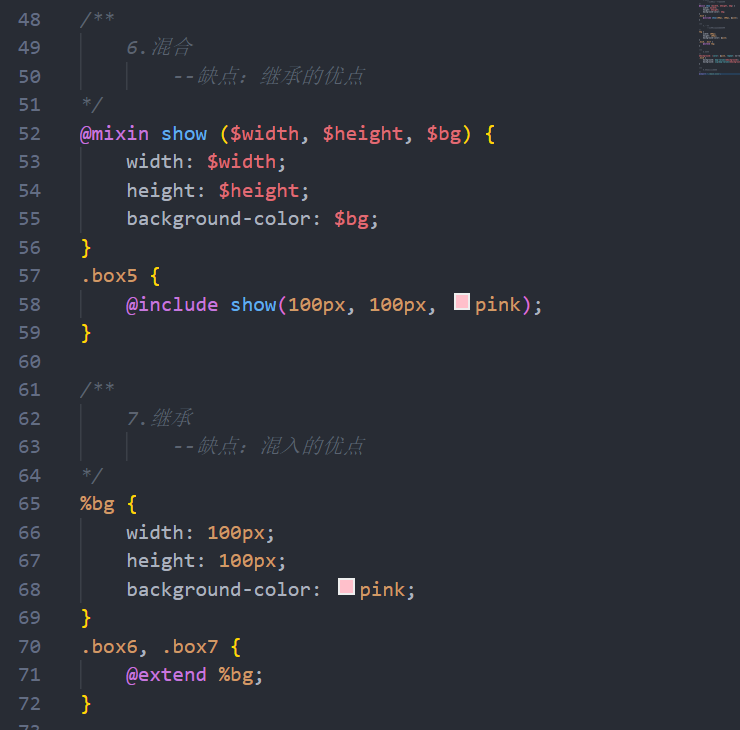
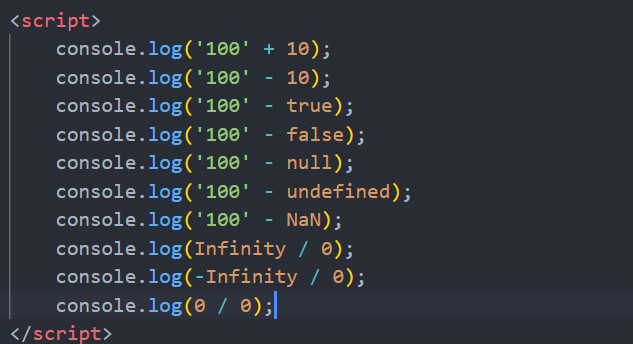
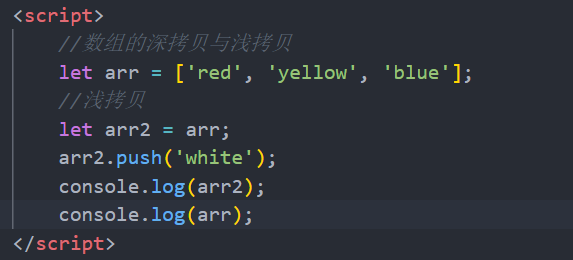
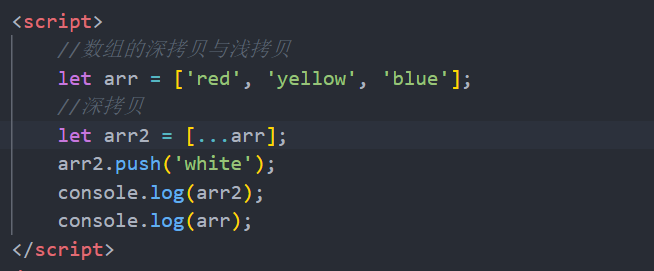
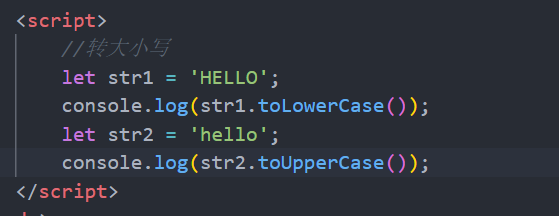
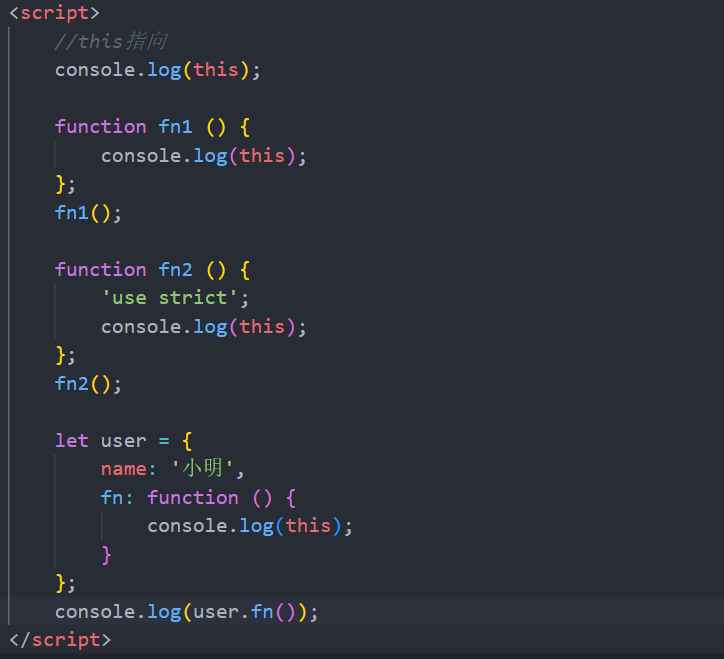
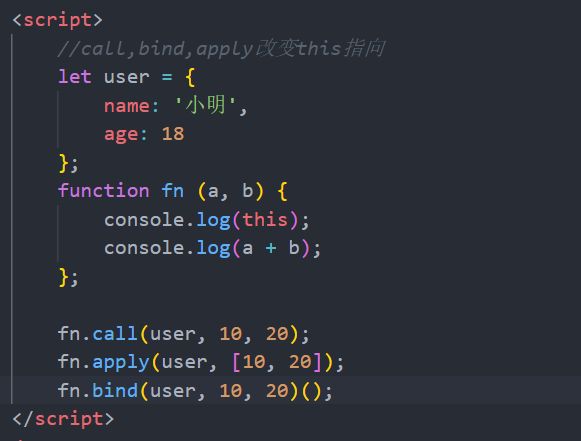
### JS-3

1. Scss高阶用法
   1. 
   2. 
   3. 
   4. 
2. JS
   1. 十进制转二进制
      1. 模二取余
      2. 8的二进制是1000
   2. 自动类型转换
      1. 字符串型 + 任意数据类型 = 字符串型
      2. 数字字符串型 - 数值型 = 数值型
      3. 数字字符串型 - true/false = true为1 false为0
      4. 数字字符串型 - null/undefined/NaN = null为0 undefined/NaN为NaN
      5. 正无穷/0 = 正无穷 负无穷/0 = 负无穷 0/0 = NaN
      6. 
   3. 强制类型转换
      1. 转数值型 Number()
      2. parseInt(‘1000’, 2); 2进制转10进制
   4. 数组反转 reverse
      1. 
   5. 数组的深拷贝与浅拷贝
      1. 
      2. 
      3. 
   6. 字符串的大小写转换
      1. toLowerCase() 转小写
      2. toUpperCase() 转大写
      3. 
   7. 事件委托
      1. 给父元素绑定事件，去获取子元素的相关信息
      2. 
3. JS高级
   1. this指向
      1. 默认指向window
      2. 在严格模式下，普通函数中的this指向undefined；在非严格模式下，普通函数中的this指向window
      3. 对象中的this指向对象
      4. 箭头函数中的this是由其父级决定
      5. 
   2. Call，bind，apply改变this指向
      1. Call改变this指向，单个传参
      2. bind改变this指向，单个传参
      3. apply改变this指向，数组传参
      4. 
   3. 原型
      1. 创建对象的方式
         1. Let obj = new Object()
            1. 缺点：需要对对象的属性和方法进行单独设置
         2. Let obj = {}
            1. 缺点：不能批量的对对象的属性和方法进行设置
         3. 工厂模式
            1. 缺点：不能具体的指向某一类对象
         4. 构造函数模式
            1. 缺点：同一属性和方法会挂载到多个实例中，存在浪费内存的问题
         5. 原型
            1. 优点：解决以上问题
      2. 原型链
         1. 找对象的属性和方法的时候，先从实例对象上找，如果没找到，通过\_\_proto\_\_去构造函数上找，如果没找到，通过\_\_proto\_\_去Object上找，如果没找到，返回undefined，Object的原型是null
   4. 执行上下文
      1. 预编译前：会创建执行上下文环境
      2. 执行上下文环境可以分为：全局执行上下文和局部执行上下文（函数的创建）
      3. 执行上下文环境会确定对应的变量，函数，作用域以及this指向
      4. 执行上下文是动态的，作用域是静态的
   5. JS事件的轮询机制