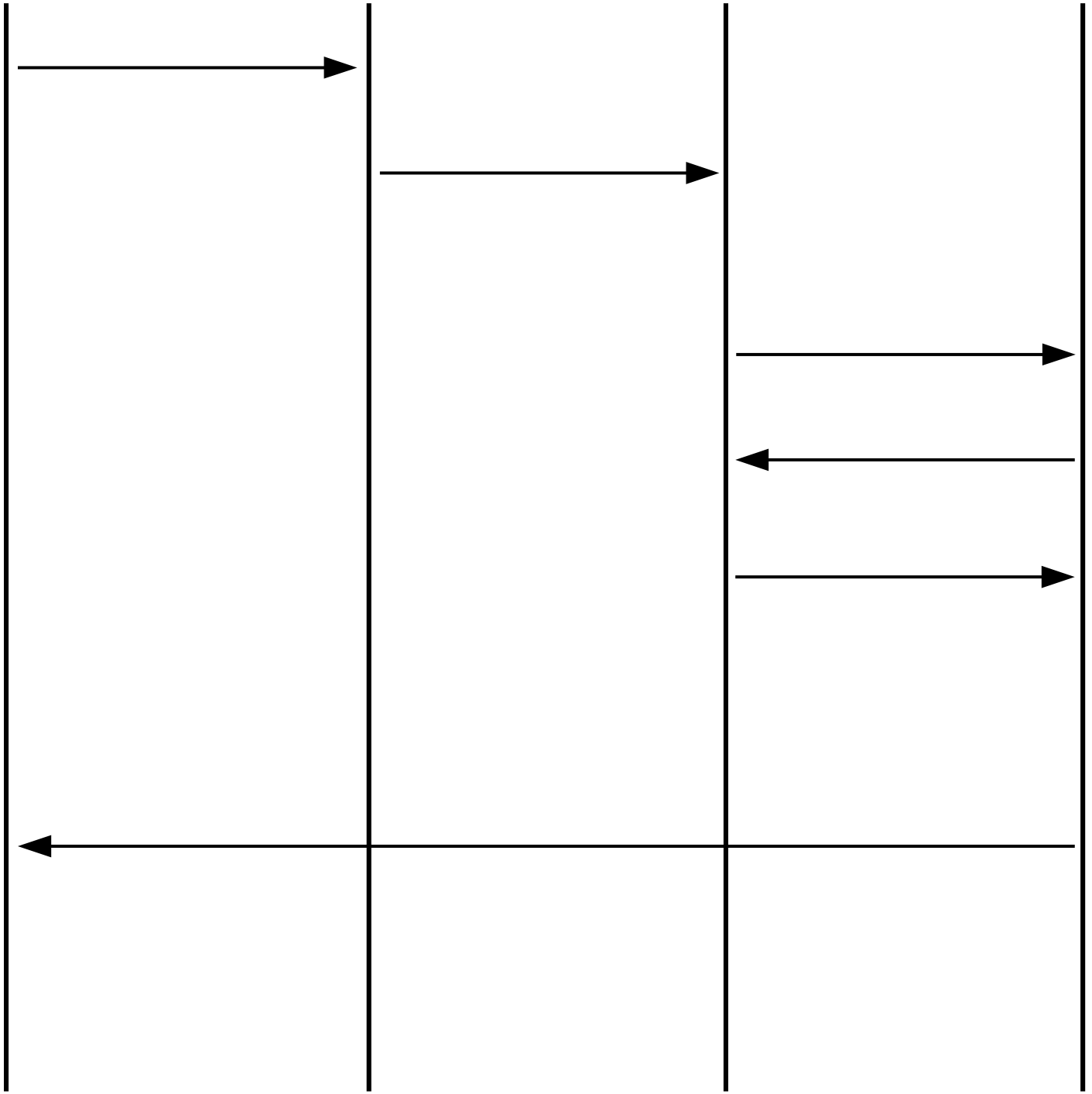
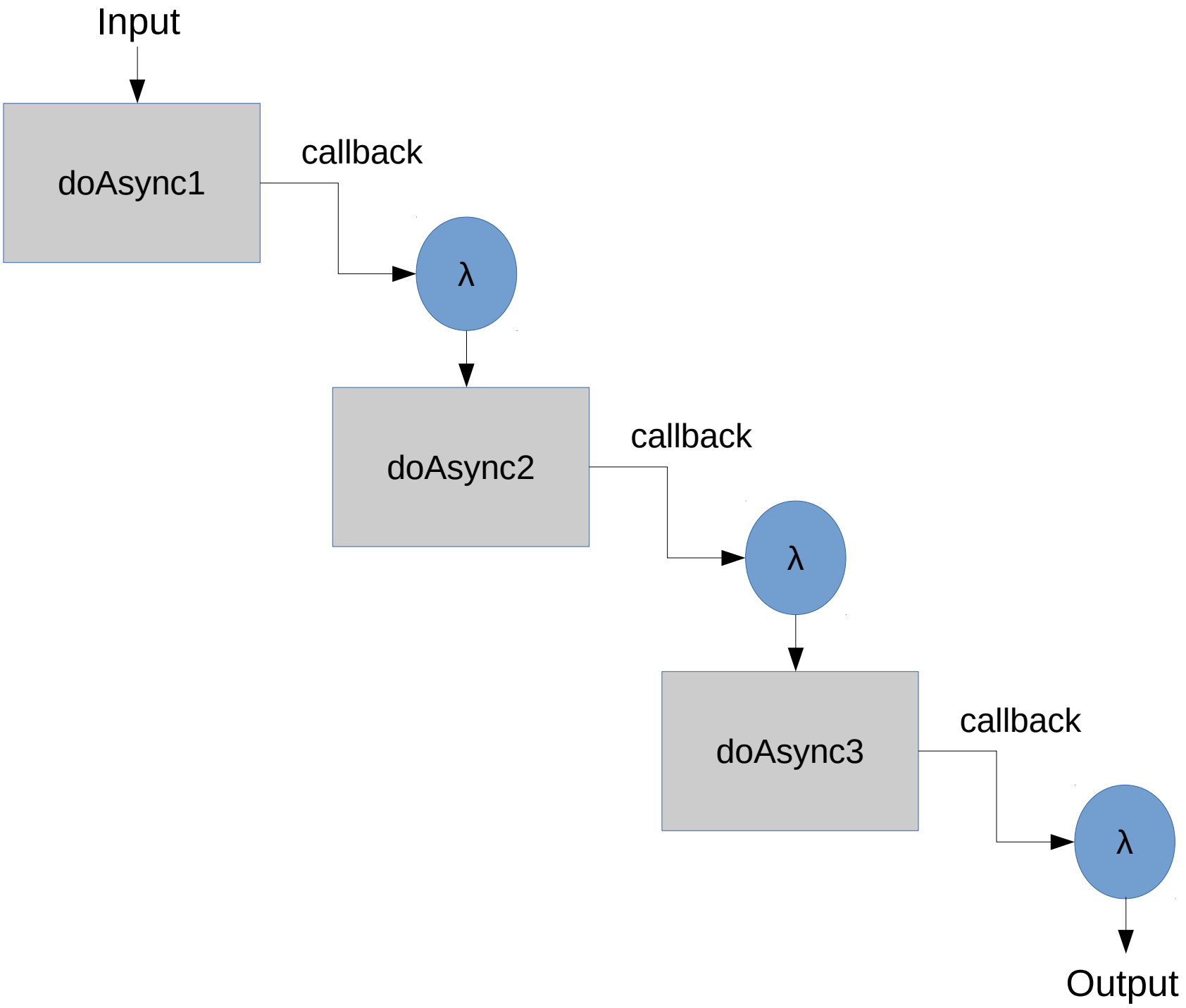
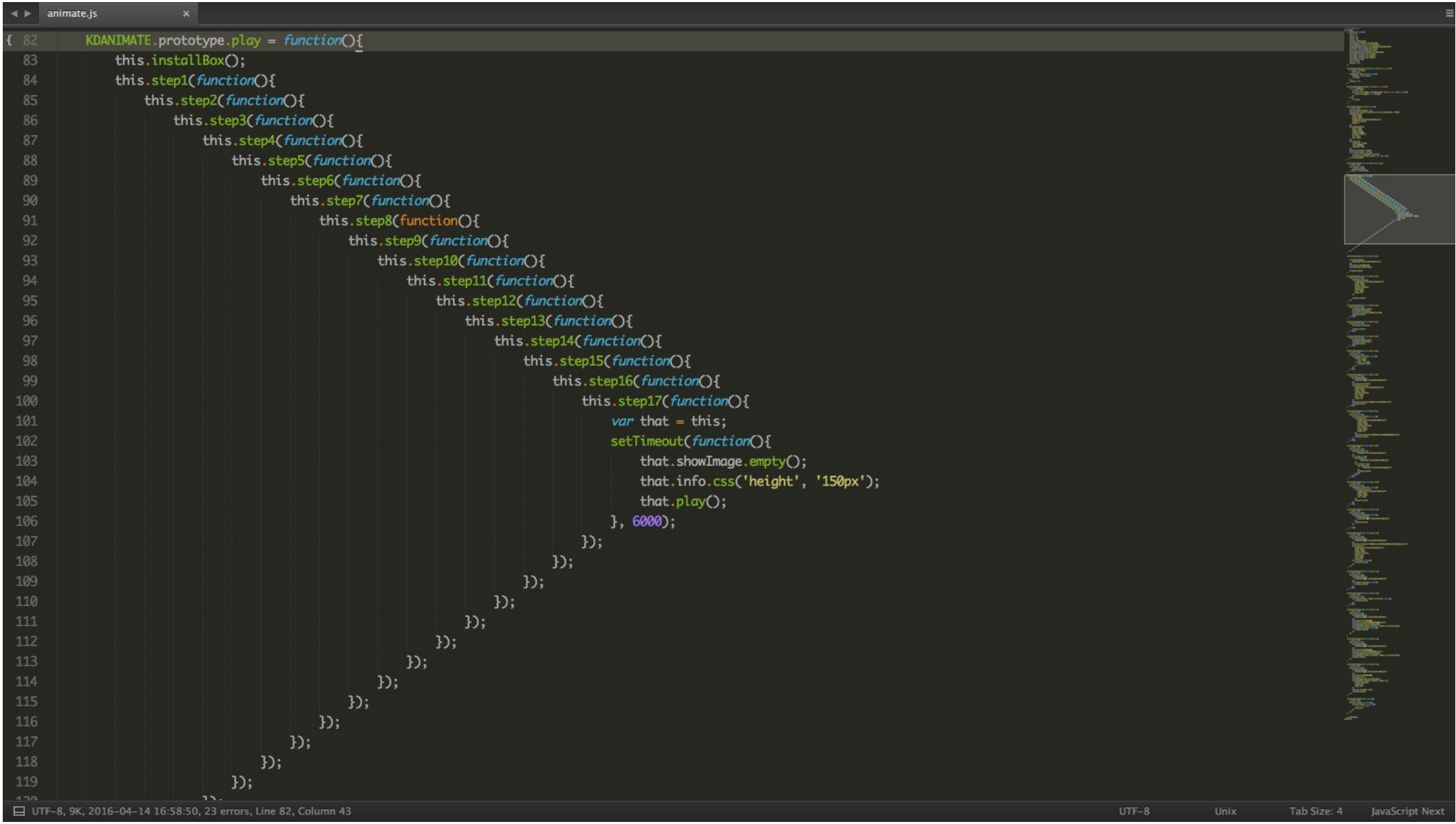
# Async/Await JavaScript

Johannes Dienst



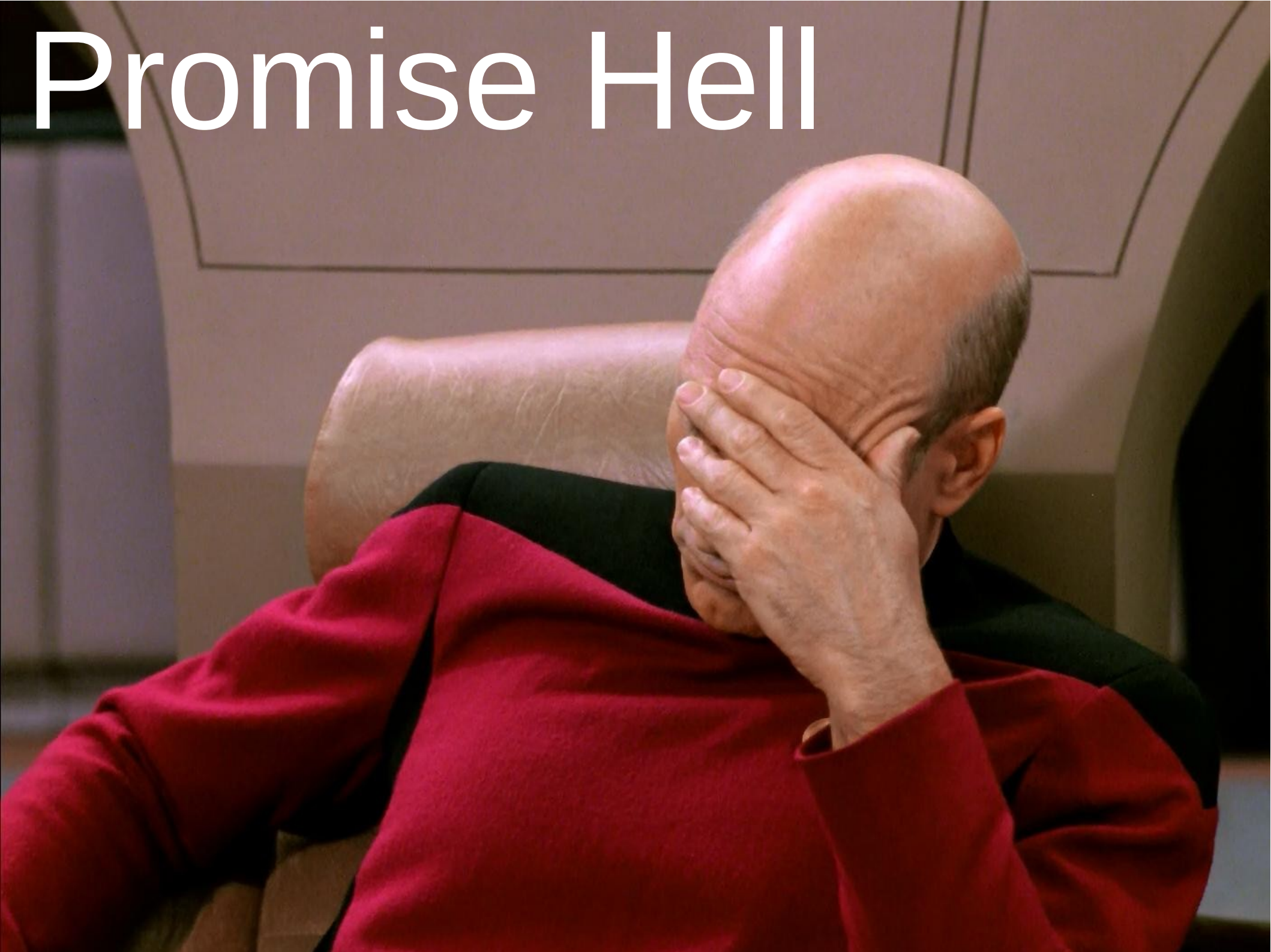

Input
doAsync1
callback
λ
doAsync2
callback
λ
doAsync3
callback
λ
Output


animate.js
KDANIMATE.prototype.play = function(){
this.installBox();
this.step1(function(){
this.step2(function(){
this.step3(function(){
this.step4(function(){
this.step5(function(){
this.step6(function(){
this.step7(function(){
this.step8(function(){
this.step9(function(){
this.step10(function(){
this.step11(function(){
this.step12(function(){
this.step13(function(){
this.step14(function(){
this.step15(function(){
this.step16(function(){
this.step17(function(){
var that = this;
setTimeout(function(){
that.showImage.empty();
that.info.css('height', '150px');
that.play();
}, 6000);
});
});
});
});
});
});
});
});
});
});
});
});
});
});
});
});
});
});
UTF-8, 9K, 2016-04-14 16:58:50, 23 errors, Line 82, Column 43
UTF-8
Unix
Tab Size: 4
JavaScript Next

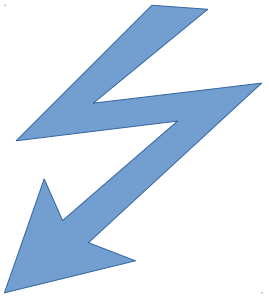
# Hasta la vista Callback Hell

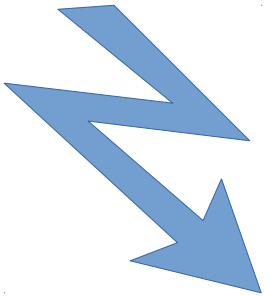
```
getFoo()
    .then((foo) => {
        return formatBar(foo);
    })
    .then((foo) => {
        return sendFooToMe(foo);
    });
```

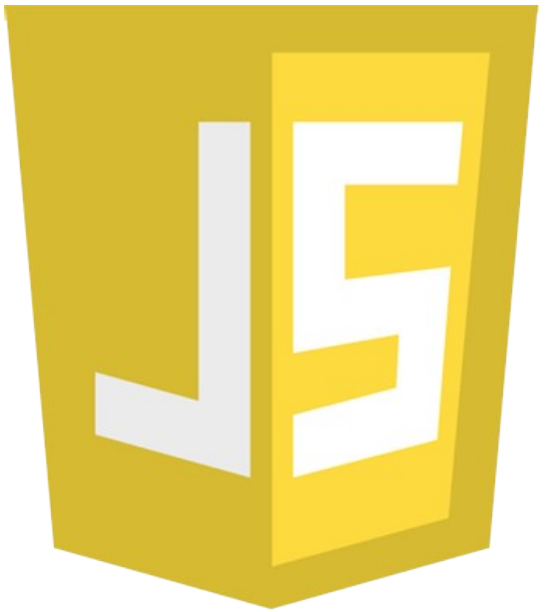
# Promise Hell



Promises/A+







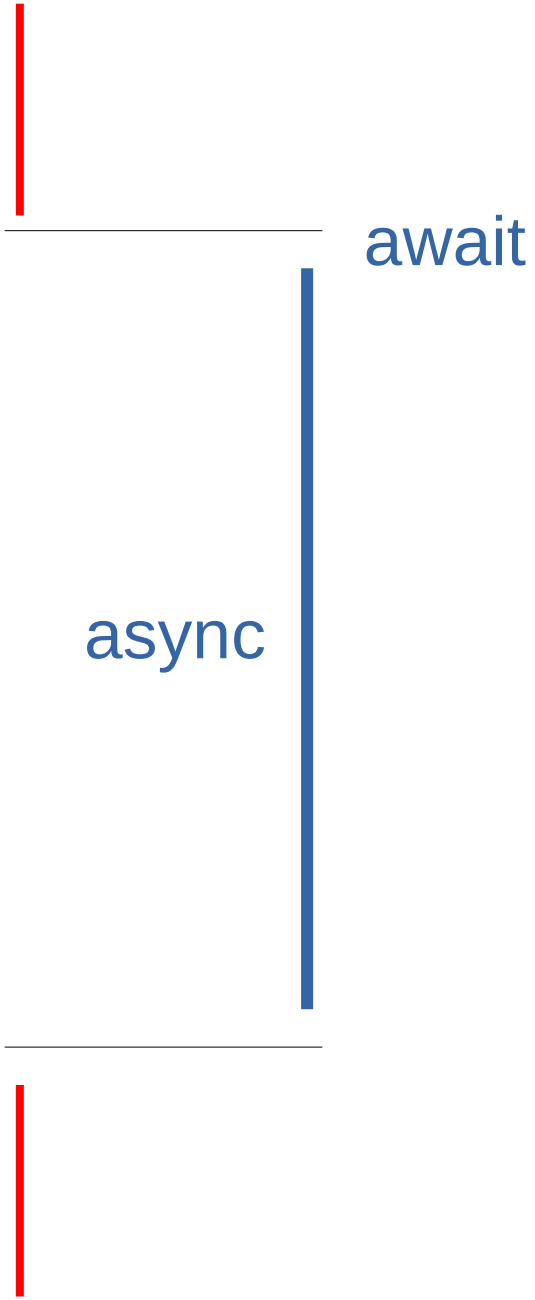
```
getDatabase()
    .then((db) => {
        return findAllFoo(db)
            .then((foos) => {
                return getBar(db)
                    .then((bar) => {
                        return pickFoobar(foos, bar);
                    });
            });
    });
```

```
function* dumbGenerator() {
    yield 1;
    yield 2;
    yield 3;
}

let gen = dumbGenerator();
gen.next()
```

```
runGenerator(

    function* main() {
        yield getPromise();
        yield getPromise();
    }

);
```

ASYNC
AWAIT

```
async function afunc() {
    let res = await getPromise();
    console.log(res);
}
```


await
async

```
async function afunc() {
    let res = await getPromise();
    console.log(res);
}

// Returns promise
let res = afunc();
```

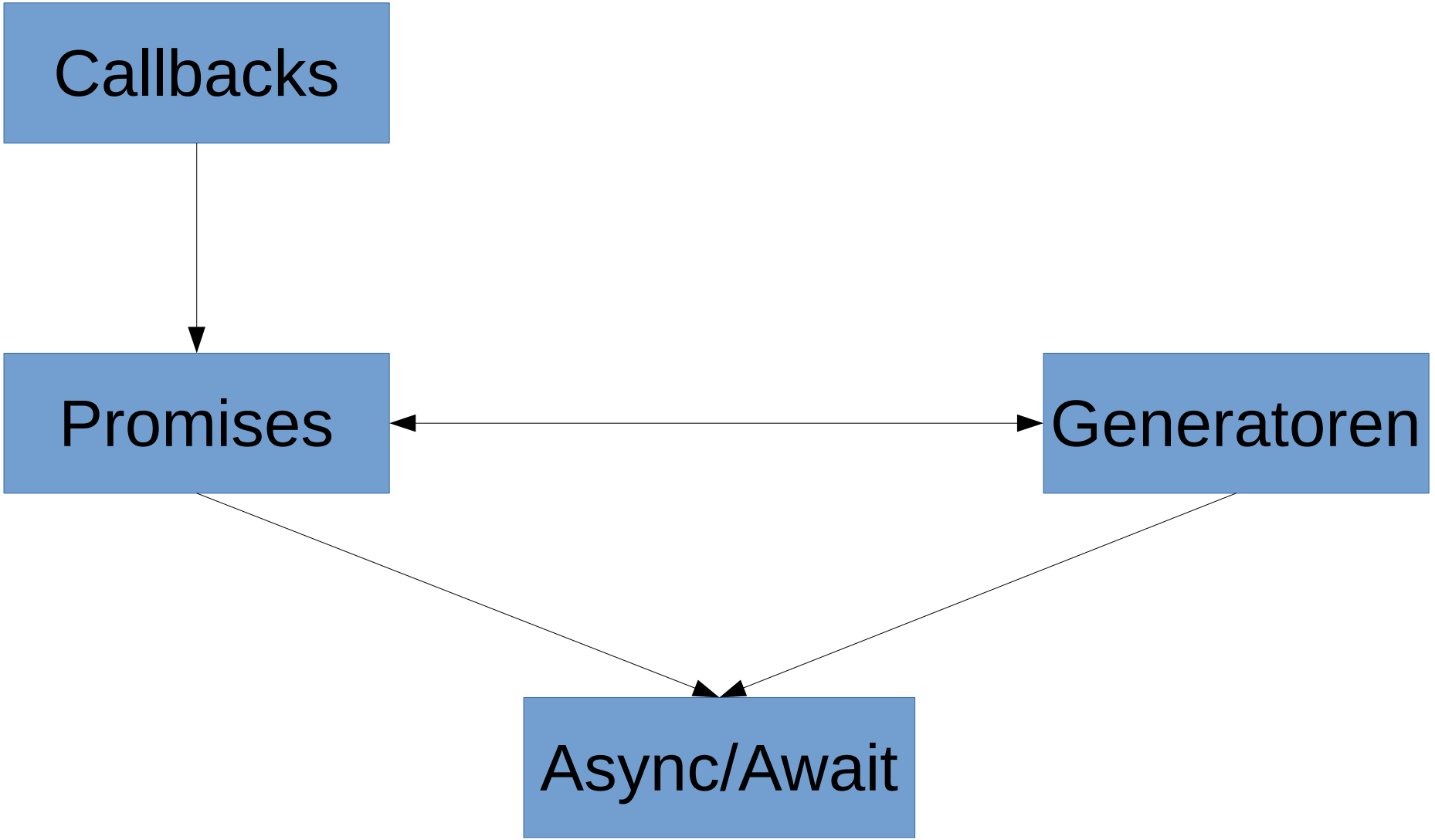
```
async function afunc() {

    await Promise.all(
        [getPromise1(), getPromise1()]
    );

}
```

```
async function afunc() {
    try {
        let res = await getPromise();
        console.log(res);
    } catch(error) {
        console.log('ERROR')
    }
}
```

# Asynchroner Code

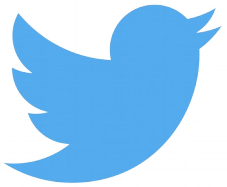
in

# synchronen Schläuchen


Callbacks
Promises
Generatoren
Async/Await

# ES2017

TypeScript



JohannesDienst

jdienst@multamedio.de

JohannesDienst.net