B/S体系软件设计

JavaScript

胡晓军

JavaScript

- **JavaScript**是一种轻型的需要解释执行的程序设计语言，而且具有面向对象的能力
- 核心与**C++**、**Java**相似
- 无类型语言
- 对象机制与**C++**、**Java**很不一样

对JavaScript的误解

- **JavaScript**并非**Java**
- **JavaScript**并不简单

JavaScript的版本

- JavaScript 1.2
- ECMAScript 6
- 客户端JavaScript
- 服务器端JavaScript
- 嵌入的JavaScript

浏览器中的JavaScript

- 控制文档的外观和内容使用
- 对浏览器的控制
- 与**HTML**表单的交互
- 与用户的交互
- 用**Cookie**读写用户的状态

在HTML中嵌入JavaScript

- <SCRIPT>…</SCRIPT>
- <SCRIPT SRC=“.js”></SCRIPT>
- <… onClick = “…”>

<SCRIPT>

- 文档装载并被**parse**到**<SCRIPT>**时执行
- **<Head>**中的**<SCRIPT>**一般用来定义在整个文档中会被执行的函数

包括JavaScript文件

- 可以简化**HTML**文件
- 可以共享**JS**
- 可以由浏览器实现缓存

- **<SCRIPT SRC="../util.js"></SCRIPT>**

事件处理器

```
<INPUT TYPE="checkbox" NAME="opts" VALUE="ignore-case"
   onClick = "ignore_case=this.checked;"
```

JavaScript的词法

- 大小写敏感
- 忽略空格和制表
- 分号或换行表示语句的结束
- 注释与Java一样

JavaScript基本数据类型

- 值类型
  - 数字
  - 字符串
  - 布尔量
  - **Null**
  - **Undefined**
- 引用数据类型
  - 对象
  - 数组
  - 函数

字符串

- 由单引号或双引号封装起来的字符序列
- \做转义序列
  - \b \f \n \t \’ \” \\
- 连接字符串 +

对象

- 命名了的数据段的集合
- 命名了的数据通常被作为对象的属性
  - **image.width**
  - **document.myform.button**
- 创建对象
  - **var now= new Date();**

数组

- 创建
  - **var a = new Array();**
- 直接量
  - **var a = [1.2, “string”,true]**

函数

- **function f(x) { return x*x; }**
  - **y = f(6); z = f(y);**
- 函数是一种基本数据类型

变量

- 声明
  - **var I,j,k;**
- 命名规则
  - 变量名的第一个字符必须是英文字母，或者是下划线符号**(underscore)_**
  - 变量名的第一个字母不能是数字。其后的字符，可以是英文字母，数字，和下划线符号符号**(underscore)_**
  - 变量名不能是**Javascript**的保留字
- 变量可以重复定义，也可以不定义就使用

运算符和优先级

Category	Operators
Primary	x.y f(x) a[x] new
Unary	+ - ++x --x
Multiplicative	* / %
Additive	+ -
Relational and type testing	< > <= >=
Equality	== === != !==
Conditional NOT	!
Conditional AND	&&
Conditional OR	||
Conditional	?:
Assignment	= *= /= %= += -=

语句

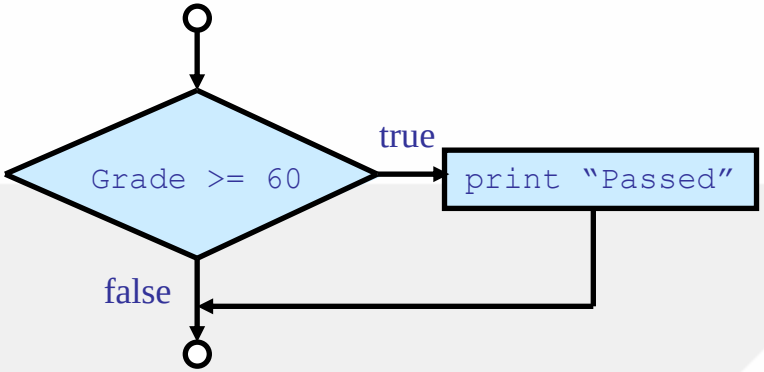
- 表达式
  - 常量
  - 变量
  - 函数
  - 用运算符组合以上内容
    - **value = Math.PI * radius * radius;**
    - 复制和相等
- 语句
  - 简单语句
  - 复合语句 **{}**
- 注释
  - **/* ... */**

程序控制流

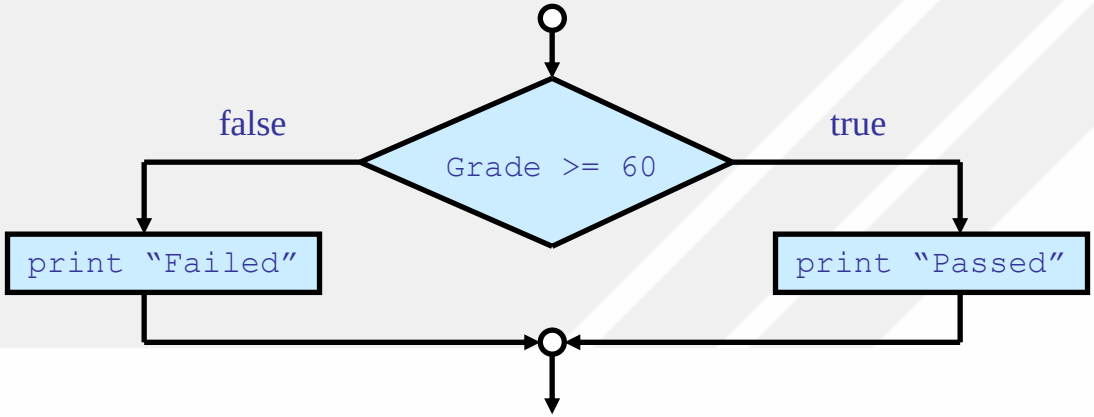
- 顺序语句
- 分支语句
  - **if .. else**
  - **switch**
- 循环语句
  - **for**
  - **while**
  - **do .. while**

if分支语句

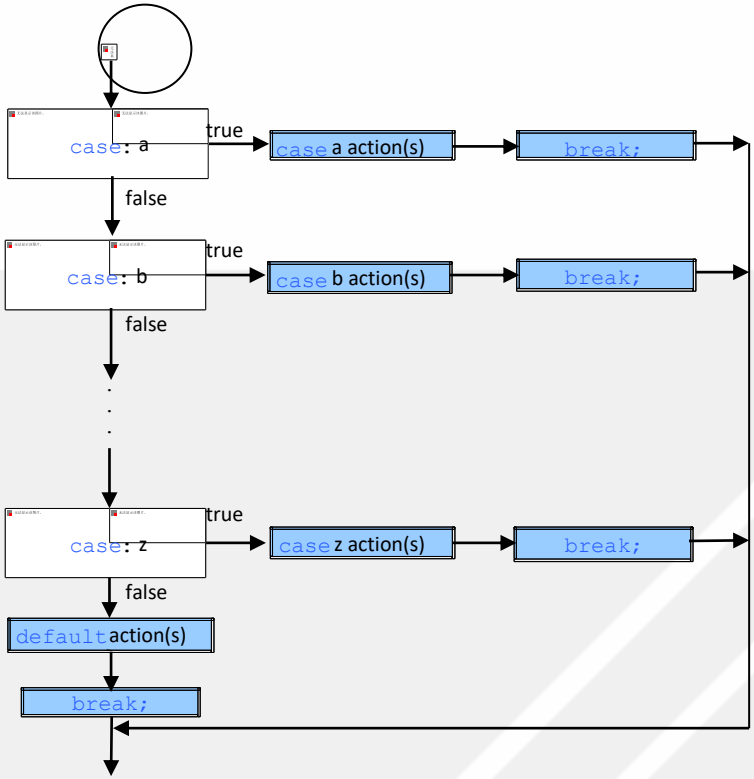
- if



- if .. else



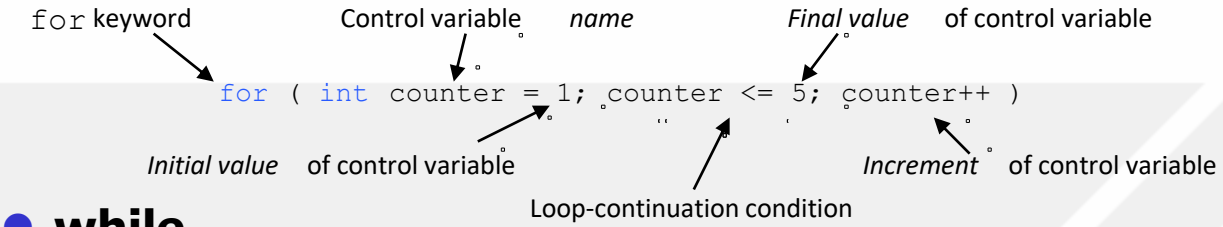
switch分支语句



- Switch on string

循环语句

- **for**



- **while**
  - **while (expression) statement**
  - 先判断再执行
- **do .. while**
  - **do statement while (expression)**
  - 先执行再判断
- **for (prop in object)**

常用对象

- Math
  - PI等
  - max,min,random,floor,round
  - sin,cos等
- String
  - length
  - charAt,substring,substr,indexOf,toUpperCase,toLowerCase
- Date
  - get/setYear(Month,Date,Hours,Minutes,Seconds)
- Array
  - []
  - push,pop,shift,unshift

Web浏览器环境

- window:当前窗口或框架(self)
  - document:当前文档
    - forms[]
      - Elements[]
    - anchors[]
    - links[]
    - images[]
  - location:当前URL
  - frames[]:当前frames

window

- status, defaultStatus
- alert(), confirm(), prompt()
- close()
- moveBy(), moveTo()
- open()
- resizeBy(),resizeTo()
- scrollBy(), scrollTo()
- setTimeout(“js”,timeInms)

DOM

- Document Object Model
- Document对象是DOM的根结点

文档对象的命名

- 每种类型是一个数组
- 命了名的**tag**直接是一个**document**对象的属性名

Document的属性和操作

- Anchors[]
- Applets[]
- fgColor
- bgColor
- Cookie
- Location
- Referrer
- Title
- URL

- write()

事件标签属性

- onblur
- onchange
- onclick
- ondblClick
- onfocus
- onkeydown
- onkeyup
- onkeypress
- onmousedown
- onmouseup
- onmouseout
- onmouseover
- onselect
- onsubmit
- onreset
- onload
- onunload

DOM事件模型

- DOM0级

```
<input type=button onclick="btnclick">
```

- DOM1级，一般meiyo

```
<script>
        document.getElementById("div1").onclick = btnclick
</script>
```

- DOM2级

```
<script>
        document.getElementById("div1").addEventListener(
                "click", btnclick)
</script>
```

实现一个网站注册网页

- 注册的基本内容
  - 用户名
  - **Email**
  - 密码及确认密码
- 利用**CSS**管理显示效果
- 用**JavaScript**判断输入内容的有效性
  - 用户名长度和字符有效性
  - **Email**格式验证
  - 密码长度和有效性验证
  - 两次密码输入匹配
- 提交到服务器
  - **CGI**
  - **ASP**
  - **php**
  - **JSP**

Javascript框架

- **prototype**
- **jQuery**：比较老的框架，组件丰富，浏览器兼容性较好
  - 轻量级 （**Lightweight**）
  - 强大的选择器
  - 出色的**DOM**操作封装
  - 可靠的事件处理机制
  - 出色的浏览器兼容性

目前主流的js前端框架

- **angular.js 2：google，ES 6语法，组件丰富，支持大型应用开发。IE 8以下版本不支持，比较复杂**
- **vue.js 2/3：尤雨溪，轻量化、高性能，view和model双向绑定。IE 8以下不支持，还在不断完善中**
- **react.js：facebook，virtual dom，JSX不是html**
  - **react native**

jQuery例子

```
<script src="jquery.min.js"></script>
<script>
 $(document).ready(function(){
   alert("Hello World!");
});
</script>
```

jQuery对象

- jQuery 对象就是通过 jQuery 包装 DOM 对象后产生的对象
- jQuery 对象是 jQuery 独有的. 如果一个对象是 jQuery 对象, 那么它就可以使用 jQuery 里的方法: $("#tab").html();
- jQuery 对象无法使用 DOM 对象的任何方法, 同样 DOM 对象也不能使用 jQuery 里的任何方法
- jQuery对象的方法和属性与DOM对象的并不相同，如
  - one.onclick()
  - $("#one").click()

jQuery对象和DOM对象的互转

- **jQuery 对象转DOM对象**
  - **jQuery 对象是一个数组对象, 可以通过 [index] 的方法得到对应的 DOM对象**
  - **$(“#msg”)[0]**
  - **使用 jQuery 中的 get(index) 方法得到相应的 DOM 对象**
  - **$("#msg").get(0)**
- **DOM对象转jQuery对象**
  - **用 $() 把 DOM 对象包装起来**
  - **$(document.getElementById(“msg”))**

```
$("#msg").html();
$("#msg")[0].innerHTML;
```

jQuery选择器

- 选择器是 **jQuery** 的根基, 在 **jQuery** 中, 对事件处理, 遍历 **DOM** 和 **Ajax** 操作都依赖于选择器
- **jQuery** 选择器的优点:
  - 简洁的写法
  - 完善的事件处理机制

```
$("#id")          //document.getElementById("id");
$("tagName")      //document.getElementsByTagName("tagName");
```

```
//若在网页中没有 id 为 "id" 的元素, 浏览器会报错
//document.getElementById("id").style.color = "red";

//需要先判断 document.getElementById("id") 是否存在
if(document.getElementById("id"))
    document.getElementById("id").style.color = "red";

//使用 jQuery 获取网页中的元素即使不存在页不会报错
$("#id").css("color", "red");
```

jQuery选择器

- 基本选择器
  - #id, element, .class
  - selector1, selector2
- 层次选择器
  - ancestor descendant
  - parent>child
  - prev + next
  - prev ~ siblings
- 过滤选择器
  - :first/last
  - :even/odd
  - :eq/gt/lt(index)
  - :hidden/visible
  - :enabled/disabled/checked/selected
  - [attribute=value]

jQuery事件

```
$(document).ready(function(){
        $("#id").click(function(){
                alert("clicked");
        });
});
```

jQuery扩展

- jQuery UI
- jQuery Mobile

JavaScript深入：对象

- 对象字面量(Literal)

```
var foo = {
        bar: 'test'
};
```

  - 与**JSON**(JavaScript Object Notation)的区别

- **new Object**

```
function Student(name) {
        this.name = name;
}
StudentA = new Student('A');
StudentB = new Student('B');
```

- **Object.create**
  - **ECMAScript 5**
  - **Object.create(Object.prototype, {})**

对象属性

- 动态
- 可配置
- 可枚举
- 可删除
- 可写性

```
var car = {}
Object.defineProperty(car, 'door', {
        writable : true,
        configurable: true,
        enumerable: true,
        value: 4
});
```

函数深入：闭包

- 变量作用域

```
function f1(){
        var n=999;
        return function (){
                alert(n); // 999
        }
}
```

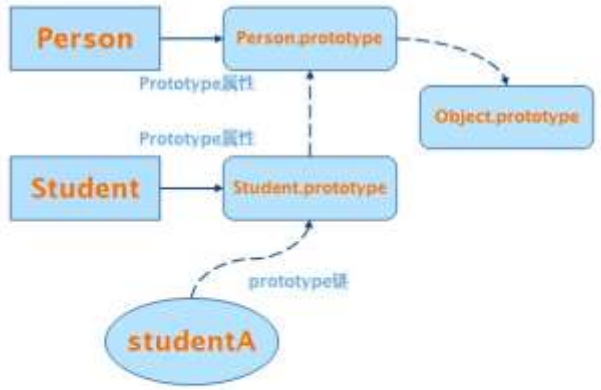
- 闭包简单理解成"定义在一个函数内部的函数”
  - 可以读取函数内部的变量，
  - 让这些变量的值始终保持在内存中

prototype

- **prototype**：对象类型原型的引用
- **__proto__**：可以理解为构造器的原型
- 每个**Function**对象都有一个 **prototype** 属性，指向另外一个对象，每个对象都有**__proto__**
- 使用对象字面量创建的对象的 **__proto__** 指向的是 **Object.prototype**
- **var obj = {"name" : "Alex"}**
- **Object.prototype**
- 用 **new** 操作符创建的对象的 **__proto__** 指向的是其构造器的 **prototype**
- **var users = new Array();**
- **Array.prototype**
- **Object.prototype**的**__proto__**是**null**

prototype链

- **prototype** 链在属性查找过程中会起作用
- 在对象中查找某个特定名称的属性
  - 首先检查该对象本身。如果找到的话，就返回该属性的值；
  - 如果找不到的话，会检查该对象的 **prototype** 指向的对象。
- 设置对象中某个属性的值
  - 如果当前对象中存在这个属性，则更新其值；
  - 否则在当前对象中创建该属性。




```
function Person() {}
function Student() {}
Student.prototype = new Person();
var studentA = new Student();
```

关于继承

- **prototype链实现prototype的继承**
- **与Java/C++的类的继承不一样**
  - **Java：**
    - 对象状态在对象实例中，方法在类中
  - **JavaScript：**
    - 状态和方法都在对象实例中
    - **Prototype的属性被这个原型的所有实例对象共享**

this

- 调用位置
- 绑定规则
  - 函数是否在 **new** 中调用（**new** 绑定）？如果是的话 **this** 绑定的是新创建的对象
    - **var bar = new foo()**
  - 函数是否通过 **call**、**apply**（显式绑定）或者硬绑定调用（**bind**）？如果是的话，**this** 绑定的是指定的对象
    - **var bar = foo.call()**
  - 函数是否在某个上下文对象中调用（隐式绑定）？如果是的话，**this** 绑定的是那个上下文对象
    - **var bar = obj1.foo()**
  - 如果都不是的话，使用默认绑定。如果在严格模式下，就绑定到 **undefined**，否则绑定到全局对象
    - **var bar = foo()**
- 箭头函数
  - 保存当前的作用域链，然后顺着当前的作用域链寻找 this，并且只会在作用域链最前端的活动对象或变量对象中寻找

函数深入：立即调用函数表达式

- **IIFE(Immediately Invoked Function Expression)**

```
(function(){
    var a = 100;
    alert('bbb' + a);
})();
```

- 优点
  - 闭包防止命名冲突
  - 块级作用域
  - 防止污染全局命名空间
  - 代码模块化

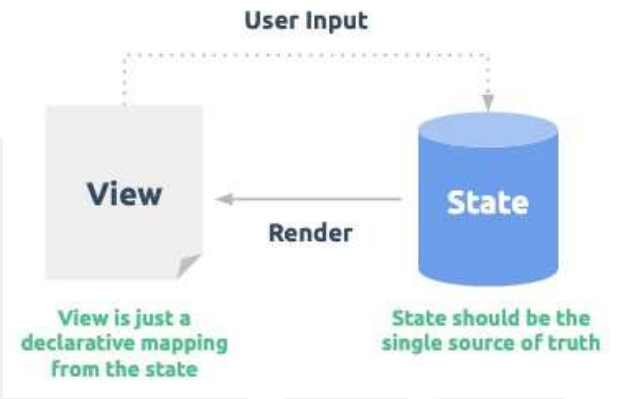
Vue.js

- **MVVM： Model+View+ViewModel**
  - **MVVM是把MVC的Controller里的Presenter改成了ViewModel。**
  - **View的变化会自动更新到ViewModel，ViewModel的变化也会自动同步到View上显示**
  - **这种自动同步是因为ViewModel中的属性实现了Observer，当属性变更时都能触发对应的操作。**
- **Vue.js是一个轻巧、高性能、可组件化的MVVM库，同时拥有非常容易上手的API**
- **Vue.js是一个构建数据驱动的Web界面的库**

Vue.js

- Vue.js的特性如下：
  - 轻量级的框架
  - 双向数据绑定
  - 指令
  - 插件化
- **Component**
- **Client Side Routing**

```
var App = Vue.extend({})
var router = new VueRouter()
router.map({
    '/movies': {
        component: Movies
    },
    '/settings': {
        component: Settings
    }
})
router.start(App, '#app')
```



UI框架

- **Element UI**
  - 一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的桌面端组件库
    - 布局
    - 大量常用组件
    - 自定义组件
  - **Vue element admin**
    - **后台管理应用的集成方案**
    - **二次开发**
    - **后端交互/Mock**
    - **权限与安全**
    - **Webpack**
- **Element Plus**
  - **Vue 3.0**

JavaScript压缩

- 减少字节数
- 代码混淆
- 工具：**UglifyJS**、**Google Closure Complier**

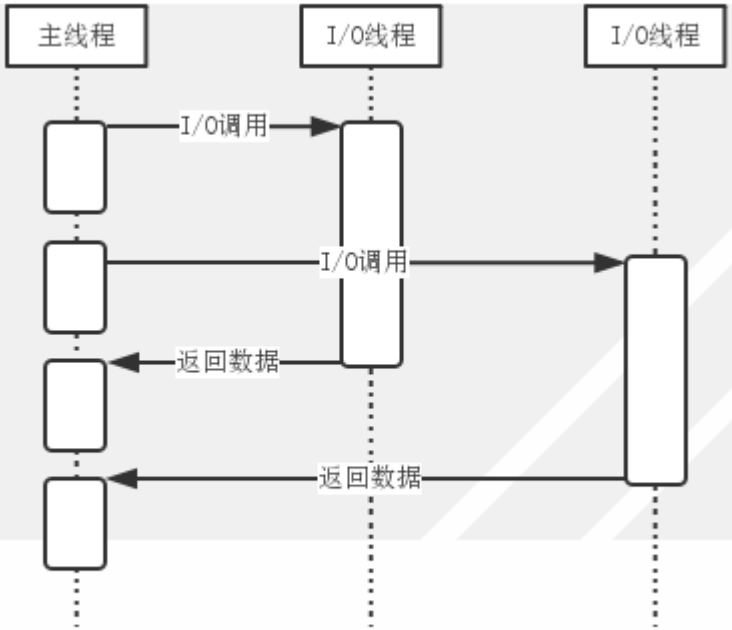
- **Webpack**
  - 打包：将多个文件打包成一个文件，减少服务器压力和下载带宽
  - 转换：将预编译语言转换成浏览器识别的语言
  - 优化：性能优化

Node.js简介

- Node.js是一个基于Chrome JavaScript运行时建立的一个平台，用来方便地搭建快速的易于扩展的网络应用
- Node.js 借助事件驱动，非阻塞I/O模型变得轻量和高效， 非常适合运行在分布式设备 的 数据密集型 的实时应用
- V8引擎执行Javascript的速度非常快，性能非常好。Node对一些特殊用例进行了优化，提供了替代的API，使得V8在非浏览器环境下运行得更好
- Node.js是服务器端的JavaScript运行环境，它具有无阻塞(non-blocking)和事件驱动(event-driven)等的特色，Node.js实现了web服务器，让你可以很方便的通过它来搭建基于JavaScript的Web App

Node.js的特点

- 单线程
- Node. js可以在不新增额外线程的情况下，依然可以对任务进行并行处理



线程池模拟异步I/O

单线程异步IO模型

- 无锁,逻辑简单
- 无多线程、多进程的开销, 高性能
- 不用等待耗时的I/O
- 异步编程很痛苦
- 单线程未捕获异常导致进程挂掉
- CPU密集型操作堵塞整个线程

- JavaScript Module
  - Node CommonJS: require
  - ES6: import/export
- package.json
- npm(node package manager)

Node.js web开发

- Express
  - route
  - database
- Socket.io