JavaScript for Python Developers

EuroPython
26th July, 2018

Žan Anderle
Twitter: @z_anderle

Raise your hand if…

JavaScript and Python developers

Jose Aguinaga Follow
Web Engineer. Previously @numbrs, @plaidhq, @getflynt, currently @MyBit_DApp. Javascript, #people, startups, fintech, privacy, blockchain.
Oct 3, 2016 · 13 min read

How it feels to learn JavaScript in 2016

No JavaScript frameworks were created during the writing of this article.

https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f


WHAT IS THIS?
THAT'S THE WHOLE JOKE: IT'S ABOUT JAVASCRIPT CONTEXT
nerator.net


JAVASCRIPT
JAVASCRIPT EVERYWHERE
makeameme.org

Overview

- JavaScript history and versions
- Basics of the language
- JavaScript ecosystem
- How to make sense of it all?

Žan Anderle @ EuroPython @z_anderle · Apr 19

Thinking about ideas for talks/tutorials at @djangocon . Wondering if Django + Angular would still be interesting? And whether that's better as a tutorial or a talk?

Hmm, what might be some other talk or tutorial ideas?

4 10

Ed Rivas
@je92rivas

Following

Replying to @z_anderle @djangocon

How about "JavaScript for Python developers"? Bet it would be valuable to those diving into frontend development or just JS in general.

11:52 PM - 19 Apr 2018

5 Likes

1 5

Overview

- JavaScript history and versions
- Basics of the language
- Different tools
- How to make sense of it all?

Edition	Date published	Changes from prior edition
1	June 1997	First edition
2	June 1998	Editorial changes to keep the specification fully aligned with ISO/IEC 16262 international standard
3	December 1999	Added regular expressions, better string handling, new control statements, try/catch exception handling, tighter definition of errors, formatting for numeric output and other enhancements
4	*Abandoned*	Fourth Edition was abandoned, due to political differences concerning language complexity. Many features proposed for the Fourth Edition have been completely dropped; some are proposed for ECMAScript Harmony.
5	December 2009	Adds "strict mode," a subset intended to provide more thorough error checking and avoid error-prone constructs. Clarifies many ambiguities in the 3rd edition specification, and accommodates behaviour of real-world implementations that differed consistently from that specification. Adds some new features, such as getters and setters, library support for JSON, and more complete reflection on object properties.[9]
5.1	June 2011	This edition 5.1 of the ECMAScript standard is fully aligned with third edition of the international standard ISO/IEC 16262:2011.
6	June 2015[10]	The sixth edition, initially known as ECMAScript 6 (ES6) and later renamed to ECMAScript 2015 (ES2015)[10] adds significant new syntax for writing complex applications, including classes and modules, but defines them semantically in the same terms as ECMAScript 5 strict mode. Other new features include iterators and `for`/`of` loops, Python-style generators and generator expressions, arrow functions, binary data, typed arrays, collections (maps, sets and weak maps), promises, number and math enhancements, reflection, and proxies (metaprogramming for virtual objects and wrappers). As the first "ECMAScript Harmony" specification, it is also known as "ES6 Harmony."
7	June 2016[11]	ECMAScript 2016 (ES2016)[11], the seventh edition, intended to continue the themes of language reform, code isolation, control of effects and library/tool enabling from ES2015, includes two new features: the exponentiation operator (`**`) and Array.prototype.includes.
8	June 2017[8]	ECMAScript 2017 (ES2017), the eighth edition, includes features for concurrency and atomics, syntactic integration with promises (async/await).[12][8]

Overview

- JavaScript history and versions
- Basics of the language
- Different tools
- How to make sense of it all?

Syntax

```
let myName = 'EuroPython 2018';
function sayHi(name) {
  console.log(`Hey there, ${name}`);
}

sayHi(myName); // 'Hey there, EuroPython 2018';

let someArray = [1, 2, 5, 10];
let newArray = [];

for (let el of someArray) {
    if (el > 2) {
      newArray.push(el);
    } else {
      console.log('Nope!');
    }
}
// 'Nope!'
// 'Nope!'
```

Syntax

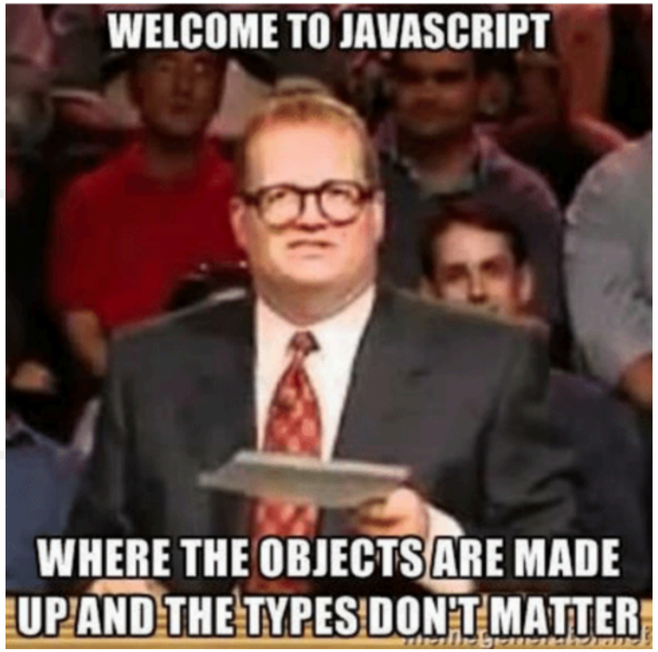
```
class Hero {
  constructor(name, superPower) {
    this.name = name;
    this.superPower = superPower;
  }

  superPower() {
    console.log('I can count really fast!');
    let count = 0;
    while (count < 1000) {
      count++;
    }
    return count;
  }
}

let superMan = new Hero('SuperMan');

superMan.superPower();
// 'I can count really fast!'
// 1001
```

Syntax

```
let x = 1;   // x is a number
x = 'Hi!';   // x is now a string
x = () => { return 1; };   // x is now a function
```

Syntax

```
> 1 + '2'
<· "12"
> '1' + 2
<· "12"
> '1' + 2 - 2
<· 10
```



Variables

```
var x = 1;
let name = 'John';
const someConstant = 45;
```

Variable hoisting

```
var x = 1;

// Some other code

var name = 'John';
```

```
var x;
var name;
x = 1;

// Some other code

name = 'John';
```

Variable hoisting

```
var txt = ["a","b","c"];

for (var i = 0; i < 3; ++i ) {
    var msg = txt[i];
    setTimeout(function() { alert(msg); }, i*1000);
}

// Alerts 'c', 'c', 'c'
```

Data Types

- Boolean
- String
- Number
- Null
- Undefined
- Object

```
let a = true;
let b = false;

let name = 'John';
name.length; // 4

let num = -124.56;
num = 10;

let empty = null;
let unknown = undefined;

let something = {key: 'A value', anotherKey: name};
let things = ['string', 2, (x, y) => { return x + y; }];
```

Object literal

```
let bigObj = {
  key: 'Some string',
  add: function(x, y) { return x + y; },
  anotherObj: {
    name: 'I am a nested object'
  }
};
```

Objects are mutable

```
> x = {a: 1}
< ▶ {a: 1}
> y = x;
< ▶ {a: 1}
> y.b = 2
< 2
> x
< ▶ {a: 1, b: 2}
```

Operators

```
if (!a && b) {
  // Some code
} else if (a || b) {
  // Some code
}
```

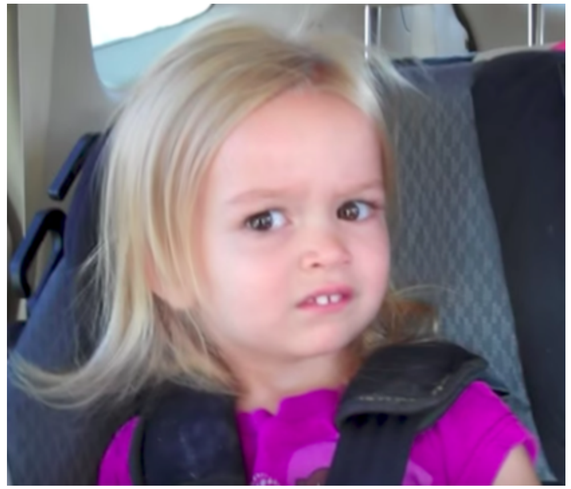
Operators

== and !=

OR

=== and !==

Operators



```
"'' == '0'" => false
"0 == ''" => true
"0 == '0'" => true

"false == 'false'" => false
"false == '0'" => true

"false == undefined" => false
"false == null" => false
"null == 'undefined'" => false
```

Functions

```
let func = function(a, b) {
  return a + b;
};

let func = (a, b) => { return a + b; };
let func = (a, b) => a + b;
```

Functions

```
function func(a = 1, b = 2) {
 return a + b;
}

func(5); // 7
```

Functions

```
function func(a = 1, b = 2) {
  // Do some calculations
}

func(5); // undefined
```

this

```
> var pets = {
    names: ['Baron', 'Chief', 'Axel'],
    owner: 'Jason',
    description: function(){
      return this.names.map(function(pet){
        return `${this.owner} knows an awesome dog named ${pet}.`
      });
    }
  };

  pets.description()
<   ["undefined knows an awesome dog named Baron.", "undefined knows
  ▶ an awesome dog named Chief.", "undefined knows an awesome dog
    named Axel."]

>
```

this

JS javascript.js ●

```
1   let pets = {
2       names: ['Baron', 'Chief', 'Axel'],
3       owner: 'Jason',
4       description: function () {
5           let that = this;
6           return this.names.map(function (pet) {
7               return `${that.owner} knows an awesome dog named ${pet}.`
8           });
9       }
10  };
11  pets.description();
12
```

this

JS javascript.js ●

```
1   let pets = {
2       names: ['Baron', 'Chief', 'Axel'],
3       owner: 'Jason',
4       description: function () {
5           return this.names.map((pet) => {
6               return `${this.owner} knows an awesome dog named ${pet}.`
7           });
8       }
9   };
10  pets.description();
11
```

```
▶ ["Jason knows an awesome dog named Baron.", "Jason knows an awesome dog named Chief.", "Jason knows an awesome dog named Axel."]
```

Classes

python.py

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def say_hi(self):
        print('Hi {}'.format(self.name))

class Dog(Animal):
    pass

dog = Dog('Billy')
dog.say_hi()
```

javascript.js

```javascript
class Animal {
    constructor(name) {
        this.name = name;
    }

    sayHi() {
        console.log(`Hi ${this.name}`);
    }
}

class Dog extends Animal {

}

let dog = new Dog('Billy');
dog.sayHi();
```

Modules

python.py

```python
from animals import Dog

dog = Dog('Billy')

```

javascript.js

```javascript
// animals.js
export class Dog extends Animal {…
}

// main.js
import { Dog } from 'animals';

dog = new Dog('Billy');

```

Template literals

```
var a = 5;
var b = 10;
console.log(`Fifteen is ${a + b} and
not ${2 * a + b}.`);
// "Fifteen is 15 and
// not 20."
```

Template literals

```
let a = 5;
let b = 10;
console.log('Fifteen is ' + (a + b) + ' and\nnot ' + (2 * a +
b) + '.');
// "Fifteen is 15 and
// not 20."
```

Promises

JS javascript.js ●

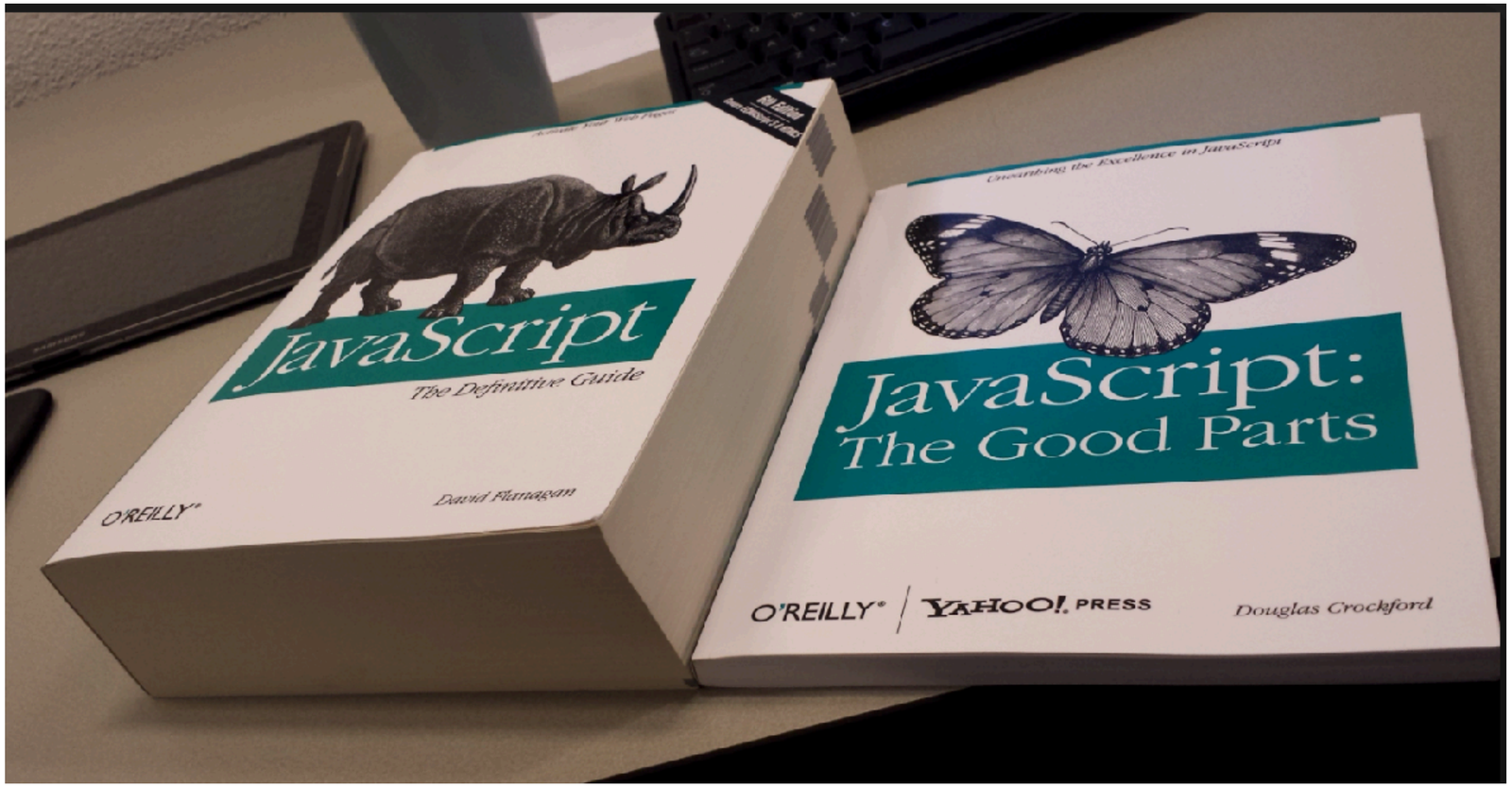
```
1    // getPage returns a Promise
2    let loadPageContents = getPage(someUrl).then((result) => {
3        return doSomething(result);
4    }).catch((error) => {
5        handleError(error);
6    });
7
8    // loadPageContents is a Promise
9    loadPageContents().then(() => {
10       changeElementOnPage();
11   });
12
```

Overview

- JavaScript history and versions
- Basics of the language
- Different tools
- How to make sense of it all?

Bad Parts

- Global variables
- ==
- +
- scope



TypeScript

```
function greeter(person: string) {
    return "Hello, " + person;
}

let user = [0, 1, 2];

document.body.innerHTML = greeter(user);
```

Overview

- JavaScript history and versions
- Basics of the language
- Different tools
- How to make sense of it all?

Different tools

- npm, bower, yarn
- Babel
- Webpack
- gulp, grunt

Different tools

- npm, bower, yarn
- Babel
- Webpack
- gulp, grunt

Different tools

- npm, bower, yarn
- Babel
- Webpack
- gulp, grunt

Different tools

- npm, bower, yarn
- Babel
- Webpack
- gulp, grunt

Different tools

- npm, bower, yarn
- Babel
- Webpack
- gulp, grunt


ember


React


Vue.js

Overview

- JavaScript history and versions
- Basics of the language
- Different tools
- How to make sense of it all?

How to get started

- Start somewhere
- Prepare your codebase
- No need to learn and use everything at once


ember


React


Vue.js

Thank you!
Questions?