



# Sincronizando Código em JS

Stéfano Stypulkowski Zanata



Stéfano  
Stypulkowski  
**Zanata**



**Código Assíncrono?**





## // Exemplo 3

```
requestPersonBasicData( basic => {  
  requestPersonExtas( extras => {  
    const person = Object.assign( {}, basic, extras );  
    const dbPerson = new mongoose.models.Person( person );  
    dbPerson.save( err => {  
      requestEnrichments( person, enrichments => {  
        dbPerson.setEnrichments( enrichments );  
        dbPerson.save( err2 => {  
          // I have a complete person  
        });  
      });  
    });  
  });  
});  
});
```

A large, bright orange and yellow nuclear mushroom cloud dominates the background. The cloud has a thick, billowing stem and a wide, glowing cap. The foreground is dark and filled with smaller, darker clouds, creating a sense of depth and scale.

**CALLBACK HELL**



三姑贈贊善大夫季明之靈  
惟尔挺生夙標幼德宗廟瑚璉  
階庭蘭玉乃具精誠每慰  
人心方期戡穀何圖逆賊開  
烟直稱兵犯順尔父繼常  
山作郡余時受命之在平  
原兒愛我思得尔傳言尔既  
歸止爰開土門之門既開先威  
大震爾巨擁眾不赴賊臣不  
孤城圍逼父國陷子死巢  
傾外覆天不悔禍復為  
蒼生念尔遺殘百  
多手衣式云云  
天早移牧河東  
首親  
接

Dificuldade de  
Entendimento



The background is a dark, almost black, space filled with vibrant blue light and particle effects. Two primary sources of light are visible: one on the left and one on the right, both emitting bright blue rays that fan outwards. These rays are composed of numerous small, glowing blue particles, giving the impression of a nebula or a high-speed explosion. The overall effect is one of depth and dynamic energy, with the light rays creating a sense of movement and exploration.

# Profundidade de **Código**





**Complicado  
tratar erros**



# Inversion of Control







**Difícil de  
testar**





**Não modular**

**E agora?**



# Promises

The background of the image is a soft-focus photograph of a road during the 'golden hour' of sunset or sunrise. The sun is a bright, glowing orb in the upper right, casting a warm, orange light across the scene. The road, which has a white dashed line, curves from the bottom left towards the center. The trees and foliage on the right side of the road are silhouetted against the bright light, creating a bokeh effect with out-of-focus light spots. The overall mood is contemplative and hopeful.

# Promises

Uma *promise* representa um valor que pode estar disponível **agora**, no **futuro**, ou pode **nunca** estar disponível

Mais que isso, uma promise é **confiável**

## // Promises

```
Promise.all( [ requestPersonBasicData(), requestPersonExtas() ] ).then( values => {  
  const person = Object.assign( {}, ...values );  
  const dbPerson = new mongoose.models.Person( person );  
  dbPerson.save().then( requestEnrichments ).then( enrichments => {  
    dbPerson.setEnrichments( enrichments );  
    return dbPerson.save();  
  }).then( () => {  
    // I have a complete person  
  });  
});
```



PROMISE



ALL THE THINGS

**Melhor?**

**SIM**







**MAS AINDA NÃÃO ESTÁ BOM**

# **Generators**

# **+**

# **Promises**

# Generators

Grosseiramente: São funções que podem ser **pausadas**

Operador ***yield*** dentro da função age como um ponto de entrada e saída

The background of the slide is a blurred photograph of a desert road with double yellow lines, receding into the distance. A teal-to-yellow gradient is applied over the image, with the teal being more prominent at the top and the yellow/orange at the bottom. The text is centered in white.

# **Generator Based Control Flow**



# Generator based control flow

Controla o fluxo do código baseado na capacidade dos generators de **pausarem sua execução**

Usa **promises** como retorno das funções

Permite um **comportamento linear**

















# Run.js

Implementação original de **Kyle Simpson** e **Benjamin Gruenbaum**

*You Don't Know JS: Async & Performance*, de **Kyle Simpson**

Fonte no github: **[github.com/madeinstefano/run](https://github.com/madeinstefano/run)**

Pacote do **npm**

```
npm install simplerunner
```

# Implementações

**co**

[github.com/tj/co](https://github.com/tj/co)



[github.com/krisKowal/q](https://github.com/krisKowal/q)

**ASQ**

[github.com/getify/async](https://github.com/getify/async)



Wow, vou  
começar a  
usar agora...

Quieto Robin,  
isso não é  
nativo!



**async / await**

```
// Async & await
```

```
async function() {  
    const basic = requestPersonBasicData();  
    const extras = requestPersonExtas();  
    const person = Object.assign( {}, await basic, await person );  
    const dbPerson = new mongoose.models.Person( person );  
    await dbPerson.save();  
    const enrichments = requestEnrichments();  
    dbPerson.setEnrichments( enrichments );  
    await dbPerson.save();  
    // I have a complete person  
});
```





# Syntax Sugar





```
// Async & await chain
```

```
async function request() {  
    return something;  
}
```

```
async function process() {  
    await request();  
}
```



```
// Async & await parellel
```

```
async function proccess() {  
    return { a: await one(), b: await two() };  
}
```



```
// Async & await error handling
```

```
async function request() {
```

```
    // do async stuff
```

```
}
```

```
async function process() {
```

```
    await request().catch( err => {
```

```
        // treat me
```

```
    });
```

```
}
```

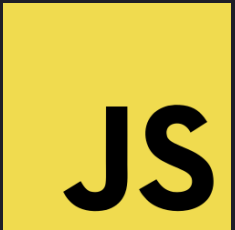


```
// Async & await error handling
```

```
async function request() {  
    // do async stuff  
}
```

```
async function process() {  
    try {  
        request()  
    } catch ( e ) {  
        // treat me  
    }  
}
```

# Aonde?



ES8 (2017)



Firefox 52



Google V8 5.5



Chrome 55



Node.js 7.6

Legal, mas **por que** devo  
usar?

**Acaba com o  
Callback  
Hell**





Código  
Simples e  
Legível



**Modular e testável**

```
// Mocha com promises
```

```
it('Should test a promise', function () {  
    return myPromise().then( result => {  
        expect( result.prop ).to.eql( 'value' )  
    });  
});
```

```
// Mocha com async & await
```

```
it('Should test a promise', async function () {  
    await process();  
    expect( assertion ).to.eql( 'value' );  
});
```





**TAKE MY MONEY!**



**OBRIGADO!**

# Referências

**Simpson, K.** You Don't Know JS: Async & Performance. Disponível em <<https://github.com/yusufdoru/Sen-JavaScript-Bilmiyorsun/blob/master/async%20&%20performance/README.md>>

\_\_\_\_\_. Going Async With ES6 Generators. Disponível em <<https://davidwalsh.name/async-generators>>

**MDN.** async function. Disponível em <[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async\\_function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function)>

**Rauschmayer, Dr. A.** ECMAScript 2017 (ES8): the final feature set. Disponível em <<http://2ality.com/2016/02/ecmascript-2017.html>>

\_\_\_\_\_. The final feature set of ECMAScript 2016 (ES7). Disponível em <<http://2ality.com/2016/01/ecmascript-2016.html>>



madein**stefano**



**szanata**.com