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）**

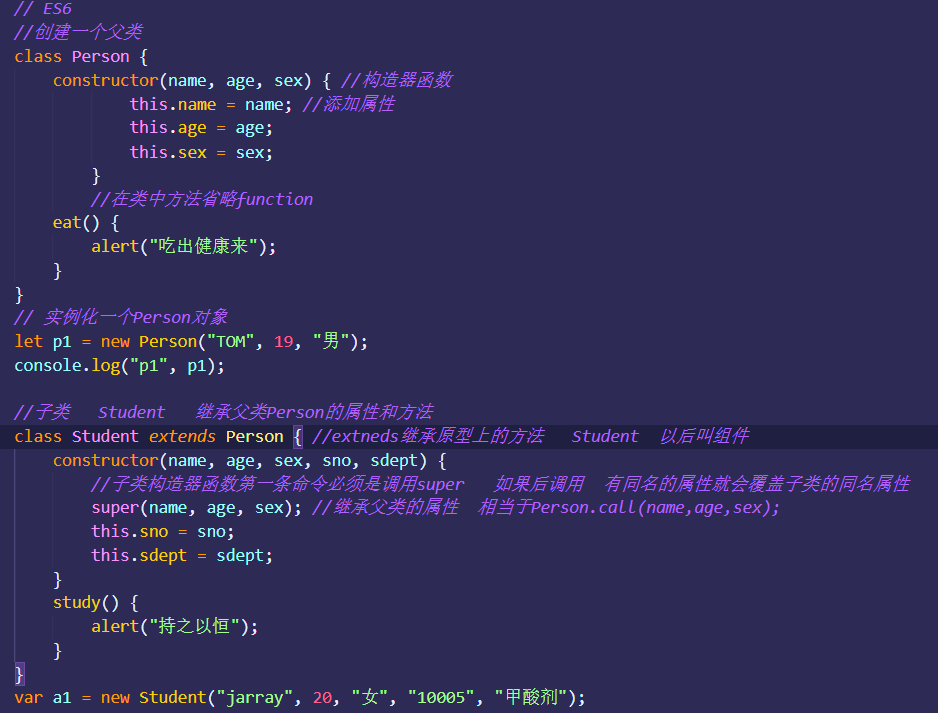
**let来创建块作用域变量的**

**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**
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**