Week 2

ES6 Promises

# What are promises?

Basic understanding of a promise and what it does

1. Promises give us a way to handle asynchronous processing in a more synchronous fashion.
2. let us use our usual language features like return and throw to handle value and error processing in synchronous flows.

# Creating a promise

How to create promises with handlers

#### Promises created using the new Promise constructor and given two handlers: Resolve and Reject

#### Resolve means the promise is fulfilled with the correct value, Reject means it is not

#### Promises have 3 states: Pending, Fulfilled, Rejected

#### You can create an immediately Resolved promise by using Promise.resolve()

# Consuming promises

To consume the Promise - meaning we want to process the Promises value once fulfilled - we attach a handler to the Promise using it's .then() method.

#### .then() takes a function that will be passed the resolved value of the Promise

#### ex: var p = **new** Promise((resolve, reject) => resolve(5));

#### p.**then**((val) => **console**.log(val)); // 5

# Handling errors

To handle errors thrown by Promises, use the .catch() method.

# Composing promises

When working with more than one Promise, you need to start your process once all of them are fulfilled.

#### Using Promise.all() we can take an array of Promises that have been fulfilled, and fulfill it’s returned Promise with an array of fulfilled values. Example below

**var** itemUrls = { '<http://www.api.com/items/1234>', '<http://www.api.com/items/4567>' },

itemPromises = itemUrls.map(fetchJSON);

**Promise**.all(itemPromises) .then(**function**(results) {

// we only get here if ALL promises fulfill results.forEach(**function**(item) {

// process item }); }) .catch(**function**(err) {

// Will catch failure of first failed promise

**console**.log("Failed:", err); });