PROMISES PROMISES…

@superstructor (On GitHub / Twitter)

Promises

DEFTJS TEAM

John Yanarella
Universal Mind

Brian Kotek
Booz Allen Hamilton

Isaac Johnston
Superstruct

Ryan Campbell
Universal Mind

David Tucker
Universal Mind

Joe Guaneri
Charles River Analytics

SYNC VS ASYNC

```
function add(a, b) {
  var result = a + b;
  return result;
}

console.log( add(2, 2) );
// 4
```

```
function asyncAdd(a, b, callback) {
  setTimeout(function() {
    var result = a + b;

    callback(result);

  }, 500);
}

asyncAdd(2, 2, function(result) {
  console.log(result); // 4
});
```

SYNC VS ASYNC RETURN VALUES

```
function multiply(a, b) {
  return a * b;
}

function add(a, b) {
  return a + b;
}

console.log(
  multiply(add(2, 2), 2)
); // 8
```

```
function asyncMultiply(a, b, callback) {
  setTimeout(function() {
    callback(a * b);
  }, 500);
}

function asyncAdd(a, b, callback) {
  setTimeout(function() {
    callback(a + b);
  }, 500);
}

asyncAdd(2, 2, function(result) {
  asyncMultiply(result, 2, function(result) {
    console.log(result); // 8
  });
});
```

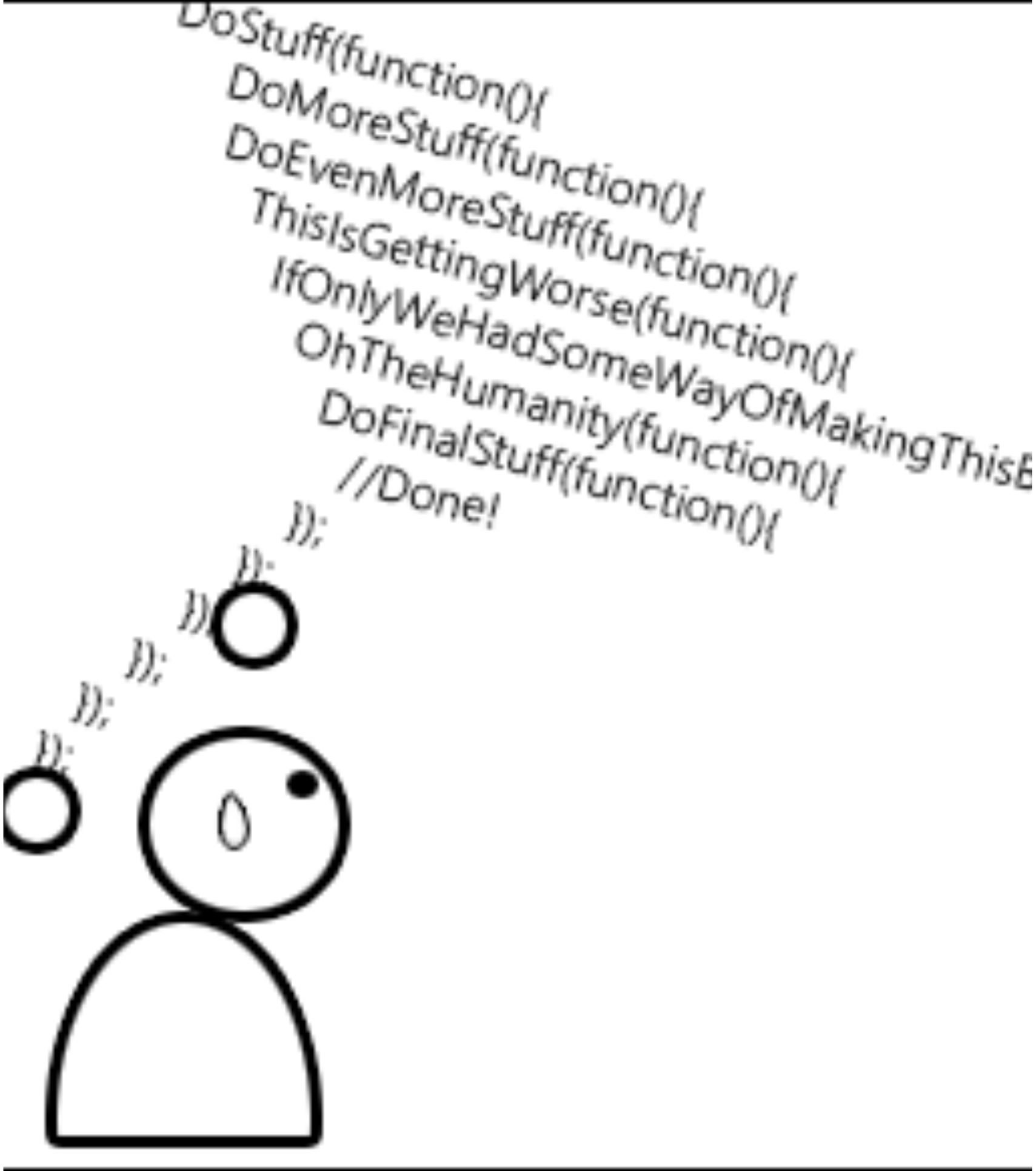
SYNC VS ASYNC EXCEPTIONS

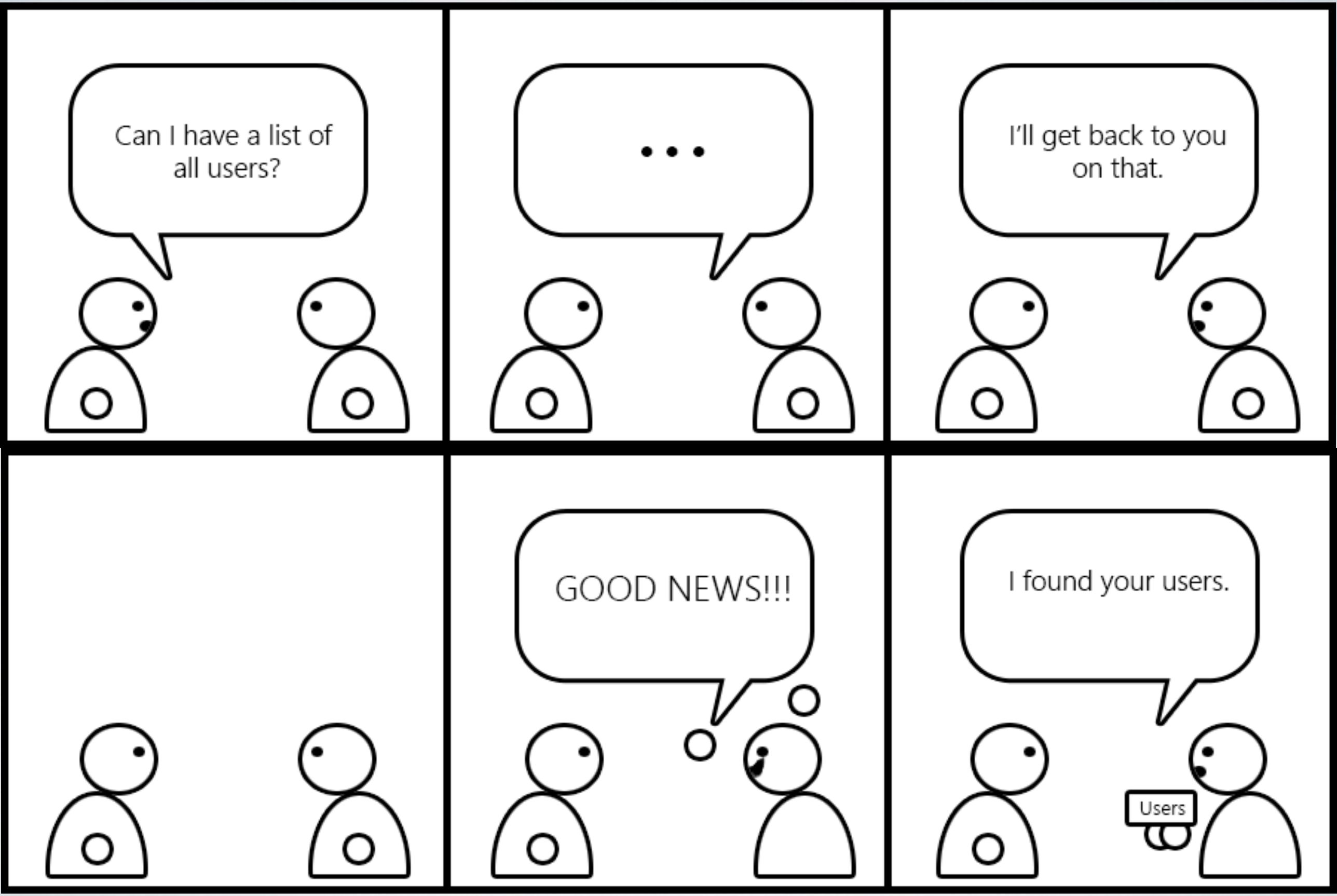
```
function add(a, b) {
  if (typeof a !== 'number') {
    throw new TypeError(
     'a is not a number');
  }
}

try {
  console.log( add('a', 2) );
} catch (e) {
  console.log(
  'you can only add numbers!');
} // you can only add numbers!
```

```
function asyncAdd(a, b, callback) {
  setTimeout(function() {
    if (typeof a !== 'number') {
      throw new TypeError(
        'a is not a number');
    }
    callback(a + b);
  }, 500);
}

asyncAdd('a', 2, function(result) {
  // never called
  console.log(result);
});
```

EXT.AJAX WITH CALLBACKS

```
Ext.Ajax.request({
  url: '/company/search/sencha',
  success: function (response) {
    Ext.Ajax.request({
      url: '/company/' + response.company.id + '/users',
      success: function (response2) {
        // use response.users
      },
      failure: function (response2) {
        // handle failure
      }
    });
  },
  failure: function (response) {
    // handle failure
  }
});
```


DoStuff(function(){
DoMoreStuff(function(){
DoEvenMoreStuff(function(){
ThisIsGettingWorse(function(){
IfOnlyWeHadSomeWayOfMakingThisB
OhTheHumanity(function(){
DoFinalStuff(function(){
//Done!
});
});
});
});
});
});


Can I have a list of all users?
. . .
I'll get back to you on that.
GOOD NEWS!!!
I found your users.
Users

EXT.AJAX WITH PROMISES

```
var promise = Ext.Ajax.request({
    url: '/company/search/sencha',
});

promise.then(function (company) {
   // use company
}).otherwise(function (reason) {
   // handle failure
});;
```

PROMISES AND RETURN VALUES

```
Ext.Ajax.request({
    url: '/company/search/sencha',
}).then(function (response) {
  return Ext.Ajax.request({
    url: '/company/' + response.company.id + '/users'
  });
}).then(function (response) {
  return response.users.map(function(user) {
    // do some transform
  });
}).then(function(users) {
  // use transformed users
}).otherwise(function (reason) {
  // handle failure
});;
```

PROMISES AND EXCEPTIONS

```
Ext.Ajax.request({
    url: '/company/search/sencha',
}).then(function (response) {
  if (somethingBad) {
    throw new Error('something bad!');
  }
  return Ext.Ajax.request({
    url: '/company/' + response.company.id + '/users'
  });
}).then(function (response) {
  // use response.users
}).otherwise(function (reason) {
  // handle failure
});;
```

CREATING A PROMISE

```
Ext.define('MyApp.service.AccountService', {
    ...

    load: function (accountId) {
        var deferred = Ext.create('Ext.Deferred');

        MyApp.model.AccountModel.load(accountId, {
            success: function (record, operation) {
                deferred.resolve(record);
            },
            failure: function (record, operation) {
                deferred.reject(operation.getError());
            },
            scope: this
        });

        return deferred.Ext.getPromise();
    }
});
```

ALWAYS()

```
accountService
    .load(accountId)
    .then(
        ...
)
.always(function () {
    // clean-up logic to execute
    // regardless of outcome
});
```

REGISTERING CALLBACKS VIA THEN()

```
accountService
    .load(accountId)
    .then(
        function (account) {
            // do something with the account
        },
        function (error) {
            // display or handle error
        },
        function (update) {
            // display progress
        },
        this
);
```

REGISTERING CALLBACKS VIA THEN() WITH NAMED PARAMS

```
accountService
    .load(accountId)
    .then({
        success: function (account) {
            // do something with the account
        },
        failure: function (error) {
            // display or handle error
        },
        progress: function (update) {
            // display progress
        },
        scope: this
});
```

CANCEL()

```
loadAccountPromise = accountService.load(accountId);
loadAccountPromise.otherwise(
    function (error) {
        // failure handler will be called with a
        // CancellationError
    }
);

loadAccountPromise.cancel();
```

PROPAGATION, RECOVERY, LOGGING

```
var accountPromise = accountService
    .load( accountId )
    .then({
        success: function (result) {
            if (result.error) {
                // detect and throw an error
                throw new Error(result.error.message);
            }
            return process(result);  // transform the result
        }
})
.otherwise(
    function (error) {
        return defaultValue;     // recover from error
    }
)
.log('Load Account:')
.done(); // ensures unhandled rejections rethrown as errors
```

TIMEOUT()

```
// Automatically reject after milliseconds
Ext.Deferred.timeout( promisesOrValues, milliseconds);
```

ALL(), ANY(), SOME(), MAP(), REDUCE()

```
// Aggregation
Ext.Deferred.all([
    obj.buyIt(),
    obj.useIt(),
    obj.breakIt(),
    obj.fixIt(),
    obj.trashIt(),
    obj.changeIt(),
    ...
]);

// Competitive Race
Ext.Deferred.any( promisesOrValues );
Ext.Deferred.some( promisesOrValues, howMany);

// Transformations
Ext.Deferred.map( promisesOrValues );
Ext.Deferred.reduce( promisesOrValues );
```

PIPELINE()

```
// Pipeline
Ext.Deferred.pipeline([
    accountService.load(),
    accountService.charge(20, ‘USD’),
    acccountService.save()
    ...
]);
```

EXT.PROMISE IS PROMISES/A+

- Implements the Promises/A+ Specification
- Passes the Promises/A+ Specification Compliance Test Suite

http://deftjs.org/

@DeftJS

Open Source

MIT License

Test Suite

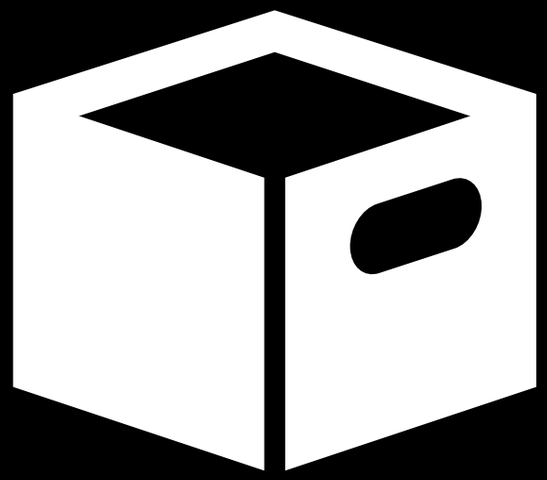
- Mocha + Chai + Sinon.JS
- Karma Test Runner
- Travis CI
  - Over 1,000 unit tests
  - ~20 supported versions of Sencha Touch and Ext JS

Supported Versions

- Ext JS 4.0.7+, 5, 6
- Sencha Touch 2.0.1+

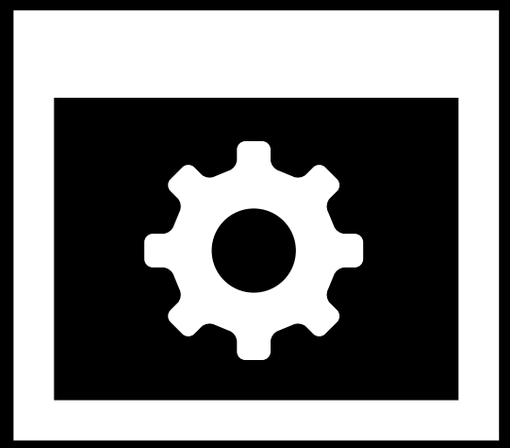
Core Features

Ext JS 4, 5, 6+



IoC Container

Ext JS 4



View Controllers

Ext JS 4, 5

Promises

Road Map

- Immutable Data ala Mori.js / Immutable.js ?
- Functional Reactive Programming (FRP) ?
- Your ideas ?
- More Documentation and Examples

PROMISES Q&A ?

```
learnPromises.then(function(knowledge) {
  // go forth and build awesome
  // applications with Sencha Ext JS 6
}).otherwise(function(question) {
  // ask questions
}).always(function(feedback) {
  // give feedback @superstructor, sencha
  // forums etc
});
```