SUMMARY OF UDACITY COURSE

JAVASCRIPT PROMISES

RATINGS: /5

LESSON 1 – CREATING PROMISES

UNIT 1: Course Introduction

- Every web developer needs to be able handle Asynchronous work with ease, native JS promises

UNIT 2: Callbacks vs Promises

- "The Promise object is used for deferred and asynchronous computations"
- Code is typically synchronous
- Promises is the best way to handle Asynchronous code

UNIT 3: Callbacks vs Thens

- It safe to assume that any operation can crash at any time, so always have an error handling strategy
- Callbacks are used for handling Async tasks but they are not the most efficient way

UNIT 4: Course Map

- Syntax of Promises > Chaining of Promises
- States of a Promises: 1) Fulfilled (resolved) 2) Rejected 3) Pending 4) Settled

UNIT 5: Promise Timeline

- A Promise can only settle once
- Promise are like a "try…catch" wrapper around an Asynchronous code

UNIT 6: Async Scenerios

- You don't gain much by wrapping a Synchronous code in Promises

UNIT 7: Syntax

- The Wrapping Stage of Promises
- Promise is a constructor
- "resolve()" leads to the next "then()" in the chain
- "reject()" leads to the next "catch()" in the chain
- Snap shot

UNIT 8: Write Your First Promise

- Promise syntax: “new Promise (function (resolve, reject) { some async code })”
- Still confusing
- “practice_1.html”

UNIT 9: Wrapping readyState

- Thening stage
- “practice_2.html”

UNIT 10: IMPORTANT! Working with Exoplanet Explorer

UNIT 11: Wrap an XHR

- Worked on “app/scripts/app.js” on the “xhr-start” branch
- Wrap an AJAX request with a Promise
- Adjusted my solution with the course solution
- Check the “xhr-solution” branch for the solution

UNIT 12: Web Technologies

- This courses teaches “native JS Promise”, it is safe to use with most browsers

UNIT 13: Fetch API

- Fetch API simplifies XHR
- Worked on “app/scripts/app.js” on the “fetch-start” branch
- Check the “fetch-solution” branch for the solution

UNIT 14: What Happens Next?

- “Thenable”: you can “.then()” from a “.then”
- Being able to chain “thenables” is a very powerful tool

LESSON 2 – CHAINING PROMISES

UNIT 1: Fetch and Show First Planet

- Promises make it possible to chain together Async Code
- Worked on “app/scripts/app.js” on the “first-thumb-start” branch
- Check the “first-thumb-solution” branch for the solution

UNIT 2: Error Handling Strategies

- “.catch ()” is equivalent to “.then(undefined, rejectFunc)”
- “.then(resolveFunc, rejectFunc)”

- It is recommended to use ".catch()", check out https://jakearchibald.com/2014/resolve-not-opposite-of-reject/

UNIT 3: Chaining Thenables

- Snap shots
- If something goes wrong the next reject function will be called

UNIT 4: Series vs Parallel Requests

- Chaining actions: "action in series" and "action in parallel" each has its advantage
- You cannot predict the time it takes for a request to resolve
- Snap shot

UNIT 5: Array Methods and Promises

- Using a array methods to create a set of promises chained after one another sequencially

UNIT 6: Promises with .forEach

- Refactoring the using ".forEach()"
- Worked on "app/scripts/app.js" on the "foreach-start" branch
- Sometimes you need your code to execute in series and not in parallel
- Fixed my code to make it execute in series

UNIT 7: Promises with .map

- Using .map to fetch all the planets in parallel
- Worked on "app/scripts/app.js" on the "map-start" branch
- Checkout solution on "map-solution" branch
- No guarantee to order when using ".map"

UNIT 8: All Promises

- Using the "Promise.all()" method
- Worked on "app/scripts/app.js" on the "all-start" branch
- Checkout the "Promise.all()" documentation
- Checkout solution on "all-solution" branch

UNIT 9: Course Outro

- Use promises to simplify your life

UNIT 10: Exoplanets 101

- Exoplanets are planets that exist outside of our solar system
- Some stuffs about astronomy

UNIT 11: Bonus Questions: Parallel Requests

- Worked on “app/scripts/app.js” on the “bonus-start” branch