stores strings.

AJAX

ASYNCHRONOUS JAVASCRIPT AND XML

- Update parts of a webpage without having to reload the entire thing
- The primary method you use to get and send data to APIs in JavaScript

AJ (the code executed asynchronously)

XML distributes data over the internet through browsers

XMLHTTPREQUEST API

- was the working standard for many years
- complicated and annoying for developers

```
function requestListener() {
  var data = JSON.parse(this.responseText);
  console.log(data);
}

function requestError(error) {
  console.log('We have an issue', error);
}

var request = new XMLHttpRequest();
request.onload = requestListener;
request.onerror = requestError;
request.open('get', 'https://jsonplaceholder.typicode.com/users', true);
request.send();
```

FETCH API

- Fetch API is a modern way to Ajax without having to use helper libraries like jQuery
- supported in modern browsers (you can use a polyfill if you need to support old browsers)

Example:

```
fetch('https://jsonplaceholder.typicode.com/users')
.then(function(response) {
  return response.json();
})

.catch(function(error) {
  console.log(error)
});
```

FETCH: STEP 1

```
fetch('https://jsonplaceholder.typicode.com/users')
```

FETCH: STEP 2

```
fetch('https://jsonplaceholder.typicode.com/users')
.then(function(response) {
  console.log(response);
})
```

FETCH: STEP 3

```
fetch('https://jsonplaceholder.typicode.com/users')
.then(function(response) {
  return response.json();
})
```

FETCH: STEP 4

```
fetch('https://jsonplaceholder.typicode.com/users')
.then(function(response) {
  return response.json();
})
.then(function(data) {
  // Finally made it to our data!
  console.log(data)
})
```

SENDING DATA WITH FETCH

- Now that we can get data, let's send data!
- You need 3 options to configure a fetch request to send data

```
fetch('URL GOES HERE', options);
```

3 OPTIONS

1. Set your request method.
    - `POST` - create a resource
    - `PUT` - update a resource
    - `DEL` - delete a resource

2. Set your headers.
    - `'Content-Type': 'application/json'`

3. Set your body.
    - `body: JSON.stringify(content);`

PUTTING IT ALL TOGETHER

```
// Create the content to send
var content = {
  title: "whoa",
  body: "testing",
  userId: 1
};

// Fetch the URL
fetch('https://jsonplaceholder.typicode.com/users', {

// Set your method
method: 'POST',

// Set your headers
headers: {
    'Content-Type': 'application/json'
},

// Set your body
body: JSON.stringify(content)
```

HANDLING ERRORS

1. You tried to fetch a resource that doesn't exist.
2. You're unauthorized to fetch the resource.
3. You entered some arguments incorrectly.
4. The server throws an error.
5. The server timed out.
6. The server crashed.
7. The API changed.
8. ...

HANDLING ERRORS

```
fetch('http://jsonplaceholder.typicode.com/404')
    .then(function(response) {
            return response.json();
    })
    .then(function(data) {
            console.log(data);
    })
```

HANDLING ERRORS

```
fetch('http://jsonplaceholder.typicode.com/404')
    .then(function(response) {
            if (!response.ok) {
                throw Error(response.statusText);
            }

            return response.json();
    })
    .then(function(data) {
            console.log(data);
    })
```

HANDLING ERRORS

```
fetch('http://jsonplaceholder.typicode.com/404')
    .then(function(response) {
            if (!response.ok) {
                throw Error(response.statusText);
            }

            return response.json();
    })
    .then(function(data) {
            console.log(data);
    })
    .catch(function(error) {
            console.log(error);
    })
```

ACTIVITY: GITHUB API

1. Explore the Github API
2. Use the Fetch API to get information about your Github profile and console.log it
    - User profile image
    - list of repositories
    - name
    - blog
    - url
    - etc.

URL for a specific user

- https://api.github.com/users/YOURUSERNAME

URL for a user's repositories

- https://api.github.com/users/YOURUSERNAME/repos