plain concepts

¡El loco, loco, loco mundo de JavaScript!

Ander Conal

18.06.20

VIRTUAL COFFEE

plain concepts

Ander Conal

SOFTWARE DEVELOPER

Cuando no estoy trabajando, lo más posible es que me encuentres haciendo boxeo, paseando con mis 2 perros y el resto de mi familia, jugando un poco a la Switch o quitando el polvo a los vinilos de mi época como promotor de eventos y DJ. En la actualidad, trabajo en Plain Concepts Bilbao donde, además de programar, doy charlas y formaciones.

@anderconal

plain concepts

Agenda completa

1. ES2019
   - Array.flat()
   - Array.flatMap()
   - Object.fromEntries()
   - String.trimStart()
   - String.trimEnd()
   - Function.toString()
   - Object Key Order
2. ES2020
   - Big int
   - Nullish coalescing
   - Optional Chaining
   - promise.allSettled
   - Dynamic import
   - string.matchAll(regex)
   - string.replaceAll
   - globalThis
   - Module Namespace Exports
   - Navigator.share() API
   - Async Hooks
   - Pipline Operator
   - Top Level Await

plain concepts

ES2019

plain concepts

Array.flat()

```
const arr1 = [1, 2, [3, 4]];
arr1.flat();
// [1, 2, 3, 4]

const arr2 = [1, 2, [3, 4, [5, 6]]];
arr2.flat();
// [1, 2, 3, 4, [5, 6]]

const arr3 = [1, 2, [3, 4, [5, 6]]];
arr3.flat(2);
// [1, 2, 3, 4, 5, 6]

const arr4 = [1, 2, [3, 4, [5, 6, [7, 8, [9, 10]]]]];
arr4.flat(Infinity);
// [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

plain concepts

Array.flatMap()

```
let arr1 = [1, 2, 3, 4];

arr1.map(x => [x * 2]);
// [[2], [4], [6], [8]]

arr1.flatMap(x => [x * 2]);
// [2, 4, 6, 8]

// only one level is flattened
arr1.flatMap(x => [[x * 2]]);
// [[2], [4], [6], [8]]
```

plain concepts

Object.fromEntries()

```
const entries = [
  ['foo', 'bar'],
  ['baz', 42]
];

const obj = Object.fromEntries(entries);

console.log(obj);
// expected output: Object { foo: "bar", baz: 42 }
```

plain concepts

String.trimStart()

```
const greeting = '   Hello world!   ‘;

console.log(greeting);
// expected output: "   Hello world!   ";

console.log(greeting.trimStart());
// expected output: "Hello world!   ";


console.log(greeting.trimLeft());
// expected output: "Hello world!   ";
```

plain concepts

String.trimEnd()

```
const greeting = '   Hello world!   ‘;

console.log(greeting);
// expected output: "   Hello world!   ";

console.log(greeting.trimEnd());
// expected output: “   Hello world!";


console.log(greeting.trimRight());
// expected output: “   Hello world!";
```

plain concepts

Function.toString()

```
function sum(a, b) {
  // This comment was not shown with Function.toString
  return a + b;
}

console.log(sum.toString());
// expected output:

"function sum(a, b) {
  // This comment was not shown with Function.toString
  return a + b;
 }“
```

plain concepts

Object key order

```
const obj = {
  'ander': 'conal',
  2: 'plain',
  '1': 'concepts',
  7: 'virtual coffee',
  'loco': 'loco, loco mundo de JavaScript’
}

console.log(Object.keys(obj));
// expected output:  [ "1", "2", "7", "ander", "loco" ]
```

plain concepts

ES2020

plain concepts

BigInt

```
const theBiggestInt = 9007199254740991n

const alsoHuge = BigInt(9007199254740991)
// ↪ 9007199254740991n

const hugeString = BigInt("9007199254740991")
// ↪ 9007199254740991n

const hugeHex = BigInt("0x1fffffffffffff")
// ↪ 9007199254740991n

const hugeBin = BigInt("0b11111111111111111111111111111111111111111111111111111")
// ↪ 9007199254740991n
```

plain concepts

Nullish coalescing operator (??)

```
const foo = null ?? 'default string';

console.log(foo);
// expected output: "default string“

const baz = 0 ?? 42;

console.log(baz);
// expected output: 0
```

plain concepts

Optional chaining (?.)

```
const adventurer = {
  name: 'Alice',
  cat:
    {
      name: 'Dinah'
    }
};

const dogName = adventurer.dog?.name;

console.log(dogName);
// expected output: undefined

console.log(adventurer.someNonExistentMethod?.());
// expected output: undefined
```

plain concepts

Promise.allSettled()

```
const promise1 = Promise.resolve(3);
const promise2 = new Promise((resolve, reject) => setTimeout(reject, 100, 'foo'));
const promises = [promise1, promise2];

Promise.allSettled(promises)
  .then((results) => results.forEach((result) => console.log(result.status)));

// expected output:// "fulfilled"// "rejected"
```

plain concepts

Dynamic Imports

```
// let module = await import('/modules/my-module.js');

const main = document.querySelector("main");
for (const link of document.querySelectorAll("nav > a")) {
  link.addEventListener("click", e => {
    e.preventDefault();

    import('/modules/my-module.js')
      .then(module => {
        module.loadPageInto(main);
      })
      .catch(err => {
        main.textContent = err.message;
      });
  });
}
```

plain concepts

String.matchAll()

```
const regexp = RegExp('foo[a-z]*','g');
const str = 'table football, foosball';
const matches = str.matchAll(regexp);

for (const match of matches) {
  console.log(` Found ${match[0]} start=${match.index} end=${match.index +
match[0].length}.`);
}
// expected output: "Found football start=6 end=14."
// expected output: "Found foosball start=16 end=24."

// matches iterator is exhausted after the for..of iteration
// Call matchAll again to create a new iterator
Array.from(str.matchAll(regexp), m => m[0]);
// Array [ "football", "foosball" ]
```

plain concepts

String.replaceAll()

```
const p = 'The quick brown fox jumps over the lazy dog. If the dog reacted, was it really lazy?';

const regex = /dog/gi;

console.log(p.replaceAll(regex, 'ferret'));
// expected output: "The quick brown fox jumps over the lazy ferret. If the ferret reacted, was it really lazy?"

console.log(p.replaceAll('dog', 'monkey'));
// expected output: "The quick brown fox jumps over the lazy monkey. If the monkey reacted, was it really lazy?"
```

plain concepts

globalThis

```
if (typeof globalThis.setTimeout !== 'function') {
  // no setTimeout in this environment!
}
```

plain concepts

Module namespace exports

```
// Importing a namespace exotic object (existing):

import * as ns from "mod";

// Exporting that name (existing):

import * as ns from "mod";
export {ns};

// Symmetric "export from" (new):

export * as ns from "mod";
```

plain concepts

Web Share API

```
const shareData = {
  title: 'MDN',
  text: 'Learn web development on MDN!',
  url: 'https://developer.mozilla.org',
}

const btn = document.querySelector('button');
const resultPara = document.querySelector('.result');

// Must be triggered some kind of "user activation"
btn.addEventListener('click', async () => {
  try {
    await navigator.share(shareData)
    resultPara.textContent = 'MDN shared successfully'
  } catch(err) {
    resultPara.textContent = 'Error: ' + err
  }
});
```

plain concepts

Async Hooks

```
const fs = require('fs');
const async_hooks = require('async_hooks');
async_hooks.createHook({
  init(asyncId, type, triggerAsyncId) {
    fs.writeSync(1, `Init ${type} resource: asyncId: ${asyncId} trigger: ${triggerAsyncId}\n`);
  },
  destroy(asyncId) {
    const eid = async_hooks.executionAsyncId();
    fs.writeSync(1, `Destroy resource: execution: ${eid} asyncId: ${asyncId}\n`);
 }
}).enable();


const eid = async_hooks.executionAsyncId();
fs.writeSync(1, `Calling setTimeout: execution: ${eid}\n`);
setTimeout(() => {
  const eid = async_hooks.executionAsyncId();
  fs.writeSync(1, `Inside setTimeout: execution: ${eid}\n`);
}, 0);
```

plain concepts

Async Hooks

```
Calling setTimeout: execution: 1
Init Timeout resource: asyncId: 6 trigger: 1
Init TIMERWRAP resource: asyncId: 7 trigger: 1
Inside setTimeout: execution: 6
Destroy resource: execution: 0 asyncId: 6
Destroy resource: execution: 0 asyncId: 7
```

plain concepts

Pipeline operator (|>)

```
const url = "%21" |> decodeURI;

const url = decodeUri("%21“);

const double = (n) => n * 2;
const increment = (n) => n + 1;

// without pipeline operator
double(increment(double(double(5)))); // 42

// with pipeline operator
5 |> double |> double |> increment |> double; // 42
```

plain concepts

Top level await

```
// Before Top level await

await Promise.resolve(console.log('🥕'));
// → SyntaxError: await is only valid in async function

(async function() {
  await Promise.resolve(console.log('🥕'));
  // → 🥕
}());


// After Top level await
await Promise.resolve(console.log('🥕'));
// → 🥕
```

plain concepts

Top level await

```
// Dynamic dependency pathing
const strings = await import(`/i18n/${navigator.language}`);

// Resource initialization
const connection = await dbConnector();

// Dependency fallbacks
let jQuery;
try {
  jQuery = await import('https://cdn-a.example.com/jQuery');
} catch {
  jQuery = await import('https://cdn-b.example.com/jQuery');
}
```

Thank you

@anderconal

@plainconcepts

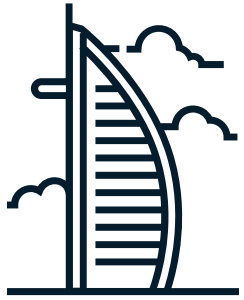
www.plainconcepts.com

plain concepts

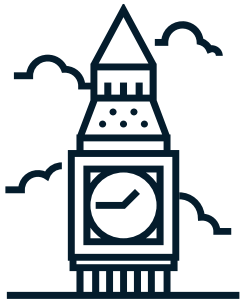
Rediscover
the meaning of technology

SPAIN

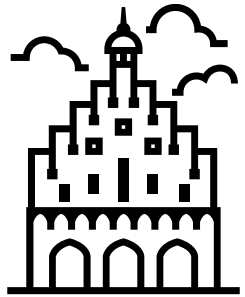
USA



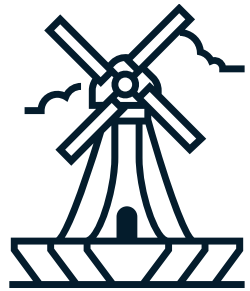
UAE



UNITED KINGDOM



GERMANY



NETHERLANDS

www.plainconcepts.com

For further information

info@plainconcepts.com