Cheat Sheet – Promises

Promises are a great addition to handle asynchronous tasks/ callbacks. As the name implies, a Promise promises to return a certain value – even if the underlying task fails. In such a failure case, the promise would be rejected but the caller would still be informed.
Therefore, a promise is created with a resolve and reject function being passed as arguments. Depending on the result, the appropriate function is executed and a possible return value is passed as an argument.

```
let promise = new Promise(function(resolve, reject) {
   setTimeout(function() { // setTimeout to simulate async task
      resolve('Done!');
   }, 1000);
});
```

The returned result may then be used in a callback specified in the then() method.

```
promise.then(function(value) {
   console.log(value); // prints Done!
});
```

You may also chain multiple then() calls which will be executed in the order specified and which might also again use promises.

```
let fnWaitASecond = function(secondsPassed) {
   return new Promise(function(resolve, reject) {
      setTimeout(function() {
         secondsPassed++;
         resolve(secondsPassed);
      }, 1000);
   });
};
fnWaitASecond(0)
   .then(fnWaitASecond) // Passes value automatically
   .then(function(seconds) {
      console.log('Promises: waited ' + seconds + ' seconds');
   });
```

Promises also offer some built-in methods you may use to control them.

With race() you may use multiple Promises at once and only use the value returned by the first Promise to resolve. You'll get a value as soon as the first Promise is resolved.
With all() you may use multiple Promises at once and take all returned values into account. Therefore, you'll only get a value once all Promises were resolved.

More information may be found here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise