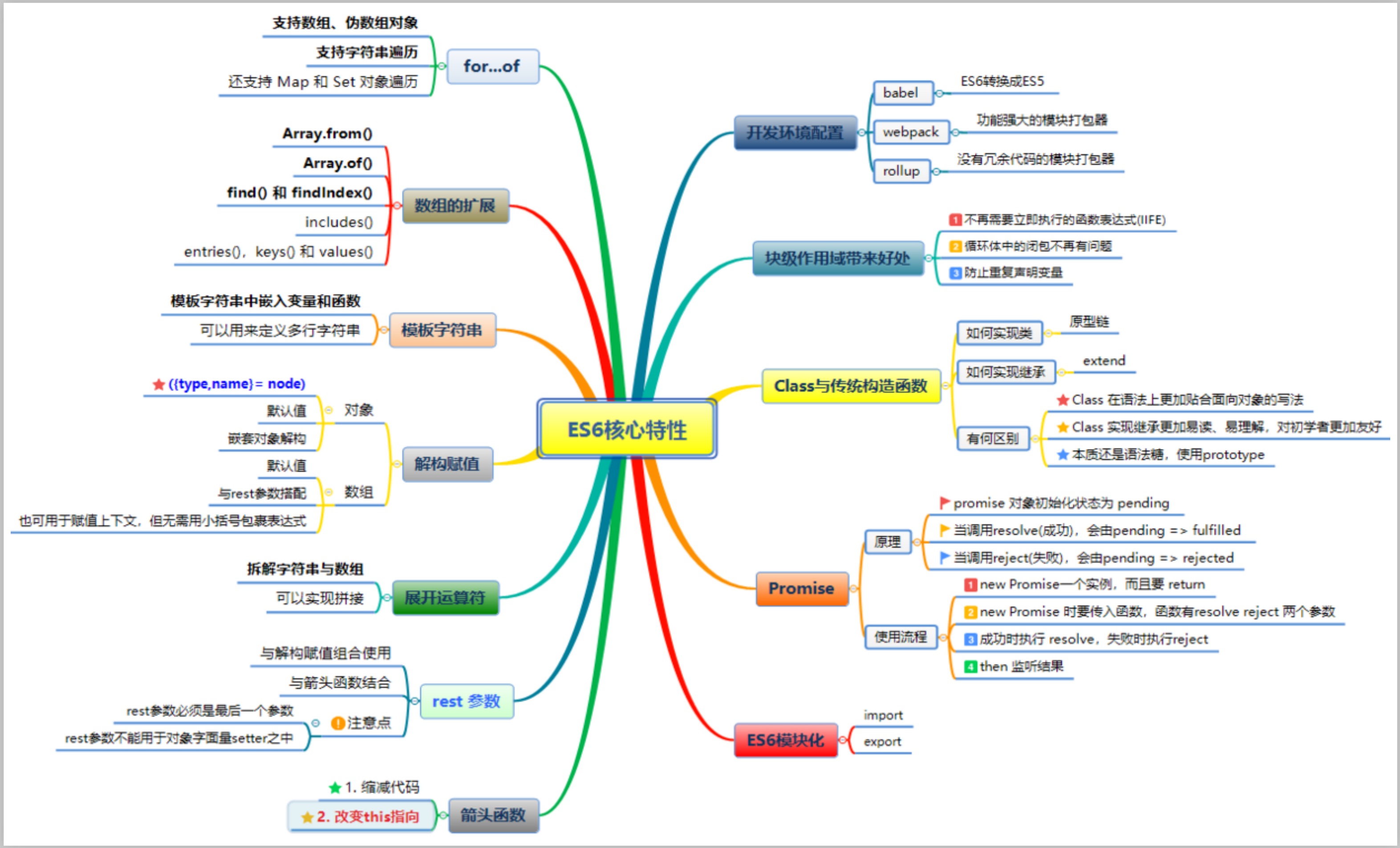
ES6 core features

wulei6@xiaomi.com

structure preview


ES6核心特性
for...of
支持数组、伪数组对象
支持字符串遍历
还支持 Map 和 Set 对象遍历
数组的扩展
Array.from()
Array.of()
find() 和 findIndex()
includes()
entries()，keys() 和 values()
模板字符串
模板字符串中嵌入变量和函数
可以用来定义多行字符串
解构赋值
对象
({type,name} = node)
默认值
嵌套对象解构
数组
默认值
与rest参数搭配
也可用于赋值上下文，但无需用小括号包裹表达式
展开运算符
拆解字符串与数组
可以实现拼接
rest 参数
与解构赋值组合使用
与箭头函数结合
注意点
rest参数必须是最后一个参数
rest参数不能用于对象字面量setter之中
箭头函数
1. 缩减代码
2. 改变this指向
开发环境配置
babel
ES6转换成ES5
webpack
功能强大的模块打包器
rollup
没有冗余代码的模块打包器
块级作用域带来好处
1 不再需要立即执行的函数表达式(IIFE)
2 循环体中的闭包不再有问题
3 防止重复声明变量
Class与传统构造函数
如何实现类
原型链
如何实现继承
extend
有何区别
Class 在语法上更加贴合面向对象的写法
Class 实现继承更加易读、易理解，对初学者更加友好
本质还是语法糖，使用prototype
Promise
原理
promise 对象初始化状态为 pending
当调用resolve(成功)，会由pending => fulfilled
当调用reject(失败)，会由pending => rejected
使用流程
1 new Promise一个实例，而且要 return
2 new Promise 时要传入函数，函数有resolve reject 两个参数
3 成功时执行 resolve，失败时执行reject
4 then 监听结果
ES6模块化
import
export

let & const: block-scoped

```
let x = 1
function getNum() {
    if(false) {
        var x = Math.random()
        //let x = Math.random()
        return x
    }

    return x
}
getNum()
```

let & const: block-scoped

```
for(var i = 0; i < 5; i++) {
    setTimeout(function() {
        console.log(i)
    })
}

for(let i = 0; i < 5; i++) {
    setTimeout(function() {
        console.log(i)
    })
}
```

let & const: TDZ

```
{
    // TDZ start
    console.log(a)

    let a = 1    // TDZ end
}
```

let & const: how to use

```
{
    const immutableVar = 123
    const immutableObj = {
        a: 1
    }

    for(let i = 0; i < 5; i++) {}
}
```

advice: const > let > var(avoid var)

template literals

```
const name = 'leo'
console.log(`My name is ${name}`)

const mix = `I'm a "teacher" named ${name}`
const html =
`
    <html>
        <body>hello ${name}</body>
    </html>
`
```

arrow functions: syntax

```
var oldFunc = function(v) {}

const fun1 = () => {}        // no param
const fun2 = (v1, v2) => {} // several params
const fun3 = v => {}         // one param
const fun4 = (a,b) => a + b // only a expression
const fun5 = (v1, v2) => {
    v3 = 1                  // several statements
    return v1 + v2 + v3
}
const func6 = () => ({name: 'leo'}) // return an obj
```

arrow functions:

tradional functions have a dynamic this

while arrow functions have a lexical this

```
function Person() {
    this.age = 0

    setInterval(function() {
        this.age++
    }, 1000)
}
var p = new Person()
```

arrow functions: this

```
function Person() {
    var that = this
    this.age = 0
    setInterval(function() {that.age++}, 1000)
}
function Person() {
    this.age = 0
    setInterval(function() {this.age++}.bind(this), 1000)
}
// arrow functions do not have its own this
function Person() {
    this.age = 0
    setInterval(() => {this.age++}, 1000)
}
```

arrow functions: how to use

advice: 1.shorter functions 2. no existence of this keyword avoid:1.call,apply 2.arguments 3.method function 4.new 5.prtotype 6.yield

proxy

=>

promise

generator

class

module