Week 7 reading

Ch. 11 Further Functions

* First-class objects – they can be passed around in the say way as every other value
* Functions = first-class objects
* All have length property that returns the number of parameters function has
* Call() – can set value of THIS to an object
  + function sayHello(){
  + return `Hello, my name is ${ this.name }`;
  + }
* Custom properties – description describes what it does
* Self-contained code blocks can be used within it’s own scope
* Functions that define and rewrite themselves – assigning anonymous function to variable that has same name as function
  + function party(){
  + console.log('Wow this is amazing!');
  + party = function(){
  + console.log('Been there, got the T-Shirt');
  + }
  + }
* Init-time branching – create functions that rewrite themselves
* Promise – represents future result of asynchronous operation
  + Helps simplify process
* Promise life-cycle – resolved and rejected
  + Resolved – operation was completed successfully
  + Rejected – didn’t work as expected, wasn’t successfully completed or resulted in an error
* Creating promise – functions called executor
  + const promise = new Promise( (resolve, reject) => {
  + // initialization code goes here
  + if (success) {
  + resolve(value);
  + } else {
  + reject(error);
  + }
  + });
* Then() method – deals with outcome
  + Accepts 2 arguments – fulfillment and promise is rejected
* Can return functions
  + function returnHello() {
  + console.log('returnHello() called');
  + return function() {
  + console.log('Hello World!');
  + }
  + }

Ch 13 AJAX

* Ajax – allows web pages to communicate asynchronously with a server
* API – collection of methods that allows external access to another program or service. Application programming interface
* Fetch – living standard for requesting and sending data asynchronously across network
* Fetch() – returns promise that resolves response returned from URL
* Redirect() – used to redirect to another URL
  + fetch(url)
  + .then( response => response.redirect(newURL)); // redirects to another URL
  + .then( // do something else )
  + .catch( error => console.log('There was an error: ', error))
* Blob() – used to read a file or raw data
  + fetch(url)
  + .then( response => response.blob() ); // transforms the data into a blob object
  + .then( blob => console.log(blob.type) )
  + .catch( error => console.log('There was an error: ', error))
* JSON responses – most common formt for AJAX, transforming a stream of JSON data into a promise that resolves to JavaScript Object
  + fetch(url)
  + .then( response => response.json() ); // transforms the JSON data into a JavaScript object
  + .then( data => console.log(Object.entries(data)) )
  + .catch( error => console.log('There was an error: ', error))