**（1）.与[ ]的区别**

中括号运算符总是能代替点运算符，但点运算符却不一定能全部代替中括号运算符。

1. 中括号运算符可以用字符串变量的内容作为属性名。点运算符不能。
2. 中括号运算符可以用纯数字为属性名。点运算符不能。
3. 中括号运算符可以用js的关键字和保留字作为属性名。点运算符不能。

**（2）事件委托的原理及优点**

原理：利用冒泡的原理，把事件加到父级上，检查事件对象的target属性，捕获真正被点击的节点元素的引用，触发执行效果。

好处：1）避免用for循环，提高性能，2）可以同时为新增元素添加事件

**（3）跨域通信**

同源：协议相同，域名相同，端口相同

跨域通信：js进行DOM操作、通信时（对静态资源没有限制，例如css文件、图片等）如果目标与当前窗口不满足同源条件，浏览器为了安全会阻止跨域操作。

方法（均是利用对静态资源没有限制这一点）：

1. 使用<script>进行jsonp请求（Ajax中$.getJSON方法会自动判断是否跨域；$.ajax中设置dataType：jsonp即可）
2. 使用HTML5的postMessage：

请求窗口：targetWindow.postMessage(data, origin);其中data是需要发送的对象，origin是目标窗口的origin

接受窗口：window.addEventListener('message', handler, false);handler的event.data是postMessage发送来的数据，event.origin是发送消息窗口的origin，event.source是发送消息的窗口引用

1. 现代浏览器可使用HTML5规范的CORS功能，只要目标服务器头部添加header("Access-Comtrol-Allow-Origin:\*");即可像普通ajax一样访问跨域资源（IE10以下不支持）

**（5）闭包的优缺点**

概念：指有权访问另一个函数作用域中变量的函数。

优点：1）保护函数内的变量安全，加强封装性；2）在内存中维持一个变量

缺点：另一个函数的活动对象无法销毁，常驻内存，可能导致内存泄漏

**（6）typeof操作符（不是函数）能检测哪几种类型**

ECMAScript有5重基本数据类型：Undefined，Boolean，String，Number，Null，1种复杂数据类型：Object。typeof能检测一个变量是不是基本数据类型

“undefined”：未定义

“boolean”：布尔值

“string”：字符串

“number”：数值

“object”：对象或null（空对象）

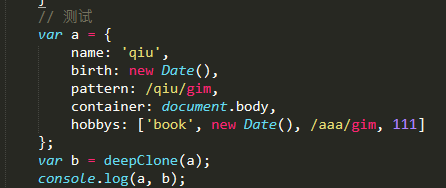
“function”：函数

**（7）相等a==b**

1. 如果a与b类型相同，则按===规则来比较
2. 类型不同，则按如下规则做转换
3. a为**数字**，b为**字符串**，将b转换为数字再比较
4. 若有**布尔类型**，true->1，false->0
5. 若**a为对象，b不是对象**，则调用对象的valueOf()方法，用得到的基本类型按前面的规则比较
6. Null 和undefined不做转换，Null==undefined；NAN不等于任何值，包括自己；

**（8）深拷贝**





**（9）回流reflow与重绘repaint**

重绘：只影响样式，不影响布局；outline, visibility, background color

回流：影响样式，会导致回流的操作有：

1. 增删DOM元素；
2. 窗口大小改变---resize
3. 改变字体大小、元素大小、位置改变

减少回流的方法：

1. 将多次样式改变写在一起，中间不要掺杂读操作（会隔断写操作）；
2. 避免一条一条的改变样式，尽量使用添加class方法或使用csstext属性，一次性改变属性；（**el.style.cssText += "; left: " + left + "px; top: " + top + "px;";）**
3. 尽量使用离线DOM操作，比如操作Document Fragment对象，完成后再加这个对象家DOM；再比如使用 cloneNode() 方法，在克隆节点上进行操作。
4. 动画中的元素尽量使用position的fixed和absolute；
5. 先设为display：none，所有操作完毕后再恢复显示；

**（10）前端性能优化方法**

1. 减少http请求次数：合并文件、CSS精灵；
2. 将css和js放在外部文件中，且css放在头（浏览器会先下载css再渲染，若css不在头，会导致从无样式到有样式的突变，用户体验不好），js放在尾（遇到js页面会停止渲染）；
3. 控制DOM数量，减少对DOM的操作次数：可将其赋给局部变量；
4. 压缩js和css文件；
5. 减少回流次数

**（11）前端要注意哪些SEO**

1. 语义化的HTML代码，似的搜索引擎更容易理解网页
2. 重要内容不要用js输出：爬虫不会执行js操作
3. 非装饰性图片带上alt
4. 合理的title、description、keywords：三者权重逐渐降低
5. 重要内容HTML放在最前面：搜索引擎抓HTML是从上至下，有的搜素引擎对长度有限制，放在前面保证重要内容能被抓取到
6. 少iframe（内联框架）：搜索引擎不会抓取iframe中的内容

**（12）什么是渐进增强**

保证所有人都能访问基本内容和功能，同时为高级浏览器提供更好的用户体验。

核心原则：

1. 所有浏览器都能访问所有基本内容和使用基本功能，
2. 通过外部css提供增强的布局
3. 通过外部js提供增加的功能

**（13）call和apply的作用和区别**

用途：在特定的作用域中调用函数（这样对象不需要与方法有任何耦合关系）

区别：call——参数必须逐个列举出来，apply——参数数组（实例或arguments对象）

由于apply方法可以把参数数组转换为参数列表，故其有一些巧妙的用法：

1. Math.max.apply(null,array)可以找出array里的最大项；
2. Array.prototype.push.apply(arr1,arr2)可以将arr2合并到arr1中；

（即当目标函数要接受n个参数列表时，而不接受数组形式时，可以用apply巧妙的解决）

### （14）函数内部arguments变量有哪些特性,有哪些属性,如何将它转换为数组

定义：arguments是所有函数都包含的一个局部变量，对应函数调用时的实参，是一个类数组；

属性：1）指向对象下标元素的索引arguments[index]；

2）arguments.length代表实参个数（functionName.length/arguments.callee.length代表形参个数）；

3）arguments.callee为当前正在执行的函数本身转换为数组：Array.prototype.slice.call(arguments)

**（15）callee和caller**

callee: callee是arguments 的一个属性成员，它表示对函数对象本身的引用;

caller: functionName.caller（arguments.callee.caller更好） 返回函数functionName的调用者，在全局作用域中调用functionName时返回null。

**（16）定义函数的几种方法**

1. 函数声明：function sum(a,b){return a+b}；
2. 函数表达式 ：var sum = function(a,b){return a+b}；
3. 使用Function构造函数：var sum = new Function(){“a”,”b”,”return a+b”}（不推荐，会解析两次代码，一次js，一次是传入的字符串）

（最后一个参数被认为识函数体）

1. es6箭头函数

函数声明与函数表达式的唯一区别：解析器会率先读取函数声明，而函数表达式必须等到解析器执行到它所在的代码行，在被真正解释执行。所以，采用函数表达式时，调用不能在定义之前。

**（17）前后端分离**

前端：负责VC部分，与客户端交互相关的部分；

后端：负责M部分，负责数据处理

联系：通过接口连接接口端

意义：1）分工明确，提高效率：两者开发可以同时进行，在后端还没提供接口时，可以利用本地文件对效果进行测试。

1. 降低维护成本：可以快速定位问题所在，增加代码可维护性和重构性。

**（18）clientX，offsetX，screenX**

screenX:鼠标在显示屏幕上的坐标。   
 clientX:鼠标在浏览器可视窗口的坐标。

OffsetX：鼠标相对于“触发事件的元素”的位置（IE特有）

**（19）offsetWidth/offsetHeight,clientWidth/clientHeight与scrollWidth/scrollHeight的区别**

* offsetWidth/offsetHeight返回值包含content + padding + border，效果与e.getBoundingClientRect()相同
* clientWidth/clientHeight返回值只包含content + padding，如果有滚动条，也不包含滚动条
* scrollWidth/scrollHeight返回值包含content + padding + 溢出内容的尺寸

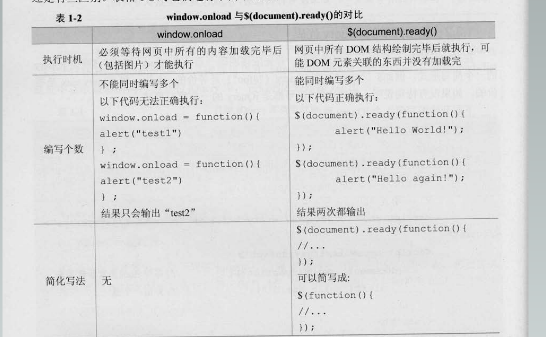
**（4）哪些事件冒泡**

每个 event 都有一个event.bubbles属性，可以知道它可否冒泡。

|  |  |
| --- | --- |
| **Event Type** | **Bubbling phase** |
| abort | ✗ |
| beforeinput | ✔ |
| blur | ✗ |
| click | ✔ |
| compositionstart | ✔ |
| compositionupdate | ✔ |
| compositionend | ✔ |
| dblclick | ✔ |
| error | ✗ |
| focus | ✗ |
| focusin | ✔ |
| focusout | ✔ |
| input | ✔ |
| keydown | ✔ |
| keyup | ✔ |
| load | ✗ |
| mousedown | ✔ |
| mouseenter | ✗ |
| mouseleave | ✗ |
| mousemove | ✔ |
| mouseout | ✔ |
| mouseover | ✔ |
| mouseup | ✔ |
| resize | ✗ |
| scroll | ✔ |
| select | ✔ |

**（20）$(document).ready(function(){})与window.onLoad=function(){}的区别**

**window.onLoad=function(){}相当于$(window).load(function(){})**



**（21）bind，on，delegate，live的区别**

bind：$("a").bind("click",function(){alert("ok");});，直接绑定在元素上，所以不能为新元素添加事件；

live：$("a").live("click",function(){alert("ok");});，通过冒泡的方式来绑定到元素上，可以为新元素添加事件；

delegate：$("#container").delegate("a","click",function(){alert("ok");})，更精确的小范围使用事件代理，性能优于.live()；

on：on(events,[selector],[data],fn)，所有delegate(),live(),bind()方法内部都是调用的on方法。

**（22）BOM**

location，navigator，screen，history均为window对象的属性；

location：提供与当前窗口中加载的文档有关的信息和url信息，window.location=document.location；

navigator：包含有关浏览器的信息，navigator.userAgent可用来判断浏览器版本信息；

screen：包含浏览器窗口的显示器信息

history：保存着用户的上网记录，但开发人员无法得知其URL，可以通过go，back，forWord进行跳转浏览；

（23）if判断

Undefined，null，0/NAN，false，“”分别会判断为false，{}会判断为true

|  |  |  |
| --- | --- | --- |
| 数据类型 | 转换为true | 转换为false |
| boolean | true | false |
| String | 任何非空字符串 | 空字符串 |
| Number | 任何非零数字值 | 0和NaN |
| Object | 任何对象 | null |
| Undefined | -- | undefined |

**（24）js异步编程**

区别标准：程序的执行顺序与任务的排列顺序是（同步）否（异步）一致；

同步模式：程序的执行顺序与任务的排列顺序是一致的、同步的；

异步模式：每一个任务有一个或多个回调函数（callback），前一个任务结束后，不是执行后一个任务，而是执行回调函数，后一个任务则是不等前一个任务结束就执行。耗时较长的操作都应该异步模式。

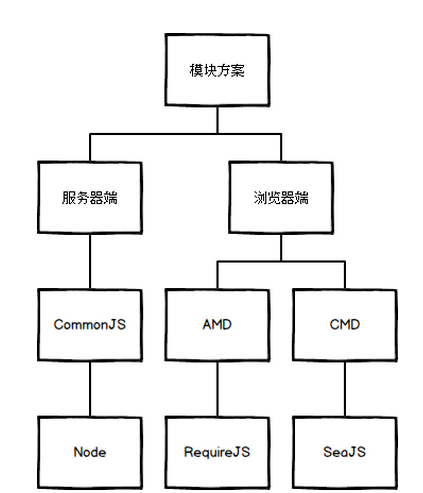
|  |  |  |
| --- | --- | --- |
| 方式 | 优点 | 缺点 |
| 回调函数 | 简单、容易理解和部署 | 代码高度耦合，不利于代码阅读和维护 |
| 事件监听 |  |  |
| 发布/订阅 | f1: jQuery.publish("done"); f2: jQuery.subscribe("done", f2); | |
| Promises对象 |  |  |

**（25）js模块化是什么？有什么好处？**  
定义：将一个系统划分为多个模块，一个模块为实现特定功能的文件。

优点：1）提高效率和可维护性：多人协作互不干扰，提高工作效率；发现问题可对单个模块进行测试

2）避免命名冲突：模块内的命名与其他模块无关

3）更好的依赖处理：在模块内部申明依赖即可，方便增删



模块化的三个规范：

1.**commonJS**：通用的模块规范（同步）；

2.**AMD**：异步模块定义规范（预加载，依赖前置）；

3.**CMD**：通用模块定义规范（延迟加载，依赖就近）。

AMD：使用define定义模块，使用require调用模块；define(id?, dependencies?, factory);

**（26）js严格模式（use strict）**

优点：1）消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

2）消除代码运行的一些不安全之处，保证代码运行的安全；

3）提高编译器效率，增加运行速度；

4）为未来新版本的Javascript做好铺垫。

如何调用：1）放在脚本文件的第一行；2）放在函数体的第一行

区别：1）**变量必须先声明，直接给变量赋值，不会隐式创建全局变量，不能用with；**

**2）禁止this关键字指向window：如call apply传入null undefined保持原样不被转换为window；**

**3）限制对调用栈的检测能力，访问arguments，callee，caller会抛出异常**

**4）重名错误：对象不能有同名属性；函数不能有同名形参；**

**5）禁止八进制表示法：如0100会报错，而不会解释为64。**

**（27）document.write和innerHTML**

1.write是DOM方法,向文档写入HTML表达式或JavaScript代码，可列出多个参数，参数被顺序添加到文档中 ；innerHTML是DOM属性,设置或返回调用元素开始结束标签之间的HTML元素。

2.两者都可向页面输出内容,innerHTML比document.write更灵活。

当文档加载时调用document.write直接向页面输出内容，文档加载结束后调用document.write输出内容会重写整个页面。通常按照两种的方式使用 write() 方法：一是在使用该方在文档中输出 HTML，二是在调用该方法的的窗口之外的窗口、框架中产生新文档（务必使用close关闭文档）。

在读模式下，innerHTML属性返回与调用元素的所有子节点对应的HTML标记，在写模式下，innerHTML会根据指定的值创建新的DOM树替换调用元素原先的所有子节点。

3.两者都可动态包含外部资源如JavaScript文件

通过document.write插入<script>元素会自动执行其中的脚本；

大多数浏览器中，通过innerHTML插入<script>元素并不会执行其中的脚本。

**（28）xml和json**

**数据体积：json相对较小，传输更快；**

**解码难度：xml父子节点嵌套复杂，解码难，json的解码难度几乎为0；**

**支持度：json语言，包括很多服务器语言，便于服务器解析，较少维护和开发成本**

**描述性：xml略优于json**

**（29）****attribute和property**

**attribute是dom元素在文档中作为html标签拥有的属性；**

**property就是dom元素在js中作为对象拥有的属性。**

**所以：**

**对于html的标准属性来说，attribute和property是同步的，是会自动更新的，**

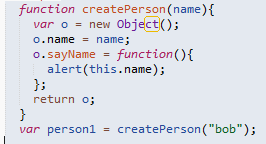
**但是对于自定义的属性来说，他们是不同步的，**

（30）isprototype，getPrototypeOf，hasOwnPrototype

|  |  |
| --- | --- |
| isprototype | 确定对象之间是否存在关系，Person.prototype.isPrototypeOf(person1)//true |
| getPrototypeOf | 去的一个对象的原型，Object.getPrototypeOf(person1) == Person.prototype;//true |
| hasOwnPrototype | 检测一个属性是存在与实例中，person1.hasOwnPrototype("name");//若不是来自原型则返回true |

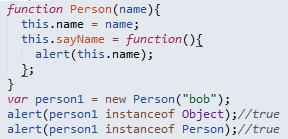
**（31）js对象创建的几种方式**

1）工厂模式

优点：解决了创建多个相似对象的问题；

缺点：无法进行对象识别

（2）构造函数模式

优点：可以将它的实例标识为一种特定的类型；

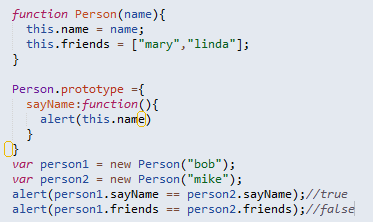
缺点：每个方法都要在实例上重新创建一遍

（3）原型模式

优点：原型中所有属性是被很多实例共享，这一点对于函数十分适合；

缺点：但对于引用类型值，会导致不同实例的属性拥有相同的引用。

（4）组合构造函数模式和原型模式

特点：构造函数模式用于定义实例属性，原型模式用于定义方法和共享的属性。

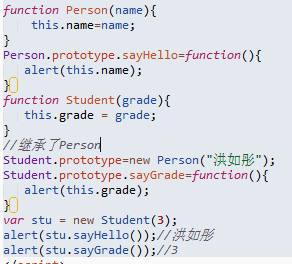
5，动态原型模式

6，寄生构造函数模式

7，稳妥构造函数模式

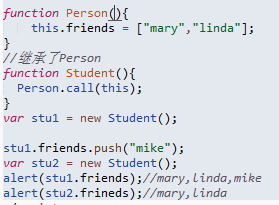
**（32）继承**

1）原型链继承

特点：采用组合式方式定义了Person对象，通过Student.prototype=new Person("洪如彤"); 实现了继承；

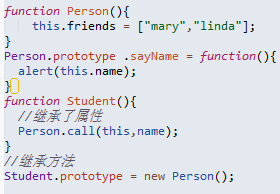
缺点：原先在构造函数Person中的实例属性，通过继承已经变成了Student的原型属性

（2）借用构造函数

特点：使用call/apply，在Student的实例的环境下调用Person，这样就会在Student实例上执行Person中的对象初始化代码，于是每个Student实例就会有自己的friends副本了。

缺点：方法定义在构造函数中，无法复用

（3）组合继承

特点：使用原型链实现对原型属性和方法的继承，借用构造函数实现对实例属性的继承。

4，原型式继承

5，寄生式继承6，寄生组合式继承

**（33）创建ajax的过程**

1. 创建XMLHttpRequest对象，
2. 创建一个新的http请求，并指定其方法、url和是否异步发送请求的布尔值；
3. 设置响应http请求的状态改变函数
4. 发送http请求
5. 获取异步请求返回的数据
6. 利用js进行局部刷新

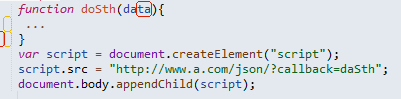
|  |  |
| --- | --- |
| readyState状态 |  |
| 0 | 未初始化：尚未调用open方法 |
| 1 | 启动：已经调用open方法，尚未调用send方法 |
| 2 | 发送：已经调用send方法，尚未接受到响应 |
| 3 | 接受：已经接受到部分响应数据 |
| 4 | 完成：已经接受到全部响应数据，而且已经可以在客户端使用 |

缺点：1）url不变，back键失去作用（如何解决？）；

2）手机等不能很好的支持ajax

【jsonp】（未实践）

**原理**：不使用XHR，采用动态script。在请求地址中指定回调函数



**优点**：简单易用，支持双向通信，可直接访问响应文本

**缺点**：1）jsonp是从其他域中加载代码执行，不能保证其他域是否安全；2）要确定jsonp请求失败不容易（可使用计时器检测指定时间内是否收到了响应）

【CORS】

**原理**：使用自定义的http头部让浏览器和服务器沟通，从而决定请求或响应是否应成功

A页面发送请求时，需要给其附加一个Origin的头部，包含A的地址信息（Origin：http://www.a.com）；B服务器在头部回发相同的地址信息（Access-Control-Allow-Origin:http://www.a.com），如果是公共资源，可以回发\*。若头部缺失或不匹配，浏览器都将驳回请求。

**优点**：由于使用的是XHR，所以错误处理方式比jsonp好；支持get，post等方法

【图片ping】