Getting comfortable with JS promises

Asa Kusuma

JavaScript Async Basics

- Callbacks do not necessarily imply async
- JavaScript code cannot run concurrently*
- setTimeout(fn,0)/setImmediate/nextTick

Promises are a replacement for:

```
doSomething(function(err, result){…});
```

Why callbacks are less than ideal

Complex flow control is difficult

Once an operation has started, you can't add more handlers

Operations have a single callback and must be directly coupled to all reacting code.

What’s a promise?

A promise is an object that represents a potentially asynchronous operation

A promise is an object with a then() function

The A+ Spec

- 3 states: pending, fulfilled, rejected (settled*)
- .then(), which takes handler functions as params
    - Can call multiple times; attach multiple handlers
    - Handler called even if then() called after settled
- Once fulfilled or rejected, cannot change state

Pending

Fulfilled

Rejected

Settled

onFulfilled()

onRejected()

.then(onFulfill, onReject)

- onFulfill called when promise is fulfilled
- onReject called when promise is rejected

- Handler execution deferred to nextTick
- Handlers have one argument
- Return val becomes promise returned by then()***

Promise lingo

- Handler parameter
  - Fulfillment value
  - Rejection reason
- Resolved == settled

```
fulfilledPromise
    .then(function(){
        console.log(‘world!’);
    });
console.log(‘Hello, ’);
```

Exercise 0:

Hello World

```
myPromise
    .then(function(num){
        return 2 + num;
    }).then(function(num2){
        return 5 + num2;
}).then(printNumber);
```

```
createContext()
.then(fetchUser)
.then(initializeRouters)
.then(initializeBindings)
```

Exercise 1:

Chaining

How are promises created?

Closure Syntax

```
var myPromise =
    new Promise(function(f, r) {
        setTimeout(function() {
            f('Hello world');
        }, 500);
});
```

Deferred Syntax

```
var d = RSVP.defer();
setTimeout(function() {
    d.resolve('Hello world');
}, 500);

var myPromise = d.promise;
```

Deferreds are the read-write parent of the read-only promise

Exercise 2:

Creating Promises

Why promises?

Compose functions without introducing coupling to the functions themselves

Async operations are 1st class citizens, represented as objects

Async or sync?

Promise don’t care

Sane flow control

Use promises to wrap any potentially non-blocking code

Common uses of promises

- Data retrieval
- Anything involving an external call
- Loading CSS
- Animation
- Template rendering

Promise vs Event vs Stream

- Promises are for operations executed once
- Events and streams are for repeated events
- Promises and streams are both object-based

High-order functions

- Functions that input and/or output functions
- Examples
    - _.partial(func, arg)
    - _.bind(func, context)
- Very useful with promises

Promise Libraries

- Promise creation
- Promise inspection
- Utility functions
- Error handling
- Native vs. Library

Utility Functions

- .all()/allSettled()
- .hash()
- .spread()
- .map()
- Static vs member functions

Error handling

- Error in onFulfill will reject returned promise
- Throwing rejects next promise in chain
- Non A+ features
  - Long stack traces
  - Global error listener

Where jQuery will $@#% you

```
$.ajax()
.then(onFulfill, onReject)
.then(output, errorPage);
```

Questions?

+

More Exercises

akusuma@linkedin.com