JavaScript

JS Resources

JaSON

To Review

https://www.youtube.com/watch?v=9YkUCxvaLEk - Wes Bos, Async Await

Classes

- Get and JsSet
- Super ?? and Extends ??

Objects

- Object calls are the same as others (Efforts/Simmering/Exam Prep/Java/Java), `#_dotNotation` & `#bracketNotation`
- Base concepts are the same.
- Inheritance is done with Prototype Inheritance
  - *Every Object has a Prototype Parent, this can be accessed using Object.getPrototypeOf(), the **return** will output the parent*
  - use of *._proto_* is a bad idea, and it has been deprecated, Prototype is the same as above.
- Can use Destructuring

Arrays

- Array Methods
  - the reason you can access methods like pop() and push() in JS, is because of Prototype Inheritance that links directly to the base array.prototype which has the methods built in. Array.prototype links to its parent, the Object.prototype
- Can use Destructuring

Async Code

- Callback Functions
- Promises
- API

Storage

- Types
  - *Cookies*
    - max value is 4kb
    - are server side
    - can incur overhead if overused
  - *Local Storage*
    - uses **setItem()**, **removeItem()**, **clear()** and **getItem**()
    - stored on users PC
  - *Session Storage*
    - limited to the current session, will disintegrate if session closed
    - uses **setItem()**, **removeItem()**, **clear()** and **getItem**()
  - *Indexed Database API*
    - is a database on the browser
      - has limitations
        - internationalized sorting os not supported by the api
        - synchronization is not instant and automatic
        - has limited commands, like **LIKE**
      - operating
        - initing a db, and opening a database both use **.open()**
        - **open()** returns an object named request that provides *success* or *error*
          - case *success*:
            - run code with **onsuccess**()
          - case *error*
            - error code with **onerror**()
        - creating an **objectStore()**
          - accept the store name and the *parameter object* that contains the metadata instructions
            - *autoIncrement()* would be in the parameter object as an id placement for auto indexing
        - ***transactions()***
          - is an object that accepts 2 parameters
          - 2nd is optional
          - transaction(*object list to be transacted*, *type of transaction*)

            1. read-only
            2. read-write
            3. snapshot
        1. **get()** is used to get info
    - Opening a Cursor
        - basically a non path getter from the database

Forms

Client-Side form validation

Attachments

NPM and Webpack

- Intro to Initing a proj Webpack
- Intro to initing node (npm) to a project
    - *for now, just use **npm init -y***

Babel

exists to repurpose your (assumedly new) code into code that is supported by the target browser

you will need to install both babel and a plugin for it to work

How to

https://github.com/babel/babel-loader