ES7 & ES8 新特性

爽

ECMAScript 制定流程

1、由TC39委员会制定标准

2、制定流程一共有5个stage，全部通过了才可以成为最终标准

1）Stage 0（Strawman阶段），可以提交各种想法

2）Stage 1（Proposal阶段），是对提交新特性的正式建议

3）Stage 2（Draft阶段），出现标准中的第一个版本，有完整描述（有两个实现性实现）

4）Stage 3（Canidate阶段），提议接近完成，需要得到提议实现方的反馈（至少通过两个测试用例）

5）Stage 4（Finished阶段），提议会被包含的标准之后（审核通过）

ECMAScript 历史

1、ES5　　2009年12月发布

ES5.1　　2011年6月发布

2、ES2015 (ES6)　　2015年6月发布（大改进，还了以前的债）

从此之后新标准的制定开始规划化，每年一小更；
成文标准要从实现标准中诞生，即要先实现了，才有可能成为规范

3、ES2016 (ES7)　　2016年3月发布

4、ES2017 (ES8)　　2017年6月底发布

ES7 新特性

1、Array.prototype.includes

用法很容易简单，用于判断数组中是否含有某元素，返回true或false。第二个参数可选，允许从特定位置开始匹配

例：

```
['a','b','c'].includes('a') // true
['a','b','c'].includes('d') // false
['a','b','c'].includes('a', 1 ) // false
```

arr.includes(x) 等价于 arr.indexOf(x) >= 0

有一点不同的是includes()方法可以找到NaN，indexOf()不行

```
[NaN].includes(NaN)   // true
[NaN].indexOf(NaN) >= 0   // false
```

ES7 新特性

2、求幂运算符

也是比较简单，之前可以使用Math.pow()来进行求幂运算，现在可以用 **

例：

```
let a = 7 ** 2 // 49
let a = 7
a **= 2;  // 49

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

ES8 新特性

1、Object.values/Object.entries

和Object.keys()类似，Object.values 方法则会返回指定对象的可枚举的属性值数组，数组中值顺序与 for-in 循环保持一致

Object.entries 方法则会将某个对象的可枚举属性与值按照二维数组的方式返回

例：

```
> let obj = {a:111, b: 'ccc' }
  Object.values(obj)
▶ (2) [111, "ccc"]
> Object.entries(obj)
▼ (2) [Array(2), Array(2)]
  ▼ 0: Array(2)
      0: "a"
      1: 111
      length: 2
    ▶ __proto__: Array(0)
  ▼ 1: Array(2)
      0: "b"
      1: "ccc"
      length: 2
    ▶ __proto__: Array(0)
    length: 2
  ▶ __proto__: Array(0)
```

ES8 新特性

2、Object. getOwnPropertyDescriptors

该方法返回对象Obj所有自身的属性描述，是Object. getOwnPropertyDescriptor的多参数版。主要是配合Object.create进行一个对象的浅拷贝

```
Object.create(
  Object.getPrototypeOf(obj),
  Object.getOwnPropertyDescriptors(obj)
);
```

例：

```
> let obj = {
      a: 111,
      b: 'ccc'
  }
  Object.getOwnPropertyDescriptors(obj)
◀ ▼Object {a: Object, b: Object}
    ▼a: Object
        configurable: true
        enumerable: true
        value: 111
        writable: true
      ▶__proto__: Object
    ▼b: Object
        configurable: true
        enumerable: true
        value: "ccc"
        writable: true
      ▶__proto__: Object
    ▶__proto__: Object
>
```

ES8 新特性

3、函数参数列表和调用中的尾逗号

在之前，函数参数带有尾逗号是会报错的，现在取消了这种处理，向对象的尾逗号看齐

```
function foo(a,
  b,
  c,
  d,) { //  现在允许了
  // ...
  console.log(d)
}
```

ES8 新特性

4、String.prototype.padStart/ String.prototype.padEnd

padStart()在开始部位填充，返回一个给出长度的字符串，第二参数用于指定填充的字符串，默认为空格。

padEnd()同理，在尾部开始填充。

例：

```
> 'aaa'.padStart(10)
<· "       aaa"
> 'aaa'.padStart(10, 'b')
<· "bbbbbbbaaa"
> 'aaa'.padEnd(10)
<· "aaa       "
> 'aaa'.padEnd(10, '123')
<· "aaa1231231"
>
```

ES8 新特性

5、async/await

async/await终于被纳入ES标准中。

使用方法：

1）async 表示这是一个async函数，await只能用在这个函数里面

2）await 表示在这里等待promise返回结果了，再继续执行。

3）await 后面跟着的应该是一个promise对象（不是则立即执行）

ES8 新特性

例：Promise写法

```
> function awaitFn(ms) {
    return new Promise((resolve, reject) => {
      setTimeout(() => {resolve(`${ms/1000}s`)}, ms)
    })
  }

  function foo() {
    const timePro = awaitFn(2000)
    timePro.then((time) => {
      console.log(`run after ${time}`)
    })
  }

  foo()
< undefined
  run after 2s
>
```

例：async/await写法

```
> function awaitFn(ms) {
    return new Promise((resolve, reject) => {
      setTimeout(() => {resolve(`${ms/1000}s`)}, ms)
    })
  }

  async function foo() {
    let time = await awaitFn(2000)
    console.log(`run after ${time}`)
  }

  foo()
< ▶ Promise {[[PromiseStatus]]: "pending", [[PromiseValue
  run after 2s
>
```

看起来就像是同步函数

ES8 新特性

6、共享内存与原子操作

此功能引入了一个新的低级别Atomics命名空间对象和一个SharedArrayBuffer构造函数，来作为更高级别并发抽象的原始构建块。这使开发人员能够共享多个service worker和核心线程之间的SharedArrayBuffer对象的数据。

PS：这个特性比较复杂，我们也不那么常用
详细的介绍可见：
https://github.com/tc39/ecmascript_sharedmem/blob/master/TUTORIAL.md
https://segmentfault.com/a/1190000009878588

参考文章

1、JavaScript(ECMAScript) 语言标准历史及标准制定过程介绍
2、ECMAScript 2017（ES8）特性概述
3、ES8 的 5 个特性以及对 ES9 的展望 - Blog | SitePen
4、tc39已完成的提案

THANKS

FOR YOUR WATCHING

02

OPEN ORIENTED

凹 凸 实 验 室