CSP

The Future of Asynchronous Javascript

Erik Person / @thaterikperson

Thank you to our sponsors…

Austin .Net User Group Codecamp 2014

GOLD

DELL

CHANDER DHALL, INC

clarify

Bronze

ALLIED CONSULTANTS, INC.
PEOPLE. PROCESSES. TECHNOLOGIES.

Happy Hour sponsor

Headspring

DEV MANAGER @HUDL

A BRIEF HISTORY OF JAVASCRIPT

CALLBACKS

PROMISES

FUNCTIONAL REACTIVE PROGRAMMING

LET'S *Compare*

BASIC CALLBACK

```javascript
var button = $('#derp-btn');
button.on('click', function () {

    button.text('Workin\'...');

    $.post('/derps', function (data) {
        button.text('Done ' + data);

        setTimeout(function () {
            button.hide();

            // Probably more callbacks here
        }, 2000);
    });
});
```

BASIC PROMISE

```
var button = $('#derp-btn');
button.on('click').then(function () {

    button.text('Workin\'...');
    return $.post('/derps');

}).then(function (data)) {

    button.text('Done ' + data);
    return setTimeout(2000);

}).then(function () {

    button.hide();
});
```

BASIC FRP

HA!

```javascript
var button = $('#derp-btn');
var stream = button.asEventStream('click');
stream = stream.map(function (event) {
    $(event.target).text('Workin\'...');
    return { url: '/derps', method: 'POST' };
});
stream = stream.ajax();
stream = stream.delay(2000);

stream.subscribe(function () {
    button.hide();
});
```

ADVANCED CALLBACK

```
function sharedDerp (data) { /* Do something crazy */ }
var isClickDone = false;
var isXhrDone = false;
var xhrData = null;

var button = $('#derp-btn');
button.on('click', function () {
    isClickDone = true;
    if (isXhrDone) {
        sharedDerp(xhrData);
    }
});

$.post('/derps', function (data) {
    xhrData = data;
    isXhrDone = true;
    if (isClickDone) {
        sharedDerp(xhrData);
    }
});
```

ADVANCED PROMISE

```
function sharedDerp (data) { /* Do something crazy */ }
var cPromise = $('#derp-btn').on('click');
var pPromise = $.post('/derps');

Promise.all([cPromise, pPromise]).then(function (results) {
    sharedDerp(results[1]);
});
```

ADVANCED FRP

```
function sharedDerp (data) { /* Do something crazy */ }
var button = $('#derp-btn');
var buttonStream = button.asEventStream('click').toProperty(false);
var postStream = Bacon.fromPromise($.ajax({ url : '/derps' }))
var stream = buttonStream.combine(postStream, function (event, data) {
    return data;
});

stream.subscribe(function (data) {
    sharedDerp(data);
});
```

WHAT DID WE LEARN?

CALLBACKS? MEH.
PROMISES? NOT BAD.
FRP? WTF BUT OK.

NEW *Hotness*

BUT FIRST

GENERATORS

```
function *derp() {
    yield null;
}
```

```
function *derp() {
    yield 'd';
    yield 'e';
    yield 'r';
    yield 'p';
}

for (var d of derp()) {
    console.log(d);
}
// d
// e
// r
// p
```

```
var it = derp();
var result = it.next();
while (!result.done) {
    console.log(result);
    result = it.next();
}
console.log(result);

// { value: 'd', done: false }
// { value: 'e', done: false }
// { value: 'r', done: false }
// { value: 'p', done: false }
// { value: undefined, done: true }
```

```
function *derp() {
    var i = 0;
    while (true) {
        yield i;
        i++;
    }
}

var it = derp();
for (var i = 0; i < 3; i++) { console.log(it.next()); }

// { value: 0, done: false }
// { value: 1, done: false }
// { value: 2, done: false }
```

```
function *derp() {
    console.log('derp');
    var x = yield null;
    console.log(x);
}

var it = derp();
it.next();
// derp
it.next(2);
// 2
```

```javascript
var button = $('#derp-btn');
button.on('click', function () {
    button.text('Workin\'...');

    $.post('/derps', function (data) {
        button.text('Done ' + data);

        setTimeout(function () {
            button.hide();

            // Probably more callbacks here
        }, 2000);
    });
});
```

```
var button = $('#derp-btn').on('click', function () {
    it.next();
});

function postDerps() {
    $.post('/derps', function (data) {
        it.next(data);

        setTimeout(it.next, 2000);
    });
}

function *derp() {
    button.text('Workin\'...');
    postDerps();
    var data = yield null;
    button.text('Done ' + data);
    yield null;
    button.hide();
}

var it = derp();
```

REWRITE TIME

COMMUNICATING SEQUENTIAL PROCESSES

WHAT DOES IT MEAN?

USE THIS

https://github.com/ubolonton/js-csp

```javascript
var csp = require('js-csp');

csp.go(function *() {
    console.log('derp');
});

// derp
```

```
var chan = csp.chan();

csp.go(function *() {
    var result = yield csp.take(chan);
    console.log(result);
});

csp.put(chan, 'derp');
console.log('herp');

// herp
```

```
var chan = csp.chan();

csp.go(function *() {
    var result = yield csp.take(chan);
    console.log(result);
});

csp.go(function *() {
    csp.put(chan, 'derp');
    console.log('herp');
});

// herp
```

```
var chan = csp.chan();

csp.go(function *() {
    var result = yield csp.take(chan);
    console.log(result);
});

csp.go(function *() {
    yield csp.put(chan, 'derp');
    console.log('herp');
});

// herp
// derp
```

```
var button = $('#derp-btn');
button.on('click', function () {
    button.text('Workin\'...');

    $.post('/derps', function (data) {
        button.text('Done ' + data);

        setTimeout(function () {
            button.hide();

            // Probably more callbacks here
        }, 2000);
    });
});
```

```
function clickChannel(element) {
    var chan = csp.chan();
    element.on('click', function (event) {
        csp.putAsync(chan, event);
    });
    return chan;
}

csp.go(function *() {
    yield csp.take(clickChannel($('#derp-btn')));
});
```

```
function postChannel(url) {
    var chan = csp.chan();
    $.post(url, function (data) {
        csp.putAsync(chan, data);
    });
    return chan;
}

csp.go(function *() {
    yield csp.take(postChannel('/derps'));
});
```

```
csp.go(function *() {
    var button = $('#derp-btn');
    var cChan = clickChannel(button);
    yield csp.take(cChan);
    button.text('Workin\'...');

    var pChan = postChannel('/derps');
    var data = yield csp.take(pChan);
    button.text('Done ' + data);

    yield csp.take(csp.timeout(2000));
    button.hide();
});
```

```javascript
function sharedDerp (data) { /* Do something crazy */ }
var isClickDone = false;
var isXhrDone = false;
var xhrData = null;

var button = $('#derp-btn');
button.on('click', function () {
    isClickDone = true;
    if (isXhrDone) {
        sharedDerp(xhrData);
    }
});

$.post('/derps', function (data) {
    xhrData = data;
    isXhrDone = true;
    if (isClickDone) {
        sharedDerp(xhrData);
    }
});
```

```
csp.go(function *() {
    var xhrData = null;
    var cChan = clickChannel($('#derp-btn'));
    var pChan = postChannel('/derps');
    var result = yield csp.alts([cChan, pChan]);
    if (result.channel === cChan) {
        xhrData = yield csp.take(pChan)
    }
    else {
        xhrData = result.value;
        yield csp.take(cChan);
    }
    sharedDerp(xhrData);
});
```

```
function all(chans) {
    var chan = csp.chan();
    csp.go(altsAll, [chan, chans]);
    return chan;
}

function *altsAll(chan, channels) {
    var copy = channels.slice(0);
    var length = channels.length;
    for (var i = 0; i < length; i++) {
        var result = yield csp.alts(channels);
        var index = copy.indexOf(result.channel);
        copy[index] = result.value;
        channels.remove(result.channel);
    }
    csp.putAsync(chan, copy);
}
```

```
csp.go(function *() {
    var cChan = clickChannel($('#derp-btn'));
    var pChan = postChannel('/derps');
    var results = yield csp.take(all([cChan, pChan]))
    sharedDerp(results[1]);
});
```

ERROR HANDLING

```
function *derp() {
    try {
        yield null;
    }
    catch (e) {
        console.log(e);
    }
}

var it = derp();
it.next();
it.throw(new Error('derperror'));

// [Error: derperror]
```

```
var chan = csp.chan();

csp.go(function *() {
    try {
        var result = yield csp.takem(chan);
    }
    catch (e) {
        console.log(e);
    }
});

csp.go(function *() {
    yield csp.put(chan, new Error('derperror'));
});

// [Error: derperror]
```

Final Thoughts

PROMISES ARE BAD, OR NAH?

SO WHEN CAN I USE IT?

YOU CAN'T

JK LOL BUT SRSLY

```
chrome://flags
```

ENABLE EXPERIMENTAL JAVASCRIPT

OR

```
node --harmony
```

REQUIRES NODE 0.11

FURTHER READING

- http://davidwalsh.name/es6-generators
- http://swannodette.github.io/2013/07/12/communicating-sequential-processes/
- http://jlongster.com/Taming-the-Asynchronous-Beast-with-CSP-in-JavaScript

THANK YOU

@THAT ERIK PERSON