**The onFulfilled and onRejected Functions**

To handle a “successful” promise, or a promise that resolved, we invoke .then() on the promise, passing in a success handler callback function:

const prom = new Promise((resolve, reject) => {

resolve('Yay!');

});

const handleSuccess = (resolvedValue) => {

console.log(resolvedValue);

};

prom.then(handleSuccess); // Prints: 'Yay!'

Let’s break down what’s happening in the example code:

* prom is a promise which will resolve to 'Yay!'.
* We define a function, handleSuccess(), which prints the argument passed to it.
* We invoke prom‘s .then() function passing in our handleSuccess() function.
* Since prom resolves, handleSuccess() is invoked with prom‘s resolved value, 'Yay', so 'Yay' is logged to the console.

With typical promise consumption, we won’t know whether a promise will resolve or reject, so we’ll need to provide the logic for either case. We can pass both an onFulfilled and onRejected callback to .then().

let prom = new Promise((resolve, reject) => {

let num = Math.random();

if (num < .5 ){

resolve('Yay!');

} else {

reject('Ohhh noooo!');

}

});

const handleSuccess = (resolvedValue) => {

console.log(resolvedValue);

};

const handleFailure = (rejectionReason) => {

console.log(rejectionReason);

};

prom.then(handleSuccess, handleFailure);

Let’s break down what’s happening in the example code:

* prom is a promise which will randomly either resolve with 'Yay!'or reject with 'Ohhh noooo!'.
* We pass two handler functions to .then(). The first will be invoked with 'Yay!' if the promise resolves, and the second will be invoked with 'Ohhh noooo!' if the promise rejects.

Let’s write some onFulfilled and onRejected functions!