JavaScript的组成

- BOM
  - 浏览器对象模型
    - 路由 vue
    - 浏览器的地址栏
    - 定时器
      - 间歇调用 setInterval（function（）{}，1000）
        - 每隔一定时间之后再去执行函数
        - 参数
          - 第一个参数 需要执行的函数（必填）
          - 第二个参数 时间（可选）:毫秒
        - 返回值：id
          - 通过id可取消定时器，clearInterval(id)
      - 超时调用 setTimeout（function（）{}，1000）
        - 经过一定时间之后再去执行函数
        - 参数
          - 第一个参数 需要执行的函数（必填）
          - 第二个参数 时间（可选）:毫秒
        - 返回值：id
          - 通过id可取消定时器，clearTimeout(id)
    - 会话框
      - alert 警告框，提示用户，阻塞代码运行，在alert下方的所有代码，只有在点击了alert的‘确定’之后才能使用
      - confirm
      - promopt
    - Location对象
      - 属性
      - 方法
        - assign(url) 打开url
        - replace(url) 打开url
        - reload() 重新加载当前页面
    - 原生ajax
      - 介绍

- ajax是一种用于向后台发送请求的技术，可以实现动态网页的构建，也可以实现页面数据的局部刷新
- 特点
  - 页面局部刷新
- 语法
  - XMLHttpRequest（）
    - 专门用于向后台发送数据
  - open（'GET',url）
    - 设置请求方式以及请求地址
  - send();
    - 将请求发送给后台
  - responseType
    - 设置相应数据的格式
  - onreadystatechange
    - 监听响应
  - readyState
  - status
    - 接口响应状态码
      - 200
      - 404
      - ...
  - response
    - 相应数据
- jquery的ajax
  - 引入jquery
    - bootcdn
  - 语法
    - 底层ajax
    - 快捷方式

- DOM
  - 文档对象模型
    - js操作HTML元素
      - DOM树
        - Node节点（根节点）
          - Element 元素节点
          - Document 文档节点

        - Comment 注释节点
        - Text 文本节点
- Document 节点
    - 属性
        - ...
    - 方法
        - getElementById('id'); 通过id获取元素
        - getElementsByClassName('class'); 通过class获取元素
        - getElementsByTagName('标签名'); 通过标签名获取元素
        - getElementsByName('name'); 通过name属性获取元素
        - 对于类数组对象以及节点数组，都可以使用【索引】来访问获取到节点
- Element节点
    - 属性
        - innerHTML
            - 获取标签内部的html内容代码（包括标签换行）
        - innerText
            - 获取标签内部文本
    - 方法
        - getAttribute（'属性名'）
            - 获取属性
        - setAttribute（'属性名'，'对应值'）
            - 设置属性
        - removeAttribute('属性名')
            - 删除属性
- 节点操作
    - 节点创建
    - 节点追加
    - 节点删除
    - 节点复制
- 事件机制
    - 事件三要素
        - 事件目标
            - 需要绑定事件的元素，事件源
        - 事件处理程序
            - 一般是一个函数，当绑定在事件目标上的事件类型触发的时候，执行该函数
              事件类型：click、mouseover、mouseout、blur、focus、scroll。。。

- 事件对象
  - 保存了事件的详细信息，一般用event.target表示
- 事件流
  - 页面元素接受时间顺序
    - 前提
    - 事件冒泡
      - 阻止
    - 事件捕获
- 事件绑定
  - on
    - dom.onxxx = function（）{}
  - addEventListener('xxx')
    - dom.addEventListener('xxx',function(){})
- 事件类型
  - UI事件
  - 焦点事件
  - 鼠标与滚轮事件
  - 键盘与文本事件
- 事件代理
  - 用法：将原本绑定在子元素上的事件，绑定在了父元素上
  - 好处：后期通过节点创造的方式新增的节点，也可以拥有该事件
  - 案例：光标悬浮到li上的时候，该li有背景色，移出时消失
- 绑定事件
- 事件监听
- ……

- ES5
  - ECMAscript5
    - script标签不受同源策略限制
      - script标签
        - defer属性：延迟到文档完全被解析和显示之后在执行。只对外部脚本文件有效。（多个defer标签按顺序执行）
        - async属性：表示应该立即开始下载，但不能阻止其他页面动作，比如下载资源或等待其他脚本加载。只对外部脚本文件有效。（标记为async的脚本并不保证能按照它们出现的次序执行）
    - 变量
      - 是一个容器，可以存放任意类型的数据
      -

弱类型和强类型

- 强类型
  - int a = 1； a = false；//错的
- 弱类型
  - var a = 1; a = false; ///可行

- 变量的声明 var a，b，c；
- 变量的初始化 a = 1；字符，数组
  - 变量的申明和初始化可以同时写
  - 如果变量只声明，但是未初始化，打印时则为undefined
  - 变量命名的规则
    - 数字、字母、下划线、$组成
    - 不能以数字、下划线开头
    - 驼峰命名法
      - var zhangsanName
    - 不能使用关键字和保留字
- 数据类型
  - 基本数据类型（保存在栈区）
    - 数字
      - Number
        - 整数
          - var a = 1；
        - 浮点数
          - var b = 1.2；
        - 非数字
          - var c = 10/'hello'; // isNaN
          - 判断是否是非数字
            - isNaN
    - 字符串
      - String
        - json字符串
          - var c ='hello';
        - 查询字符串
          - 'key1=val1&key2=val2'
    - 布尔
      - Boolean
        - var d = true/false;

- null
  - var obj = {}; 空对象
- undefined
  - 变量声明但是未初始化
- (检测：typeof)
  - 可以检测除了null的其他数据类型

- 引用数据类型（保存在堆区）在栈区中保存的是指针
  - 数组
    - array
      - var arr = [ahs,ashgd,ashd]
  - 函数
    - function
      - var fun = function(){}
  - 对象
    - obejct
      - var obj = { name：‘tom'};
      - var obj = null;

- 值传递和引用传递
  - 栈区

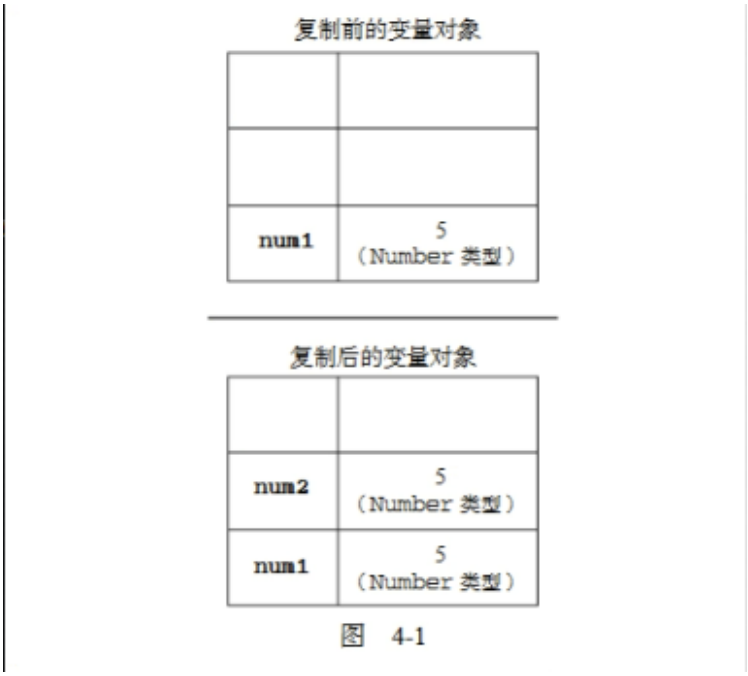


图 4-1

  - 堆区

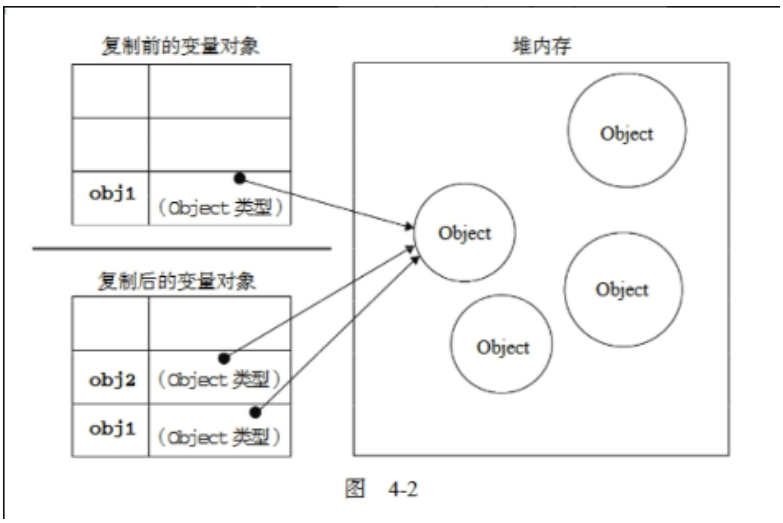


图 4-2

- 注释
- 流程控制语句
- 数组
  - 特点

```
存放多个值的集合【有序列表】
1）特点
    1．数组中每一项的数据类型没有任何限制，
    也就是可以存放任意的数据类型
    2．数组的大小可以动态的改变
    3．数组中存在索引的概念，索引从0开始
    4．以[]为边界，[]内的每一项通过,隔开
```

  - 创建数组
    - 数组字面量
      - var 数组名 = 【1，undefined，null，’a‘，function（），{}，【】】；
    - 构造函数Array
      - var 数组名 = new Array（）；
        -

```
var arr2 = new Array(3);
console.log(arr2); // [,,]
如果参数为单个数字类型，创建出来的数组则为相对应的空插槽
该数组与空数组[]不一样，该数组有长度，但是空数组[]长度为0
```

  - 数组的属性
    - length表示数组中元素的个数
  - 数组访问
    - 数组名【索引】
      - 索引从0开始
      - 如果索引超过了数组的长度，则返回undefined
      - 数组存在大小限制
      -

可以利用数组的长度和索引为数组新增项

```
var arr = ['a','b','c'];
arr[arr.length] = 'd';
arr[arr.length] = 'e';
console.log(arr); // ['a','b','c','d','e']
```

- 数组遍历
  - for

```
for
    var arr = ['a','b','c'];
    for(var i=0;i<arr.length;i++){
        console.log(i); // 0 1 2
        console.log(arr[i]); // a b c
```

  - while

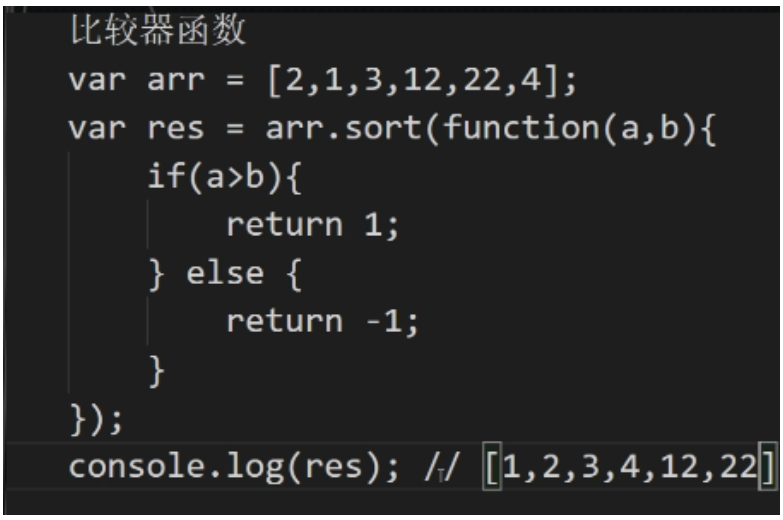
```
var arr = ['a','b','c'];
var i = 0;
while(i<arr.length){
    console.log(arr[i++]);
}
```

  - do-while
  - for-in

```
for-in
    var arr = ['a','b','c'];
    for(var key in arr){
        // console.log(typeof key);
        console.log(arr[key]);
    }
    arr为被遍历的数组，key是被遍历数组的索引【类型为string】，arr[key]是数组中的每一项
```

- 数组的检测
  - instanceof
    - 判断某一个变量是否是某个构造函数的实例
    - 语法：A instanceof B（a是否是b的实例）
    - 返回值：true/false
    - 结论：instanceof中数组、函数、对象都是一个对象
      - instanceof只能检测引用数据类型
      - instanceof检测数组、对象、函数时，返回值都是对象，也就是将数组、对象、函数都划分为对象
  - Array.isArray（）
    - 检测某个变量是否是数组

        - 返回值：true/false
        - 结论：专门检测数组
    - typeof
        - 检测某个变量的基本数据类型
        - typeof（A）
        - 结论：typeof检测基本数据类型的时候，除了null返回值为object之外，其他的都正确
        - typeof检测引用数据类型的时候，除了function的返回值正确之外，其他的都是object
    - Object.prototyoe.toString.call();
        - 检测所有的数据类型
- 数组api
    - push【改变原值】
        - 向数组的最后插入一个或多个元素
        - arr.push()
        - 返回值：新增完元素之后的数组的长度
    - pop
        - 把数组最后一个元素移出
        - arr.pop()
        - 返回值：被移出的元素
    - shift
        - 把数组的第一个元素移出
        - arr.shift()
        - 返回值：被移出的元素
    - unshift
        - 向数组的开头插入一个或多个元素
        - arr.unshift（）
        - 返回值：新增完元素之后的数组的长度
    - reverse（）
        - 反转数组
    - sort（）
        - 自定义排序



- 默认排序，var arr = 【3,2,1】//比较首个数字
- 序列化方法
  - toString（）
  - join（）
  - JSON.stringify（）
- 截取方法
  - concat（）
    - 拼接两个或者多个数组
  - slice（） （开始，结束）
    - 从数组中截取一段作为子数组
  - splice（）
    - 删除、替换、插入数组
      - 两个参数时是删除
        - splice（删除开始的位置，删除的个数）
      - 三个参数时插入
        - splice（开始位置，0，新增元素）
      - 三个参数替换
        - splice（起始位置，删除个数，替换元素）
- 迭代方法
  -
    - forEach没有返回值

- 函数
  - 构造函数和普通函数的区别
    - 构造函数首字母大写
    - 调用方式
      - 构造函数以new关键字调用
      - 普通函数直接调用

- 构造函数Object
- 作用
  - 利用函数封装某些具有特殊功能的代码，执行之后实现某种效果
  - 利用函数创建对象【构造函数】高级面向对象
- 函数的创建
  - 函数声明
    - function 函数名（形参）{ //函数体 }
  - 函数表达式
    - var 变量名 = function（形参）{ //函数体 ）
- 函数的调用
  - 函数名（实参列表）
  - 函数名.call (this,实参列表)
  - 函数名.apply (this,实参数组)
  - 函数名.bind()
- 函数使用括号和不使用括号的区别
  - 使用（）表示立即调用当前函数，执行函数体中的代码
  - 不使用（）表示当前函数的指针
- console.log（）与renturn的区别
  - console.log仅仅打印某个代码，而不会为函数提供返回值
  - return才是函数的返回值
- 提升
  - 代码从它们在代码中出现的位置被移动到当前作用域最上方进行执行，这个过程叫做提升
    - 变量提升
    - 函数声明提升
    - 函数声明与函数表达式提升
      - 函数声明的提升优先于变量的提升
- 引用数据类型的比较
- 函数内部属性
  - 只有在函数内部才能访问到的属性
    - 形参
      - 接收参数的快捷方式
      - 实参应该与形参的位置一一对应
    - arguments
      - 类数组对象
      - 当前函数所有实参真正存储的地方

- 分支主题
- callee指向当前的函数
- this
- 执行环境
- this的指向性问题：根据函数调用的方式判断
-
- 分支
- 如果使用（）调用
- 如果（）左边为函数名，再看函数名的左边是否有对象
- 如果没有，this指向全局对象
- nodejs -》global对象
- 浏览器 -》window对象
- 如果有
- 则this指向当前函数的对象
- 使用call或apply
- call和apply中，传递的是this需要指向的对象
- 分支主题
- this原本指向obj，但是通过call更改为指向obj2
- 箭头函数的this指向 外部函数 的this
- function test（）{} es5
- （）=>{} es6
- 闭包
- 函数内部的函数
- 匿名函数与自执行函数
- function（）{}
- （function（）{}）（）
- 对象
- 规则
- {}为边界
- {}内部保存的是一个个键值对，键值对由属性名和属性值组成
- 不同键值对之间以，隔开
- 属性名与属性值之间以：隔开
- 属性名一般不加‘’，除非包含特殊字符
- 属性值一般要使用‘’，但是数字类型可以不使用
- 创建
- 对象字面量

- 访问
- 删除
- 序列化和反序列化
  - 序列化：将对象转为json字符串
    - JSON.stringify（）；
  - 反序列化：JSON字符串转化为Object
    - JSON.parse（）；
- 遍历
  - for-in循环

- 正则表达式
  - 用法
    - 按照一定的规则对某个字符串进行验证
  - 作用
    - 手机、邮箱、表单验证
  - 定义方式
    - 正则表达式字面量
      - var pattern = /主体/修饰符
    - 构造函数RegExp
      - var pattern = new RegExp（'abc{3,5}','img'）；
  - 原型属性
    - global
    - ignoreCase
    - multiline
    - source
    - flags
    - lastIndex
      - 记忆exec下一次执行开始的位置
  - 原型方法
    - test（）；
      - true/false
    - exec();
      - 数组 /null
  - String对正则表达式的支持
  - 字符类
    - .
  - 数量词

- 包装器类型
  - 介绍
  - 执行过程
  - 原型方法
    - match（pattern）
    - legth
    - charAt(index)
    - indexOf
    - lastIndexOf（）
    - concat（）
    - 字符串截取
      - slice（）
      - substr()
      - substring()
    - 大小写
      - toLowerCase（）
      - toUpperCase（）
- Math
  - 比较
    - Math.min();
    - Math.max();
  - 舍入
    - Math.ceil(); 向上舍入
    - Math.floor(); 向下舍入
    - Math.round(); 四舍五入
  - 随机数
    - Math.random(); 0-1之间小数
- Date对象
  - 处理时间
    - 后台接口->时间->显示时间
      - 实例化日期对象
        - var date = new Date（）；
      - 字符串转换成日期
      - 方法
        - date.xxx
          - getFullYear();获取年份

- getMonth（）；获取月份范围：0-11
- getDate（）；获取日 1-31
- getDay（）；获取星期几 0-6
  - 0：星期天 6：星期六
- getHours（）；时
- getMinutes（）；分
- getSeconds（）；秒
- getTime（）；时间戳

- 使用
  - 原生js
    - 将获取到的年月日转换为字符串
    - 字符串中有padStart方法，可以补全位数

- 案例
  - 将时间转换为正常的格式
  - 第三方库
    - momentjs
      - 下载
        - script引入的方式
          - bootcdn
        - npm下载【包管理机制】
  - 使用
    - 官方文档
    - 博客

- 高级面向对象
  - 目的：利用函数封装一些具有特殊功能的代码（对象）
  - 封装【api的本质】
    - 工厂函数模式
      - 函数中创建对象，然后作为函数的返回值，之后，调用一次函数，即可创建一个对象
        - 缺点：虽然解决了创建多个相似对象的问题，但却没有解决对象识别问题（即再怎样知道一个对象的类型）
    - 构造函数（手机）
      - 内置构造函数（自带app）
      - 自定义构造函数（商城下载app）
        - 说明
          - 1.不需要在构造函数内创造对象

- 2.构造函数的首字母大写
- 3.没有return
- 4.属性与方法都赋值给了this
- 使用new关键字调用构造函数创建实例的时候，步骤如下
- 1.新建一个对象
- 2.将构造函数的作用域复制给该对象（也就是this指针指向了该对象）
- 执行构造函数内部的代码
- 为对象添加属性与方法
- 返回这个新建的对象
- 构造函数与普通函数的区别
- 构造函数与原型相结合的方式
- 将属性放在构造函数中，将方法放在构造函数的原型上
- 继承
- 原型链继承（方法）
- 子构造函数的原型指向父构造函数的实例
- 借用构造函数（属性）
- Person2.call（this，属性列表）
- ……