

Promises and chaining

Into callback hell ... back again

About Hans

@otype

hans@otype.de



meltwater

```
function Disclaimer() {  
    // Please note ...  
}
```



Intro: sync, async, callbacks



Humans are multi-threaded!

Javascript is single-threaded &
synchronous.

Browsers provide asynchrony*.

start

a()

b()

c()

finish

a()

b()

c()

start

a()

b()

c()

finish

a() b() c() c(),a()

b() a() b() b() ...

c() c() a()



sync



async



Why async?

```
function getRemoteData() {  
    // Long-running call  
}  
  
var remoteData = getRemoteData();  
alert(remoteData);
```

Synchronous flow, here? Bad idea!

“Callbacks to the rescue!”

A close-up photograph of a person's hands holding a gold-colored iPhone. The phone is held horizontally, showing its back side with the camera lens and flash visible. The person is wearing a dark green shirt. The background is blurred, suggesting an indoor setting.

“Call me back!”

```
function getRemoteData(done) {  
  var results = {};  
  // ... long-running call with results ...  
  done(results);  
}  
  
getRemoteData(function (remoteData) {  
  alert(remoteData);  
});
```

Use a callback! Way better!



Got it!

a() → b() → c() → d() → e() → f()

Callbacks in order?

```
a(function (resultsFromA) {  
  b(resultsFromA, function (resultsFromB) {  
    c(resultsFromB, function (resultsFromC) {  
      d(resultsFromC, function (resultsFromD) {  
        e(resultsFromD, function (resultsFromE) {  
          f(resultsFromE, function (resultsFromF) {  
            console.log(resultsFromF);  
          })  
        })  
      })  
    })  
  })  
});
```



"Pyramid of doom!"

```
a(function (resultsFromA) {  
  b(resultsFromA, function (resultsFromB) {  
    c(resultsFromB, function (resultsFromC) {  
      d(resultsFromC, function (resultsFromD) {  
        e(resultsFromD, function (resultsFromE) {  
          f(resultsFromE, function (resultsFromF) {  
            console.log(resultsFromF);  
          })  
        })  
      })  
    })  
  })  
});
```



"I am NOT going to debug that!"

A large, intense fire with bright orange and yellow flames, filling most of the frame. The fire is highly detailed, showing various shades of orange, yellow, and red, with dark smoke rising from the bottom. The texture of the flames is visible, creating a sense of heat and movement.

"Free ticket into callback hell!"

```
95     };
96
97     /**
98      * Retrieve data from REST API, then store into database
99      *
100     * @param companyId
101    * @param done
102   */
103  campaignsService.retrieveAndStore = function (companyId, done) {
104    Log.d('Retrieving Campaigns for companyId=' + companyId + '.');
105
106    /* 1. Get all from DB */
107    DbCampaigns.fetchByCompanyId(companyId, function (dbCampaigns) {
108
109      /* 2. Retrieve campaigns from API */
110      ApiCampaigns.fetchAllByCompanyId(companyId, function (apiCampaigns) {
111        if (apiCampaigns.length !== 0) {
112
113          /* Filter */
114          var filteredCampaigns = campaignsService.filterCampaignLists(
115            campaignsService.createIdList(dbCampaigns),
116            campaignsService.createIdList(apiCampaigns)
117          );
118
119          /* convert the ID lists back into campaign lists with full objects */
120          var dbCampaignsToDelete = campaignsService.createCampaignsListFromIdList(
121            filteredCampaigns.dbCampaignIdsToDelete,
122            dbCampaigns
123          ),
124          apiCampaignsToStore = campaignsService.createCampaignsListFromIdList(
125            filteredCampaigns.apiCampaignIdsToAdd,
126            apiCampaigns
127          );
128
129          if (dbCampaignsToDelete && dbCampaignsToDelete.length > 0) {
130            DbCampaigns.deleteAll(dbCampaignsToDelete, function () {
131              DbCampaigns.storeAll(apiCampaignsToStore, companyId, done, done);
132            }, done);
133          } else {
134            DbCampaigns.storeAll(apiCampaignsToStore, companyId, done, done);
135          }
136        } else {
137          if (done) {
138            done();
139          }
140        }
141      }, done);
142    }, done);
143  };
144
145  /**
```

Core:
Promises to the rescue

*“I hereby give you a promise that I
will answer your question some time
in the future ... but not right now!”*

My definition of a promise

Promise

fulfilled
rejected
pending
settled

kriskowal/q
cujojs/when
tildeio/rsvp.js
WinJS.Promise

Still experimental

```
function process() {  
  var  
    deferred = $q.defer(),  
    results = {};  
  
  // long-running process ...  
  deferred.resolve(results);  
  
  return deferred.promise;  
}
```

\$q.defer()

```
process().then(function (results) {  
  alert(results);  
});
```

.then()

```
processResponse()  
  .then(successCb, errorCb)  
  .catch(/* func */)  
  .finally(/* func */);
```

.resolve()
.reject()

```
getApiData('https://my.api.com').then(function (apiResponse) {  
  processResponse(apiResponse).then(function (results) {  
    alert(results);  
  });  
});
```

.then() in .then() in .then() ...



"I've seen that *\$#%\$* before ..."

```
getApiData()  
    .then(processResponse)  
    .then(storeToDatabase)  
    .then(provideToUI);
```

Chaining



“Wicked, Hans!”

“And what about collecting
results from multiple promises?”

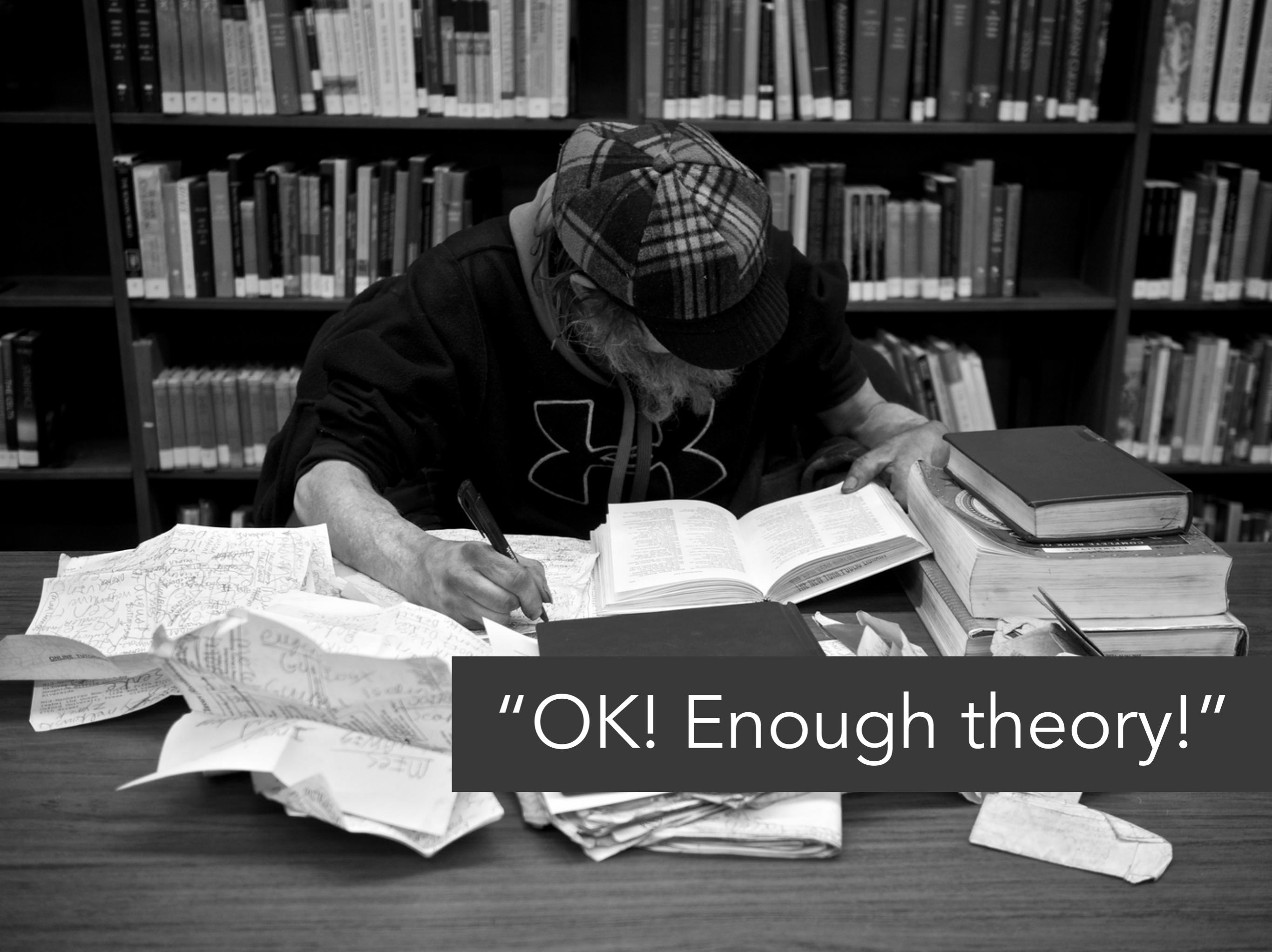
\$q.all()

Combined
results

```
function getAllApiData(urls) {
  var allPromises = [];

  angular.forEach(urls, function (url) {
    var deferred = $q.defer();
    getApiData(url).then(function (response) {
      deferred.resolve(response);
    });
    allPromises.push(deferred.promise);
  });

  return $q.all(allPromises);
}
```



"OK! Enough theory!"

Get fancy:
Demo Time!

Outro:
Questions?

<http://ejohn.org/blog/how-javascript-timers-work/>

<http://www.html5rocks.com/en/tutorials/async/deferred/>

[https://docs.angularjs.org/api/ng/service/\\$q](https://docs.angularjs.org/api/ng/service/$q)

<https://github.com/kriskowal/q>

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

<https://www.promisejs.org/>

<https://github.com/promises-aplus/promises-spec>

A wide-angle photograph of a coastal scene. The foreground is filled with the vibrant turquoise water of the sea. In the middle ground, a long, rocky peninsula or island extends from the left towards the right, its slopes covered in sparse vegetation and small bushes. To the right of the peninsula, a sandy beach curves along the coastline. The background features a range of mountains with rugged, light-colored peaks under a clear blue sky.

“Thank you! Over & out!”