ECMeowScript

What's new in JavaScript Explained with Cats

@Web Directions Code://Remote 2020

Tomomi Imura
@girlie_mac

🐱-as-a-Service

Hello 👋

Tomomi is:

- Working at Microsoft
- JavaScript all the things
- Live & work in San Francisco
- Cat-as-a-Service provider

@girlie_mac

Cat-as-a-Service

🐱aaS is a development and distribution model using cats to make it more enjoyable

Cat-as-a-Service: HTTP Status Cats

@girlie_mac

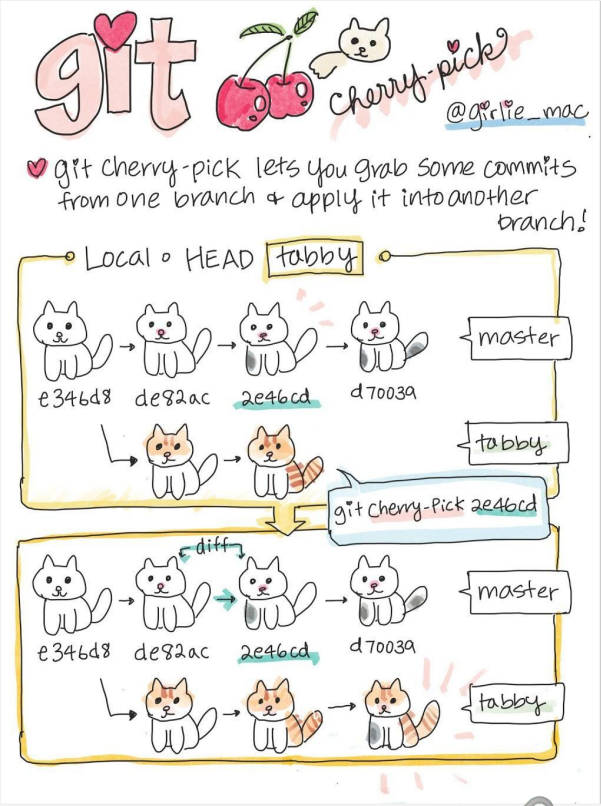
https://http.cat (not my site, but I let the owner to park my images because the TLD is awesome)

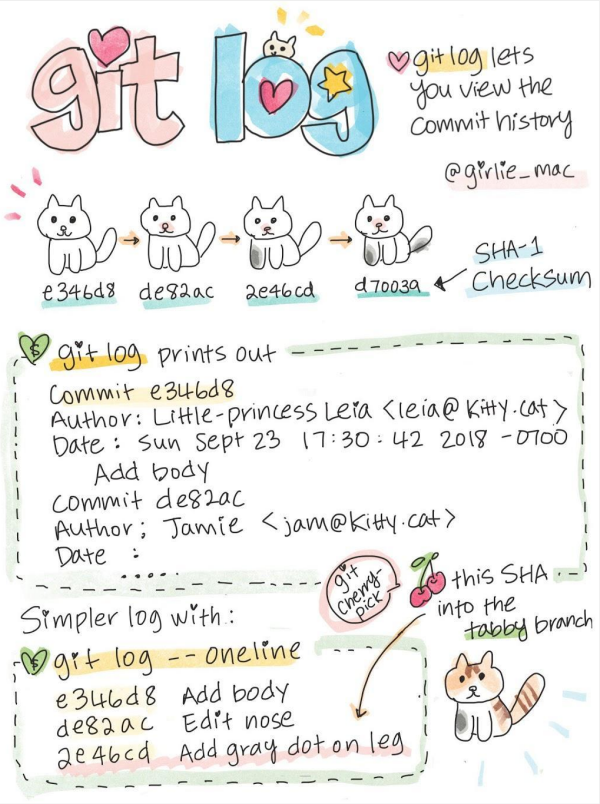
Cat-as-a-Service: Raspberry Pi KittyCam

@girlie_mac

https://github.com/girliemac/RPi-KittyCam

Cat-as-a-Service: Git Purr





@girlie_mac

https://girliemac.com/blog/2017/12/26/git-purr/

Yay, JavaScript!

Yay, EcmaScript!

Yay, EcmaScript with cats!

ECMeowScript!

WTF!

Meow

@girlie_mac

Image credit: @tikkafromeast (Instagram)

ES6 / ES2015

ES6 / ES2015: Major Changes & Improvement

ES6 / ES2015: In a nutshell

- Arrow functions
- `const` & `let` (block scope, vs. `ver` as function scope)
- Template literals
  - Expressions interpolation `` `Hi, ${cat.name}` ``
  - String interpolation (multi-lines without annoying `\n +`) etc.
- Promises
- Iterators & generators (function*)
- `for..of` iterator

ES6 / ES2015: In a nutshell

- Class definition & inheritance
- Destructuring assignment `const {name, age, breed} = cat`
- Rest parameter & Spread operator ("Three dots") `...cats`
- Set object
- Map object ("dictionary")

Also:

- Internationalization API -- e.g. `Intl.DateTimeFormat()`

My article on the ECMA-402 International API, `Intl.DateTimeFormat`

https://girliemac.com/blog/2019/04/02/javascript-i18n-reiwa-era/

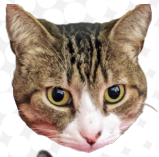
April 02, 2019

Reiwa - JavaScript International Date Format and Japan's New Imperial Era

As Emperor Akihito of Japan is set to abdicate soon, the Japanese government announced on April 1st that the reign of the next emperor will be known as the Reiwa (令和) era.

@girlie_mac

ES6 / ES2015: Examples

```
const cats = [
 {name: 'Jamie', type: 'tabby', img:  },
 {name: 'Leia', type: 'tuxedo', img:  },
 {name: 'Alice', type: 'tortoiseshell', img:  }
];
```

ES6 / ES2015: Examples

```
const cats = [
 {name: 'Jamie', type: 'tabby', img: '00_jamie.jpg'},
 {name: 'Leia', type: 'tuxedo', img: '01_leia.jpg'},
 {name: 'Alice', type: 'tortoiseshell', img: '02_alice.jpg'}
]
```

From the given array above, create a new array, `catNames`

`['Jamie', 'Leia', 'Alice']`

ES6 / ES2015: Examples

Pre-ES6

```
var catNames = [];

cats.forEach(function(cat) {
  catNames.push(cat.name);
});

// catNames is ['Jamie', 'Leia', 'Alice'];
```

ES6 / ES2015: Array.prototype.map()

`.map()` **creates a new array** from calling a function on each item in the original array.

```
const nums = [1, 2, 3, 4];

const newNums = nums.map(function(n) {
  return n * 2;
});
// [2, 4, 6, 8]
```

Map calls a callback on each element in an array!

ES6 / ES2015: Array.prototype.map()

with ES6, array.map()

```
const catNames = cats.map(function(cat) {
  return cat.name;
});

// ['Jamie', 'Leia', 'Alice'];
```

ES6 /ES2015: Arrow Function

write it simpler with with ES6 Arrow function

```
const catNames = cats.map((cat) => {
  return cat.name;
});

// ['Jamie', 'Leia', 'Alice'];
```

ES6 / ES2015: Arrow Function (Concisely!)

Even simpler with ES6 concise arrow function syntax!

```
const catNames = cats.map(cat => cat.name);

// ['Jamie', 'Leia', 'Alice'];
```

ES6 / ES2015: Destructuring

Alternatively with ES6 Object destructuring

```
const catNames4 = cats.map(({name}) => name);

// ['Jamie', 'Leia', 'Alice'];
```

extract **{ name }** property out of the **cats** object

ES6 / ES2015: Destructuring

```
const cat01 = {
  name: 'Jamie',
  type: 'tabby',
  img: '001.jpg'
};

const {name01, type01, img01} = cat01;
```

```
// Pre-ES6
var name01 = cat01.name;
var type01 = cat01.type;
var img01 = cat01.img;
```

High-five!

@girlie_mac

ES6 / ES2015: Spread operator “Three-dots”

Spread operator allows an iterable (array or string) to be expanded.

e.g. concat arrays:

```
catNames = ['Jamie', 'Leia', 'Alice'];

const newCatNames = [...catNames, 'Yugi','Snori'];


// newCatNames is ['Jamie','Leia','Alice','Yugi','Snori']
```

Three-dots Tricks

girliemac.com/blog/2020/06/18/javascript-spread-operator

Concat

```
const coats1 = ['solid', 'bicolor', 'tabby'];
const coats2 = ['calico', 'tortoiseshell'];
const coats = [...coats1, ...coats2];
// ['solid', 'bicolor', 'tabby', 'calico', 'tortoiseshell']
```

Convert a string to array

```
const str = 'kitty';
const newArr = [...str]; // ['k', 'i', 't', 't', 'y'];
```

Find the largest

```
const catWeights = [10, 9, 6, 12];
const maxWeight = Math.max(...catWeights); // 12
```

Remove duplicates from an array

```
const allCatNames =['chewie', 'leia', 'yoda', 'chewie', 'luke', 'leia'];
const catNames = [...new Set(allCatNames)];
// ['chewie', 'leia', 'yoda', 'luke'];
```

ES6 / ES2015: Map

Yay, JS HashMap! Let's **cat**alog a collection of cats!

Key	Value
'Jamie'	1234
'Leia'	1455
'Alice'	3456

```
const map = new Map();
map.set('Jamie', 1234);
map.set('Leia', 1455);
map.set('Alice', 3456);

console.log(map);
// {"Jamie" => 1234, "Leia" => 1455, "Alice" => 3456}

map.has('Jamie'); // true
map.get('Jamie'); // 1234
```

@girlie_mac

ES7 / ES2016

ES7 / ES2016: In a Nutshell

- Exponentiation Operator
- Array.prototype.includes()

ES7 / 2016: Exponentiation Operator **

a * * b

@girlie_mac

ES7 / 2016: Exponentiation Operator **

The exponentiation operator returns the result of raising the first operand (var a) to the power of the second operand (var b).

So:

```
a ** b
```

is same as

```
Math.pow(a, b)
```

3

2

```
// 3 ** 2 returns  9
```

@girlie_mac

ES7 / ES2016: Array.prototype.includes()

```
const coatColors =
['solid', 'bicolor', 'tabby', 'calico', 'tortoiseshell'];

coatColors.includes('tabby') // true
coatColors.includes('neon') // false
```

To do the same operations with indexOf()

```
coatColors.indexOf('tabby') > -1 // true
coatColors.indexOf('neon') > -1 // false
```

ES8 / ES2017

ES8 / ES2017: In a Nutshell

- Async / await
- padStart() and padEnd()
- Trailing commas
- Object.entries() - returns an object w/ key-value pairs in an array as an array of arrays
- Object.values() - returns an array of a given object's enumerable property values
- Shared memory and atomics

ES8 / ES2017: Async & Await

An **async** function with **await** expression:

- is a syntactic sugar on top of promises
- pauses the execution of the async function to wait for the passed Promise's resolution
- makes the async code look like synchronous code (= simpler!)

ES8 / ES2017: Async & Await

litter·robot®

litter-robot.com

https://youtube.com/watch?v=zZHeZ25NFso

ES8 / ES2017: Async & Await

```
const runLitterRobot = async() => {
  await cat.poop();
  clean();
};
```

cat.poop() → clean()

@girlie_mac

ES8 / ES2017: Async & Await

```
const runLitterRobot = () => {
  cat.poop();
  clean();
};
```

What can it *poo*sibly go wrong?

@girlie_mac

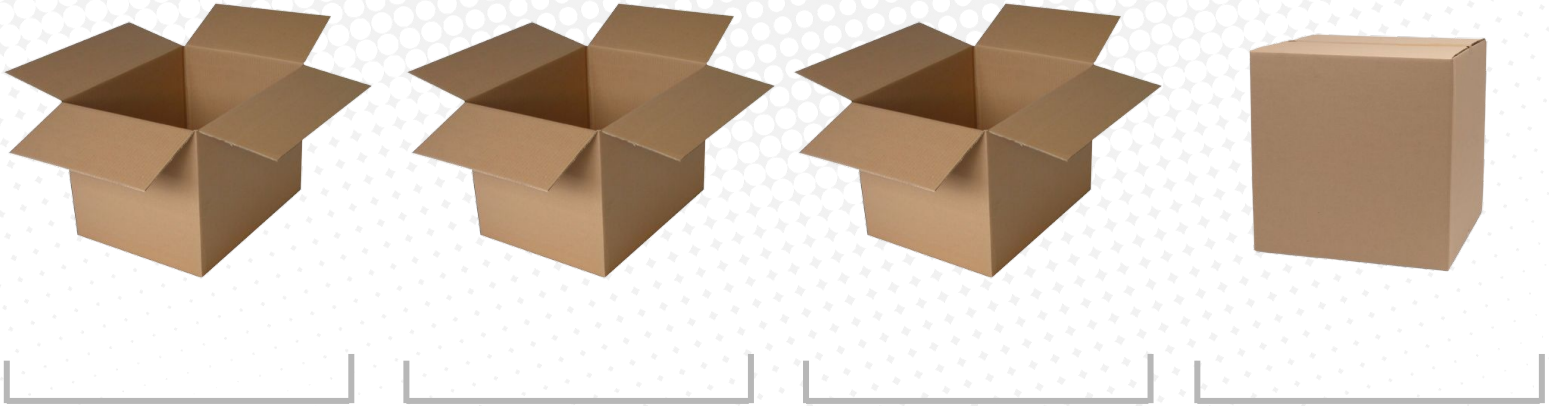
ES8 / ES2017: String Padding

`String.prototype.padStart()` and `padEnd()`

```
const boxStr = '📦'
```

ES8 / ES 2017: String Padding

Let's make a string, total 4 chars long, with some paddings in the beginning of the string.



@girlie_mac

ES8 / ES 2017: String Padding

```
boxStr.padStart(4, '😺');
// boxStr is now ('😺😺😺📦')
```

@girlie_mac

ES8 / ES 2017: String Padding

Example with `padStart()` to pad out a value with leading 0

```
const str = '5';
str.padStart(4, '0'); // '0005'
```

0005.jpg

ES8 / ES 2017: String Padding

`padStart()` and `padEnd()` work as expected with **RtL** strings too.
(This is why the methods are not called padLeft/Right!)

```
const strArabic = 'أنا قطة'
strArabic.padEnd(10,'قط');


// 'أنا قطةقطقطقط'
```

"I am a cat"

@girlie_mac

ES8 / ES 2017: Trailing comma in Func Param

(param)

Not allowed in pre-ES8

@girlie_mac

ES8 / ES 2017: Trailing comma in Func Param

```
(param, )
```

@girlie_mac

ES8 / ES 2017: Trailing comma in Func Param

In ES5: ✅ Trailing comma in object literals

In ES8: ✅ Trailing comma in function parameter list & calls

```
const makeFood = (param1, param2,) => {..};
makeFood('tuna', 'salmon',);
```

Approved!

@girlie_mac

ES9 / ES2018

ES9 / ES2018: In a Nutshell

- Spread & Rest properties - now you can use the three-dots with objects!
- RegEx improvements
- Asynchronous iteration
- Promise finally()

```
fetch('https://api.github.com/users/octocat')
  .then(result => {···})
  .catch(error => {···})
  .finally(() => {console.log('🐙🐱')});
```

ES10 / ES2019

ES10 / ES2019: In a Nutshell

- String.prototype.trimStart() and trimEnd()
- Array.prototype.flat() and flatMap()
- Object.prototype.fromEntries()
- Function.prototype.toString()
- Well-formed JSON.stringify()
- Better & faster array sort()

ES2019: String.prototype.trimStart() & trimEnd()

Top Performance Cat Grooming Bag By Top Performance

34 Reviews | 11 Answered Questions

Price: **$12.99** FREE 1-2 DAY SHIPPING OVER $49

In stock

Size: **Small**

Small | Medium | Large

Quantity 1

Add to Cart

Add to Favorites

Chewy.com

@girlie_mac

ES2019: String.prototype.trimStart()

@girlie_mac

ES2019: String.prototype.trimStart()

The `trimStart()` removes whitespace from the beginning of a string

```
const greetStr = '  meow  ';
greetStr.trimStart(); // 'meow  '
```

ES2019: String.prototype.trimEnd()

Snip!

@girlie_mac

ES2019: String.prototype.trimEnd()

The `trimEnd()` removes whitespace from the end of a string

```
const greetStr = '  meow  ';
greetStr.trimEnd(); // '  meow'
```


Thanks, Szabolcs Szabolcsi-Toth (@_Nec) for the initial ideas!

ES2019: Array.prototype.flat() & flatMap()

GIF: Maru flattens himself (https://sisinmaru.com/)

ES2019: Array.prototype.flat()

```
const colors =
      [['black', 'gray', ['orange', 'light orange']], 'bicolor', 'calico'];

const colors1 = colors.flat();
// ["black", "gray", Array(2), "bicolor", "calico"]

// Optional depth level
const colors2 = colors.flat(2);
// ["black", "gray", "orange", "light orange", "bicolor", "calico"]
```

ES11 / ES2020

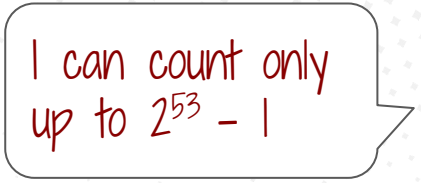
ES11 / ES2020: The Latest

- BigInt
- globalThis
- Dynamic import
- Nullish coalescing operator, `??`
- Optional chaining, `?`
- Promise.allSettled()
- String.prototype.matchAll()
- Module `export`

ES2020: BigInt

JavaScript has two types of number types:

- `Number` (largest number is `Number.MAX_SAFE_INTEGER`, $2^{53} - 1$)
- `BigInt` allows you to use even bigger number!



"Max" the cat

BigCat

ES2020: BigInt

Use suffix, n

```
let max = Number.MAX_SAFE_INTEGER; // 9007199254740991
max++ // 9007199254740992
max++ // 9007199254740992

let largerThanMax = 9007199254740992n;
largerThanMax++ // 9007199254740993n
largerThanMax++ // 9007199254740994n
largerThanMax++ // 9007199254740995n
```

"Max" the cat

BigCat

@girlie_mac

ES2020: BigInt

BigCat

Roarrrrr! BigInt allows JavaScript devs to access the Twitter 64-bit unsigned integers IDs as numbers.

9007199254740992n

@girlie_mac

Well, I'm running out of cat puns so ending this presentation im*purr*fectly.

But I hope you are *feline* good about ES.next by now.

Thank you,

you're *paw*esome!

Tomomi Imura

@girlie_mac