一、ECMASCRIPT3/5开发中存在的问题：

1. 作用域只有function，变量容易冲突
2. Function（）{}匿名函数，写function
3. 遍历数组，对象麻烦
4. 对面向对象编程支持不好
5. 代码可读性差，可维护性差

二、ECMASCRIPT6/7/8/9/10

1. 引入块作用域减少冲突

2. 匿名函数（）=>

3. 对象和数组解构赋值

4. 支持面向对象编程—类 构造器 继承

5. 改写异步，提升代码可读性 可维护性

6. 隐式迭代 链式写法

7. 引入新语法 提升开发效能 --装饰器语法 **await async异步改写**

**说明：ES6大部分语法是ES3/5的语法糖**

**语法糖：换一种写法 比如less就是css的语法糖**

**Coffeejs 就是js语法糖**

**三、ECMAscript和Javascript之间的关系**

**1. ECMAscript是Javascript的标准**

**2.javascript是ECMAscript的具体实现**

**四、 let const 块作用域 {}就是个块 相比较函数的作用域小**

**1. var不受到{}限制 var只受到functuion的限制**

**2. let不允许生名多个同名变量**

**For块 普通的块{} while块**

**五、js的作用域**

**1. 全局作用域**

**2. 局部作用域**

**3. 块作用域—ES6新增**

**If块 while块 函数块function（）{} for循环块 单独块{}**

**块作用域使用let或者是const**

**let和const都是块作用域**

**六、 let用来创建块作用域的变量的**

**const用来创建块作用域的常量的**

**七、 const 创建常量：不能被修改的变量—除了常量的定义出**

**开发场景： 有一个很重要的变量 不想后面被覆盖 这时候定义个常量**

1. **ECMAscript存在的问题**
   1. **作用域只有functuion，变量容易冲突**
   2. **Function（）{}匿名函数，总function**
   3. **遍历数组对象麻烦**
   4. **面向对象编程支持不好**
   5. **代码可读性差，可维护性差**
2. **ECMASCRIPT6/7/8/9/10**
   1. **引入块作用域**
   2. **匿名函数（）=>{}**
   3. **对象和数组解析赋值**
   4. **支持面向对象编程 类 构造器 继承**
   5. **Promise改写异步 提升可维护性 可读性**
   6. **隐式迭代链式写法**
   7. **引入新语法 提升开发效能 比如装饰器**

**说明：ES6大部分语法是ES3/5的语法糖**

**语法糖：就是换一种写法**

1. **ECMASCRIPT和JAVASCRIPT之间的关系**
   1. **ECMASCRIPT是javascript的标准**
   2. **JavaScript是ecmascript的具体实现**
2. **块作用域** 
   1. **let 同一个块内不允许重声明**
   2. **const**
   3. **{}**
3. **快的类型**
   1. **for循环块**
   2. **while块**
   3. **if块**
   4. **{}单独块**
   5. **函数块**

**说明：块作用域使用let const**

1. **const用来定义常量—不能被修改的变量（除了变量定义的时候）**

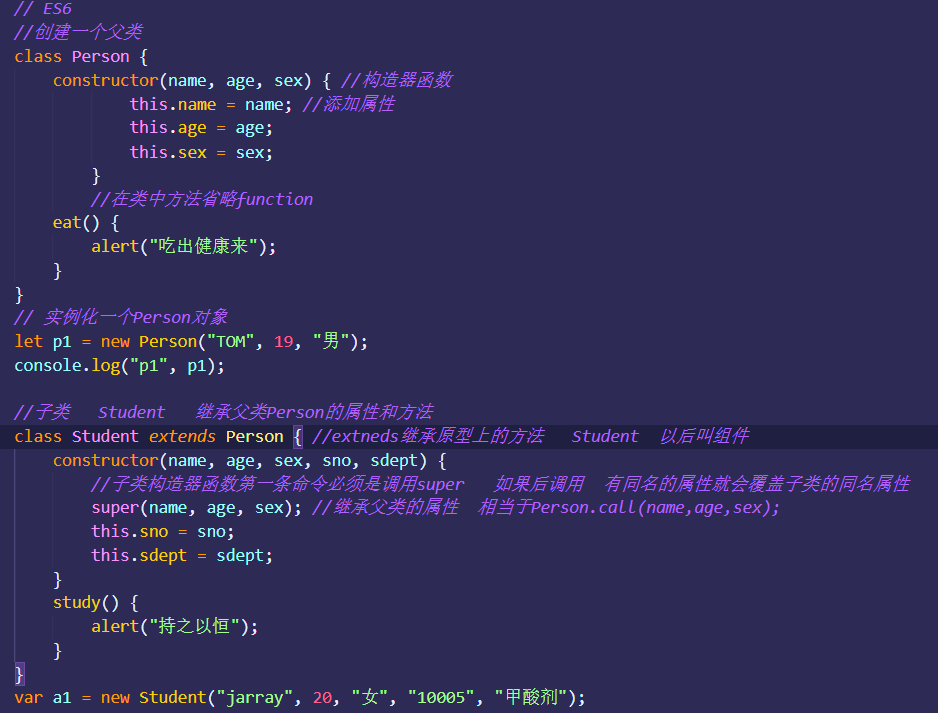
**let来创建块作用域变量的**

**let 和 const的作用域都是块作用域**

**七、箭头函数返回值是对象的时候，后面添加一个小括号**

1. **ES6转换成ES5**
   1. **babel js转换器**
2. **箭头函数和普通的函数的区别---面试题**
   1. **箭头函数是匿名函数**
   2. **箭头函数写法简洁 没有functuion**
   3. **自带bind（this） 指向外部的对象**
   4. **箭头函数不能用作Generator函数，不能使用yield关键字**
   5. **箭头函数不能用作构造器函数**
   6. **不能使用arguments callee caller**
3. **React组件分类：** 
   1. **类组件----class**
   2. **静态组件----function**
4. **ES6中引入面向对象编程 类 构造器函数 继承的概念**
   1. **class 类 创建一个类**
   2. **constructor构造器 new类 的时候自动调用**
   3. **super超级 超类 继承父类的属性 相当于父构造器.call（this）**
   4. **extends继承 方法继承**

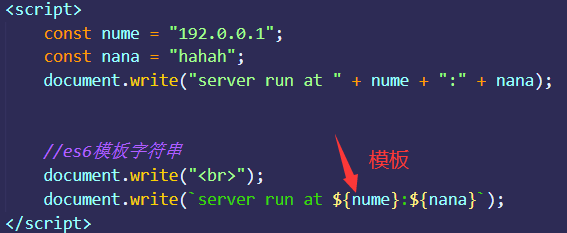
**ES6继承**



**十一、 \_ \_ proto\_ \_ 对象继承**



**十二、模板字符串**



**${}是模板**

**{} react里面的模板**

**{{}} vue里面的模板**

**十三、 前后端分离开发方案**

**方案1. Ajax**

**方案2. 后端模板 ：把前端代码和后端代码php/java等分开写，然后动态合并**

**前端人员：html/.vm 静态页面---也叫作模板页**

**问题：页面显示动态内容荣---来自后端**

**解决办法：使用模板语法**

**.html${userName}运行的时候，php后端模板，将模板替换成“tom”**

**复杂数据：模板for循环**

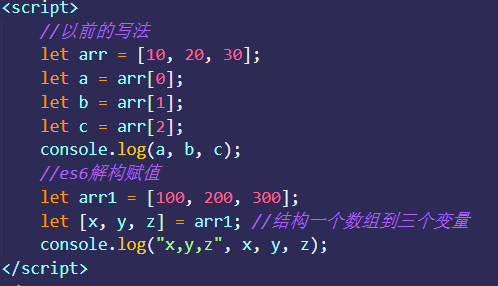
**在react中{} 大括号就是模板 vue{{}} 这就是模板**

**jsx语法指的是在xml中嵌入js语法**

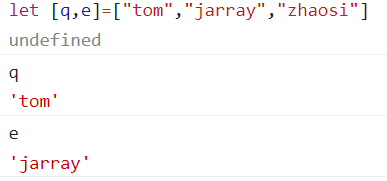
**十四、 解构赋值----重点-重点-重点**

**他是啥意思呢？就是将数组元素或对象的指定的值，赋值给指定的变量**

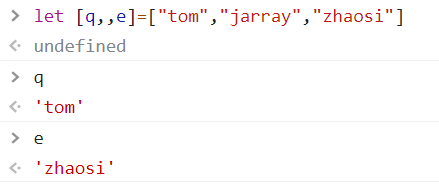
**分类：**

1. **数组的解构赋值- - 将数组元素指定的值，赋值给指定的变量**

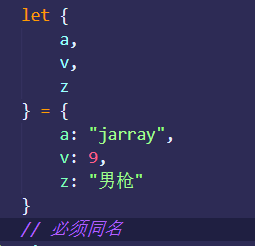
**变量长度和数组长度不一致**



**我想要 e =zhaosi**



1. **对象的解构赋值---将对象指定属性的值，赋值给指定的同名变量**



**对象结构中的rest解构**

**十五、 不定参数**

**十六、for in for of ---记住**

1. **for of 遍历的是值 而不是键值 不管是什么对象，只要是部署了Symbol.iterator接口 就可以遍历**
2. **for in 遍历数组或对象的键名**

**说明：啥是遍历---就是将数组的元素或者是对象的属性，一个一个拿出来进行显示或者是处理**

**十七、 set 唯一性 可以用来数组去重**

**set内部如何判断是否相等？**

**内部结构用的是=== 绝对等于 除了NaN**

**set对象个数 size**

**add（） 添加某个值**

**delete（）删除某个值**

**has（） 判断是否有这个值**

**clear（）清除所有**

**十八、 垃圾回收机制---会定期对那些我们不再使用的变量、对象所占用的内存进行释放**

**重点 标记清楚和引用计数**

1. **Js采用自动垃圾回收机制**
2. **变量生命周期结束，内存就会被回收**
3. **全局变量：生命周期一直延续，知道页面卸载**
4. **局部变量：函数调用结束，局部变量不再使用，占用的内存也就回收了**
5. **闭包：由于闭包原因，局部变量一直在使用，所以就不会回收**

**循环引用的时候计数永远都是不是0 需要手动清除**

**十九、 内存泄漏的问题---不再被需要的内存，由于某种原因，无法被回收**

**常见场景：**

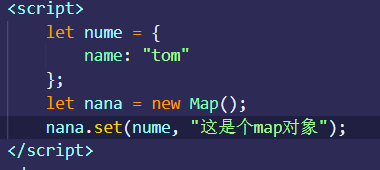
1. **全局变量造成的**
2. **没有销毁的定时器和回调函数**
3. **闭包造成的内存泄漏**
4. **DOM引用造成的内存泄漏**

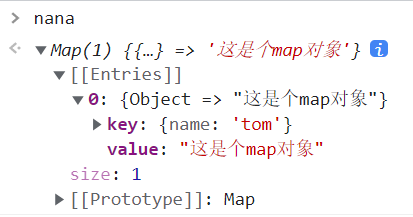
**二十、 weakest 和set类似**

1. **Weakest参数只能是对象，不能是其他的**
2. **Weakest都是弱引用 它是不可遍历的**

**用处：存储DOM节点，不用担心这些节点从文档移除是会引发内存泄漏**

**二十一、 Map**





**获得nume这个键的值**

**nana.get(nume)**

**判断nana里面是不是有nume这个值**

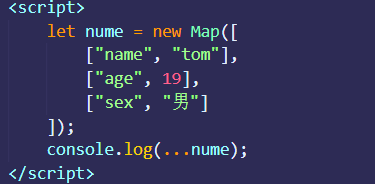
**nana.has(nume) 有返回true 没有false**

**删除nana中的nume**

**nana.delete(nume)**

**快速创建map**

**map拓展成数组**

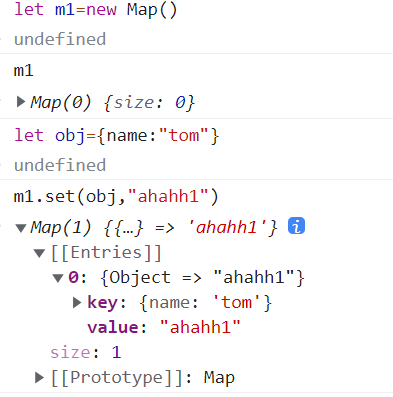


**Map转json map先转对象 然后转json**

**二十二、 iterator 遍历器 ---统一给（array， map，set）提供统一的遍历方式for of 使用的for of的时候就会自动寻找iterator，只要这些对象部署了Symbol.iterable属性，for of就可以自动遍历成员**

**二十三、 Map对象 ---值：值 对**

**set添加成员**



**二十四、 Generator函数：是一个状态机，封装了多个内部状态 还是一个遍历器对象生成函数，调用generator函数会创建一个遍历器对象**

**和普通函数的区别：1. function 在和函数匿名之间有个\***

**2. 有yield（产出）**

**作用：改写异步promise 改成同步写法**

**解决：解决地域回调问题**

**二十五、 严格模式：面试题**

1. **IE10+支持**
2. **消除代码运行不安全**
3. **提高效率**
4. **为新版本javascript做铺垫**

**开启严格模式 “use strict”**

**规则：严格模式下都必须先加var 然后再定义变量**

**二十六、 with**

**二十七、 作用域**

1. **正常模式下有两种作用域:**
   1. **全局作用域**
   2. **函数作用域**
2. **严格模式下作用域**
   1. **全局作用域**
   2. **函数作用域**
   3. **Eval作用域**

**二十八、 严格模式下的this**

1. **严格模式下this没有指向 指向undefined**
2. **使用构造器函数时候 内有写new this也指向undefined**

**二十九、 严格模式禁止删除变量 之后configurable为true的时候才可以删除**

**三十、 模块化**

**Es5 导出 module.exports 加载 require(“./模块名”)**

**ES6 导出 export{变量 对象 函数} 导入 inport{变量 对象 函数} from ‘./模块名’ 导出的名字和导入的名字要一致**

**默认导出 export default 导入import from‘./模块化’**

**三十一、webpack：模块打包加载工具**

**webpack ./源文件.js mode=development -output ./dist/main.js**

**三十三、导出和导入**

|  |  |  |
| --- | --- | --- |
| **ES5** | **导出 module.exports** | **加载 require(“./模块名”)** |
| **ES6** | **导出export{变量 函数 对象}**  **（导出和导入的名字要一致）** | **导入 import{变量 函数 对象} from ‘./模块名’** |
| **ES6** | **默认导出 export default**  **注意：一个模块只能有一个默认导出，导入对象名字随便起** | **导入 import from ‘./模块化’** |

**三十四、 webpack就是模块加载以及打包工具**

**三十五、 路由组件**

**三十六、proxy 代理 拦截器**

**三十七、js数据类型种类 7中**

1. **Number 数字**
2. **String 字符串**
3. **Boolean 布尔**
4. **Undefined 空**
5. **Object 对象**
6. **Function 函数**
7. **Symbol 符号**

**三十八、symbol**

**三十九、ES6新增的string的API**

1. **Es5中的indexof（子字符串）：在字符串中搜索子字符串 从左到右 返回第一个匹配的位置 如果搜索不到返回-1**
2. **includes（）是否包含指定字符串 返回布尔值**
3. **startsWith（）表示指定字符串是否在字符串头部 返回布尔值**
4. **endsWith（）表示指定字符串是否在尾部 返回布尔值**
5. **repeat（）将原字符串重复**

**四十、 多行字符串**

**使用模板字符串 ``**

**四十一、 number新增API**

1. **toFixed(保留小数的为数) 将数值转换成字符串**

**误差值：Number.EPSILON 如果0.1+0.2的误差值小于这个误差值 就是认为是正确的**

1. **includes 是否包含**

**四十二、 promise 改写异步ajax 可维护性差 可读性差 所以才改**

**Promise存在地域回调问题**

**Promise三个状态：pending进行中**

**Fulfilled成功**

**Rejected失败**

**一旦进入某个状态就无法改变**

**四十三、 Promise Api**

**Promise对象.then（res=>{}）成功api**

**Promise对象.catch（error=>{}）失败api**

**Promise对象.all（[“url1”, “url2”, “url3”]）同时发送多个请求 谁慢以谁为准**

**Promise对象.race是谁快以谁为准**