Promises in Javascript

Three examples of Asynchronous Programming Patterns

About Me

- Tiang (tiangc@gmail.com, @tiang)
- Full Stack Developer for ratemyagent.com.au
- .Net + AngularJS + AWS

Asynchronous

- To be asynchronous is to be in a state of Not Being
- No guarantees when something will happen
- Kinda like Carly Rae Jepsen: “Call Me, Maybe?”
- Promises make sure the right things happen, in the right order, at the right time - Christian Lilley

Schrödinger’s cat
It’s alive and it wants revenge

Schrödinger’s cat
It’s alive and it wants revenge

Javascript Promises

- A Promise is a javascript object where the value is calculated **at a unknown time in the future.**
- We “resolve” the value _when_ an event has occured.
- An event can be “resolved” successfully, or as a failure
- What is an event? AJAX, Closing a Modal, Submitting a form, any user interaction

Declarative Asynch

- Functions can be attached/declared to a promise ***before***, ***during***, or ***after*** the event has occurred
- These functions will run when the promise has been resolved.
- If the function is attached after the promise has been resolved, it will be called immediately.

$http example

- $http.get() returns a Promise
- var Promise = $http.get(‘someURL’);
- Promise.success(function(data) {

```
                    DoSomething();
                });
```

- Promise.error(function() {

```
                    HandleError();
                });
```

Javascript Promises

- Three types of Promise Methods (depends on the libary):
- ***Resolved***: .Then(), .Done(), .Success()
- ***Rejected***: .Error(), .Failure()
- ***Always***: .Always(), Finally()

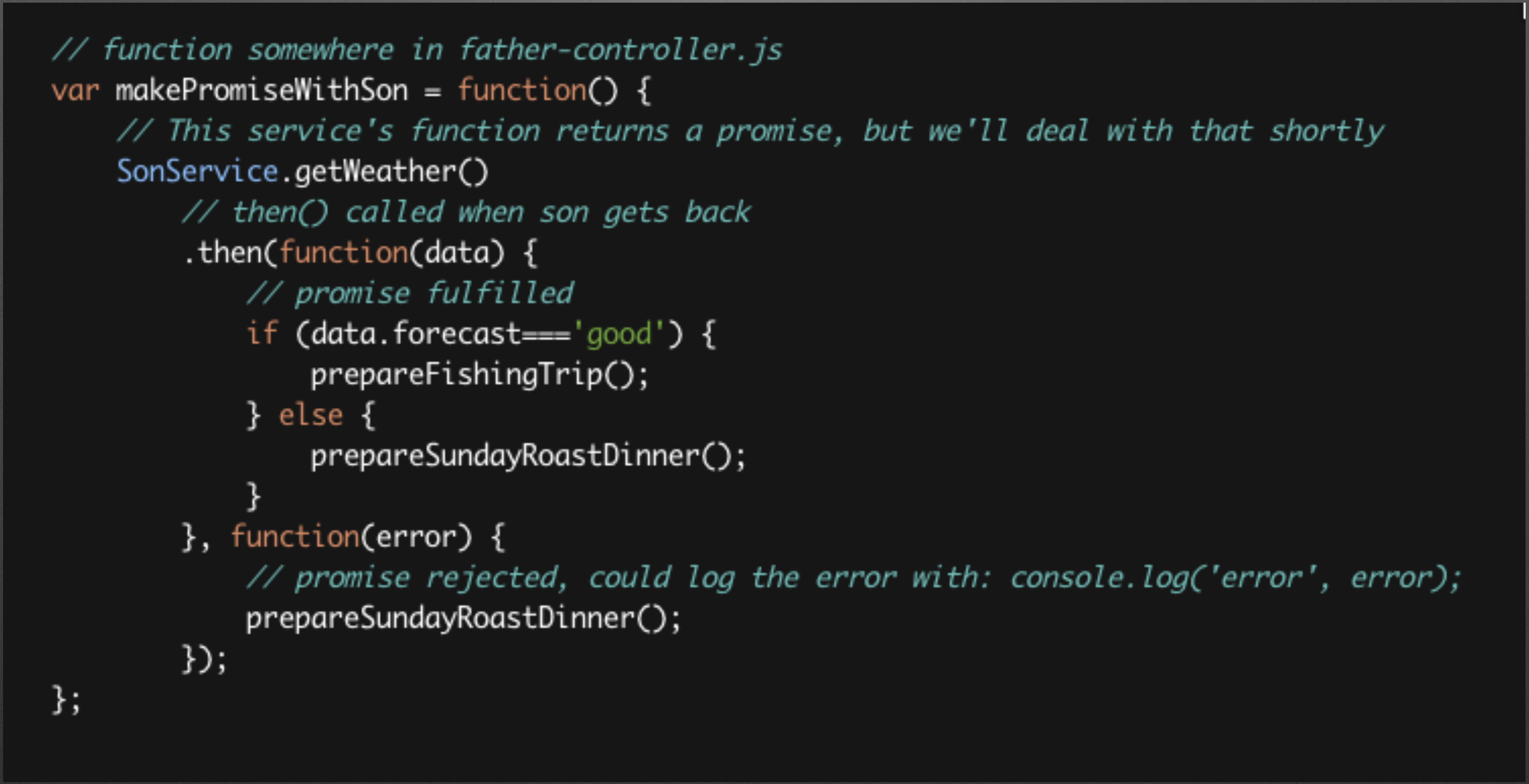
Promises in Angular

- Promises are built into the fabric of AngularJS
- $http, $timeout, $interval are promise objects

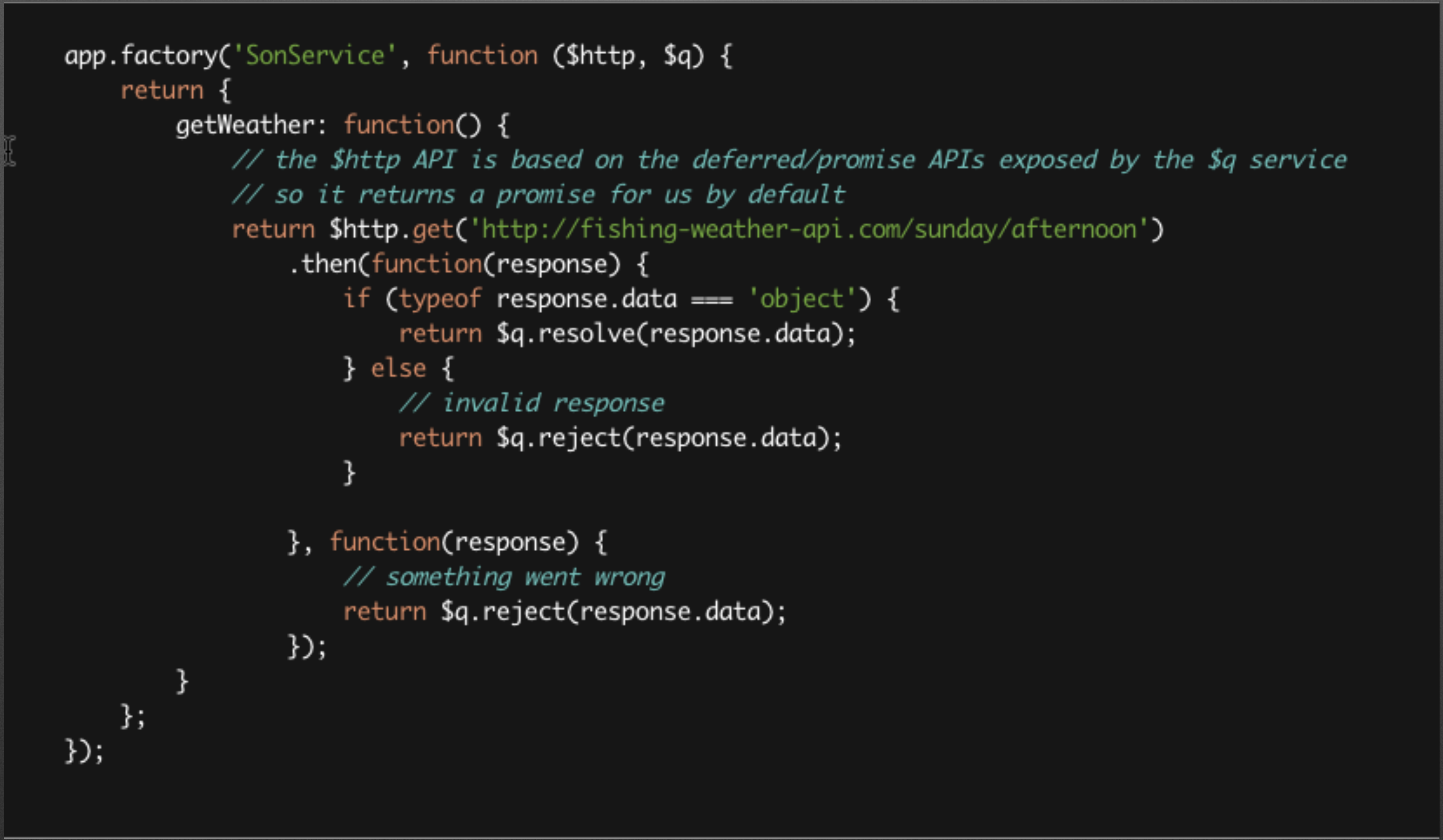
Promises - A Story by Andy Shora

- A Father likes to go fishing, but only if the weather is good.
- He sends his son to the hill to check the weather
- His son promises to return with either good weather, or bad weather
- If weather is good, Father goes fishing, if weather is bad, he stays home

http://andyshora.com/promises-angularjs-explained-as-cartoon.html



```
// function somewhere in father-controller.js
var makePromiseWithSon = function() {
    // This service's function returns a promise, but we'll deal with that shortly
    SonService.getWeather()
        // then() called when son gets back
        .then(function(data) {
            // promise fulfilled
            if (data.forecast==='good') {
                prepareFishingTrip();
            } else {
                prepareSundayRoastDinner();
            }
        }, function(error) {
            // promise rejected, could log the error with: console.log('error', error);
            prepareSundayRoastDinner();
        });
};
```



```
app.factory('SonService', function ($http, $q) {
    return {
        getWeather: function() {
            // the $http API is based on the deferred/promise APIs exposed by the $q service
            // so it returns a promise for us by default
            return $http.get('http://fishing-weather-api.com/sunday/afternoon')
                .then(function(response) {
                    if (typeof response.data === 'object') {
                        return $q.resolve(response.data);
                    } else {
                        // invalid response
                        return $q.reject(response.data);
                    }

                }, function(response) {
                    // something went wrong
                    return $q.reject(response.data);
                });
        }
    };
});
```

Pattern #1: Promise Chaining

- The Promise interface also allows you to chain multiple .done(), .fail(), and .always() callbacks on a single AJAX request
- and even to assign these callbacks after the request may have completed.
- If the request is already complete, the callback is fired immediately.

Promises Demo

jsbin.com/pigix/15/watch

Pattern #2: Route Resolve

- Resolve dependencies BEFORE you instantiate your controller (Dependency Injection)
- Separate your controllers from implementation logic

Pattern #3:
User Uncertainty

- Users are asynchronous.
- They can do things in any order, at any point in time.
- Use Promises to resolve or reject when a user-driven event has occurred.
- Example of user - driven events?

The uncertain web

- Calls to your API
- Waiting for a response from your user
- Events that may occur at any time (login/logout)
- Must Watch: Promises in Angular by Christian Lilley https://www.youtube.com/watch?v=XcRdO5QVlqE

Today's topic

- What is the Promise pattern?
- Using $q
- Real Life examples #1: $http chaining
- Real Life examples #2: Angular Controller Resolving
- Real Life examples #3: User Uncertainty
- Learn THE Design Pattern for the web
  (Ace that interview!)

Thank you!

Leave a review on the meetup.com Group Review section if you had a great time!