# 公司面试题集锦

目录

**[1. SDC](#_Toc11982_WPSOffice_Level1)** **[4](#_Toc11982_WPSOffice_Level1)**

[1.1 HTTP状态码知道哪些？](#_Toc22867_WPSOffice_Level2) [4](#_Toc22867_WPSOffice_Level2)

[1.2  块级元素、行内元素和空元素（void）](#_Toc16912_WPSOffice_Level2) [4](#_Toc16912_WPSOffice_Level2)

[1.3 HTML5文档类型和字符集](#_Toc11078_WPSOffice_Level2) [5](#_Toc11078_WPSOffice_Level2)

[1.4 请写出jQuery绑定事件的方法，不少于两种](#_Toc15693_WPSOffice_Level2) [5](#_Toc15693_WPSOffice_Level2)

[1.5 jsp有哪些内置的对象(至少写出5个)， 注： 写出其常用方法](#_Toc13936_WPSOffice_Level2) [5](#_Toc13936_WPSOffice_Level2)

[1.6 js有哪些内置对象？](#_Toc13522_WPSOffice_Level2) [5](#_Toc13522_WPSOffice_Level2)

[1.7 优化页面加载速度的方法](#_Toc22011_WPSOffice_Level2) [6](#_Toc22011_WPSOffice_Level2)

**[2. 凡普金科](#_Toc22867_WPSOffice_Level1)** **[7](#_Toc22867_WPSOffice_Level1)**

[2.1 DOM事件流包含那几个阶段？](#_Toc20609_WPSOffice_Level2) [7](#_Toc20609_WPSOffice_Level2)

[2.2 事件委托的原理，优点和缺点](#_Toc28190_WPSOffice_Level2) [7](#_Toc28190_WPSOffice_Level2)

[2.3 请写出jQuery绑定事件的方法，不少于两种](#_Toc14377_WPSOffice_Level2) [7](#_Toc14377_WPSOffice_Level2)

[2.4 什么是原型链，什么是作用域链，有什么作用？](#_Toc21015_WPSOffice_Level2) [8](#_Toc21015_WPSOffice_Level2)

[2.5 请使用原生JavaScript为一下li实现事件委托，点击后打印其对应的node-type属性值](#_Toc14864_WPSOffice_Level2) [8](#_Toc14864_WPSOffice_Level2)

[2.6 请说出一下代码输出的内容，并解释为什么？](#_Toc18185_WPSOffice_Level2) [8](#_Toc18185_WPSOffice_Level2)

[2.7 请说出一下代码输出的内容，并解释为什么？](#_Toc24289_WPSOffice_Level2) [9](#_Toc24289_WPSOffice_Level2)

[2.8 函数节流和函数防抖](#_Toc29377_WPSOffice_Level2) [9](#_Toc29377_WPSOffice_Level2)

[2.9 已知构造函数A，请定义构造函数B，要求B继承A](#_Toc7637_WPSOffice_Level2) [10](#_Toc7637_WPSOffice_Level2)

[表 1题目](#_Toc15792_WPSOffice_Level2) [11](#_Toc15792_WPSOffice_Level2)

[表 2答案](#_Toc15181_WPSOffice_Level2) [11](#_Toc15181_WPSOffice_Level2)

**[3. 铂金智慧](#_Toc16912_WPSOffice_Level1)** **[11](#_Toc16912_WPSOffice_Level1)**

[3.1 如何垂直居中div(尽可能多的方法)](#_Toc10648_WPSOffice_Level2) [11](#_Toc10648_WPSOffice_Level2)

[3.2 em和rem的相同点和不同点](#_Toc27974_WPSOffice_Level2) [13](#_Toc27974_WPSOffice_Level2)

[3.3 跨域问题和解决方案](#_Toc27325_WPSOffice_Level2) [13](#_Toc27325_WPSOffice_Level2)

**[方法1:](#_Toc11078_WPSOffice_Level1)** **[31](#_Toc11078_WPSOffice_Level1)**

**[方法2:](#_Toc15693_WPSOffice_Level1)** **[33](#_Toc15693_WPSOffice_Level1)**

**[1.vue-router刷新的几种方式](#_Toc13936_WPSOffice_Level1)** **[33](#_Toc13936_WPSOffice_Level1)**

**[2.说出至少4中vue当中的指令和他的用法?](#_Toc13522_WPSOffice_Level1)** **[35](#_Toc13522_WPSOffice_Level1)**

[1、v-text](#_Toc31812_WPSOffice_Level2) [35](#_Toc31812_WPSOffice_Level2)

[2、v-html](#_Toc16289_WPSOffice_Level2) [36](#_Toc16289_WPSOffice_Level2)

[3、v-model](#_Toc3231_WPSOffice_Level2) [36](#_Toc3231_WPSOffice_Level2)

[4.v-if](#_Toc9105_WPSOffice_Level2) [37](#_Toc9105_WPSOffice_Level2)

[5.v-on](#_Toc23401_WPSOffice_Level2) [38](#_Toc23401_WPSOffice_Level2)

**[3. vuex有哪几种属性?说出具体使用场景](#_Toc22011_WPSOffice_Level1)** **[39](#_Toc22011_WPSOffice_Level1)**

[1. state](#_Toc6788_WPSOffice_Level2) [39](#_Toc6788_WPSOffice_Level2)

[2. getters](#_Toc21309_WPSOffice_Level2) [39](#_Toc21309_WPSOffice_Level2)

[3. mutations](#_Toc2340_WPSOffice_Level2) [39](#_Toc2340_WPSOffice_Level2)

[4. actions](#_Toc16559_WPSOffice_Level2) [39](#_Toc16559_WPSOffice_Level2)

[5. modules](#_Toc25081_WPSOffice_Level2) [39](#_Toc25081_WPSOffice_Level2)

**[4. 一个页面上有大量的图片(大型电商网站),加载很慢,你有哪些方法优化这些图片的加载,给用户更好的体验](#_Toc20609_WPSOffice_Level1)** **[39](#_Toc20609_WPSOffice_Level1)**

**[5. Cookie,LocalStorage与SessionStorage三者的异同](#_Toc28190_WPSOffice_Level1)** **[40](#_Toc28190_WPSOffice_Level1)**

[5.1 生命周期](#_Toc27463_WPSOffice_Level2) [40](#_Toc27463_WPSOffice_Level2)

[5.2 作用范围](#_Toc30988_WPSOffice_Level2) [40](#_Toc30988_WPSOffice_Level2)

[5.3 可存储的数据大小](#_Toc16157_WPSOffice_Level2) [40](#_Toc16157_WPSOffice_Level2)

[5.4 存储的位置](#_Toc6525_WPSOffice_Level2) [41](#_Toc6525_WPSOffice_Level2)

[5.5 数据的存储形式](#_Toc21652_WPSOffice_Level2) [41](#_Toc21652_WPSOffice_Level2)

[5.6 与服务器端通信](#_Toc5339_WPSOffice_Level2) [41](#_Toc5339_WPSOffice_Level2)

[5.7 应用场景](#_Toc20175_WPSOffice_Level2) [41](#_Toc20175_WPSOffice_Level2)

**[6.简单用代码实现父子组件参数,方法的传递,分别写出父传子,子传父方式](#_Toc14377_WPSOffice_Level1)** **[41](#_Toc14377_WPSOffice_Level1)**

**[7.设计一个权限控制方式,如何根据用权限实现菜单的动态加载(19-3-22)](#_Toc21015_WPSOffice_Level1)** **[42](#_Toc21015_WPSOffice_Level1)**

**[8.怎么理解前后端分离](#_Toc14864_WPSOffice_Level1)** **[43](#_Toc14864_WPSOffice_Level1)**

[1. 彻底解放前端](#_Toc585_WPSOffice_Level2) [43](#_Toc585_WPSOffice_Level2)

[2. 提高工作效率，分工更加明确](#_Toc3785_WPSOffice_Level2) [43](#_Toc3785_WPSOffice_Level2)

[3. 局部性能提升](#_Toc21422_WPSOffice_Level2) [43](#_Toc21422_WPSOffice_Level2)

[4. 降低维护成本](#_Toc26686_WPSOffice_Level2) [43](#_Toc26686_WPSOffice_Level2)

**[9. 开发过程中有遇到哪些兼容问题,怎么解决的](#_Toc18185_WPSOffice_Level1)** **[43](#_Toc18185_WPSOffice_Level1)**

**[10.代码调试工具使用](#_Toc24289_WPSOffice_Level1)** **[43](#_Toc24289_WPSOffice_Level1)**

[1.如何开启调试模式](#_Toc22916_WPSOffice_Level2) [43](#_Toc22916_WPSOffice_Level2)

[2. 如何进行调试操作](#_Toc11970_WPSOffice_Level2) [44](#_Toc11970_WPSOffice_Level2)

# SDC

## HTTP状态码知道哪些？

1. 200：请求成功
2. 301：被请求的资源已永久移动到新位置，并且将来任何对此资源的引用都应该使用本响应返回的若干个 URI 之一。如果可能，拥有链接编辑功能的客户端应当自动把请求的地址修改为从服务器反馈回来的地址。除非额外指定，否则这个响应也是可缓存的。
3. 302：请求的资源临时从不同的 URI响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在Cache-Control或Expires中进行了指定的情况下，这个响应才是可缓存的。
4. 303：对应当前请求的响应可以在另一个 URI 上被找到，而且客户端应当采用 GET 的方式访问那个资源。这个方法的存在主要是为了允许由脚本激活的POST请求输出重定向到一个新的资源。这个新的 URI 不是原始资源的替代引用。同时，303响应禁止被缓存。当然，第二个请求（重定向）可能被缓存。
5. 304：重定向到浏览器缓存
6. 404：请求路径未找到，通常是发送请求的路径设置错误
7. 500：服务器错误

## [块级元素、行内元素和空元素（void）](https://blog.csdn.net/you23hai45/article/details/49051179)

* 块级元素：

<address>、<caption>、<dd>、<div>、<dl>、<dt>、<fieldset>、<form>、<h1>、<h2>、<h3>、 <h4>、<h5>、<h6>、<hr>、<legend>、<li>、<noframes>、<noscript>、<ol>、<ul>、<p>、 <pre>、<table>、<tbody>、<td>、<tfoot>、<th>、<thead>、<tr>

* 行内元素：  
  <a>、<abbr>、<acronym>、<b>、<bdo>、<big>、<br>、<cite>、<code>、<dfn>、<em>、<i>、<img>、<input>、<kbd>、<label>、<q>、<samp>、<select>、<small>、<span>、<strong>、<sub>、<sup>、<textarea>、<tt>、<var>
* 空元素（void）：  
  <area> <base> <col> <command> <embed> <link> <meta><keygen> <param> <source> <track> <wbr>

## HTML5文档类型和字符集

1.3.1 文档类型

<!DOCTYPE HTML>

1.3.2 字符集

UTF-8, GB2312

## 请写出jQuery绑定事件的方法，不少于两种

1. on
2. bind
3. deletegate
4. once

## jsp有哪些内置的对象(至少写出5个)， 注： 写出其常用方法

1. out对象
2. request对象
3. response对象
4. session对象
5. application对象

## js有哪些内置对象？

JavaScript常见的内置对象有Global,Object,Math,String,Array,Number,Function,Boolean,JSON等,其中Object是所有对象的基类,采用了原型继承方式。

其中Global, Math, JSON(内置对象，为本地对象的子集)为直接使用的对象，不需要实例化，其他的内置对象(本地对象)需要实例化再去使用。

## 优化页面加载速度的方法

（1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。

（2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数

（3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。

（4） 当需要设置的样式很多时设置className而不是直接操作style。

（5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

（6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

（8） 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。

对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如join查询），减少磁盘IO指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。

# 凡普金科

## DOM事件流包含那几个阶段？

三个阶段：事件捕捉，目标阶段，事件冒泡

## 事件委托的原理，优点和缺点

1. 原理：
   1. 事件冒泡机制
2. 优点
   1. 可以大量节省内存占用空间，减少事件注册
   2. 新增加的子元素也可以享受之前绑定的事件，对于动态内容尤为合适
3. 缺点
   1. 事件代理的应用场景仅限于上述需求(父元素，多个子元素，动态添加子元素)，如果把所有的事件都进行事件委托的话反而会出现事件误判，即本不该被触发的事件也被绑定上了事件。

## 请写出jQuery绑定事件的方法，不少于两种

1. on 较新版本推荐使用的标准的绑定事件的方式，
2. Bind 可以绑定自定义事件，由tragger触发
3. Deletegate 事件委托的形式
4. Once 事件只触发一次

## 什么是原型链，什么是作用域链，有什么作用？

1. 原型链：每个对象都有一个\_\_proto\_\_属性，这个属性指向的对象叫隐式原型对象，作用是：查找对象属性的属性的时候现在当前的对象中找如果没有找到，会沿着\_\_proto\_\_这条链去找，知道找到Object的原型，有则返回值没有返回undefined，\_\_proto\_\_这条链就是原型链
2. 作用域链：当在某一个作用域使用某一个变量的时候会现在当前的作用域中查找，如果没有会上级作用域去找，知道找到全局作用域，有则返回值，没有通常是报错。查找的链就是作用域链。

## 请使用原生JavaScript为一下li实现事件委托，点击后打印其对应的node-type属性值

|  |
| --- |
| <**ul id="test"**>  <**li node-type="1"**>111111</**li**>  <**li node-type="2"**>222222</**li**>  <**li node-type="3"**>333333</**li**> </**ul**>  <**script type="text/javascript"**>  **var *ulEle*** = **document**.getElementById(**'test'**);  ***ulEle***.addEventListener(**'click'**, **function** (event) {  **console**.log(event.**target**.getAttribute(**'node-type'**));  }) </**script**> |

## 请说出一下代码输出的内容，并解释为什么？

|  |
| --- |
| **var *num*** = 100; **function** *print*(){  **console**.log(num);  **var** num; } *print*(); // undefined 函数预解析，变量提升 |

* 1. 请说出一下代码输出的内容，并解释为什么？

|  |
| --- |
| (**function** (num) {  **console**.log(num); // 100 形参相当于在函数内部声明的变量var num = 100;  **var** num = 10; })(100) |

## 函数节流和函数防抖

前言： 函数节流和函数防抖都是优化高频率执行js代码的一种手段

### 函数节流

1. 定义：是指一定时间内js方法只跑一次。比如人的眨眼睛，就是一定时间内眨一次。这是函数节流最形象的解释。
2. 应用场景
   1. 在监听页面元素滚动事件的时候会用到。因为滚动事件，是一个高频触发的事件。
3. 代码实例：

|  |
| --- |
| *// 函数节流* **var *canRun*** = **true**; **document**.getElementById(**"target"**).onscroll = **function**(){  **if**(!***canRun***){  *// 判断是否已空闲，如果在执行中，则直接return* **return**;  }  ***canRun*** = **false**;  setTimeout(**function**(){  **console**.log(**"函数节流"**);  ***canRun*** = **true**;  }, 300); }; |

### 函数防抖

1. 定义：是指频繁触发的情况下，只有足够的空闲时间，才执行代码一次。比如生活中的坐公交，就是一定时间内，如果有人陆续刷卡上车，司机就不会开车。只有别人没刷卡了，司机才开车。
2. 应用场景
   1. 最常见的就是用户注册时候的手机号码验证和邮箱验证了。只有等用户输入完毕后，前端才需要检查格式是否正确，如果不正确，再弹出提示语。
3. 代码示例：

|  |
| --- |
| *// 函数防抖* **var *timer*** = **false**; **document**.getElementById(**"debounce"**).onscroll = **function**(){  clearTimeout(***timer***); *// 清除未执行的代码，重置回初始化状态* ***timer*** = setTimeout(**function**(){  **console**.log(**"函数防抖"**);  }, 300); }; |

## 已知构造函数A，请定义构造函数B，要求B继承A

|  |
| --- |
| **var** *A* = **function** (name) {  **this**.**name** = name; } *A*.**prototype** = {  fun1: **function**(){},  fun2: **function**(){} } |

表 1题目

|  |
| --- |
| **var** *A* = **function** (name) {  **this**.**name** = name; } *A*.**prototype** = {  fun1: **function**(){},  fun2: **function**(){} } *B*.**prototype** = **new** *A*(**'kobe'**); **function** *B*(){  } |

表 2答案

# 铂金智慧

## 如何垂直居中div(尽可能多的方法)

1. 不知道自己高度和父容器高度的情况下, 利用绝对定位只需要以下三行：

|  |
| --- |
| parentElement{  position:relative;  }  childElement{  position: absolute;  top: 50%;  transform: translateY(-50%);  } |

1. 若父容器下只有一个元素，且父元素设置了高度，则只需要使用相对定位即可

|  |
| --- |
| parentElement{  height:xxx;  }  .childElement {  position: relative;  top: 50%;  transform: translateY(-50%);  } |

1. Flex 布局：

不考虑兼容老式浏览器的话，用Flex布局简单直观一劳永逸：

|  |
| --- |
| parentElement{  display:flex;/\*Flex布局\*/  display: -webkit-flex; /\* Safari \*/  align-items:center;/\*指定垂直居中\*/} |

## em和rem的相同点和不同点

一、 rem的特点：

1、rem的大小是根据html根目录下的字体大小进行计算的。

2、当我们改变根目录下的字体大小的时候，下面字体都改变。

3、rem不仅可以设置字体的大小，也可以设置元素宽、高等属性。

二、em的特点：

1、字体大小是根据父元素字体大小设置的。

## 3.3 跨域问题和解决方案

跨域： 浏览器处于安全考虑规定了同源策略：协议，域名，端口号完全相同，如果有一个不一样就是不同源，不同源就是跨域，将不能正常访问其资源。

1.JSONP

回调函数+数据就是 JSON With Padding，简单、易部署。（做法：动态插入script标签，设置其src属性指向提供JSONP服务的URL地址，查询字符串中加入 callback 指定回调函数，返回的 JSON 被包裹在回调函数中以字符串的形式被返回，需将script标签插入body底部）。缺点是只支持GET，不支持POST（原因是通过地址栏传参所以只能使用GET）

2.document.domain 跨子域

document.domain 跨子域 （ 例如a.qq.com嵌套一个b.qq.com的iframe ，如果a.qq.com设置document.domain为qq.com 。b.qq.com设置document.domain为qq.com， 那么他俩就能互相通信了，不受跨域限制了。 注意：只能跨子域）

3. iframe

window.name + iframe ==> http://www.tuicool.com/articles/viMFbqV，支持跨主域。不支持POST

4.postMessage()

HTML5的postMessage()方法允许来自不同源的脚本采用异步方式进行有限的通信，可以实现跨文本档、多窗口、跨域消息传递。适用于不同窗口iframe之间的跨域

5.CORS

CORS（Cross Origin Resource Share）对方服务端设置响应头

设置相应头：”Access-Control-Allow-Origin“

CORS请求默认不发送Cookie和HTTP认证信息。如果要把Cookie发到服务器，一方面要服务器同意，指定Access-Control-Allow-Credentials字段。

6.服务端代理

服务端代理 在浏览器客户端不能跨域访问，而服务器端调用HTTP接口只是使用HTTP协议，不会执行JS脚本，不需要同源策略，也就没有跨越问题。简单地说，就是浏览器不能跨域，后台服务器可以跨域。（一种是通过http-proxy-middleware插件设置后端代理；另一种是通过使用http模块发出请求）

* 1. 请写出下边的运行结果

|  |
| --- |
| **var *bar*** = 0; **var *obj*** = {  **bar**: 1,  foo: **function**(){  **console**.log(**this**.**bar**);  } } **var *obj2*** = {  **bar**: 2,  **foo**: ***obj***.foo }  **var *obj3*** = {  **bar**: 3,  foo: **function** () {  **return *obj***.foo;  } }  **var *tempFoo*** = ***obj2***.**foo**; ***obj***.foo(); *//1* ***obj2***.**foo**(); *//2* ***obj3***.foo()(); *//0* ***tempFoo***(); *//0* |

# 东方之星

## 4.1 html标记中，用什么方法可以将整个表格在页面中居中 A、place=middle<br> B、type=middle<br> C、align=center<br> D、type=center

## 4.2 下列语句中，正确地执行当i等于5时执行一些语句的条件语句是(A)。

A.if(i==5)

B.if i=5 then

C.if i=5

D.if i==5 then

## 4.3 声明css有几种方式

1，内部引用，在html文档中在<style>标签里面写的css样式

|  |  |
| --- | --- |
| 1  2  3 | <style>    bdoy{font-size:14px;}  </style> |

2，外部引用，用<link>标签引用外部的[css文件](https://www.baidu.com/s?wd=css%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9mWFhmHfsPhD1uWD1mhDz0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHDdrj61rHT1" \t "https://zhidao.baidu.com/question/_blank)，将样式引用到html文档来。

|  |  |
| --- | --- |
| 1 | <link rel="stylesheet" type="text/css" href="style.css"> |

 3，在标签中使用，使用style属性将当前的标签样式改变。

|  |  |
| --- | --- |
| 1 | <div style="color:red"></div> |

## 4.4 前端页面优化方案

（1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。

（2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数

（3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。

（4） 当需要设置的样式很多时设置className而不是直接操作style。

（5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

（6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

（8） 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。

对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如join查询），减少磁盘IO指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。

## 4.5 适配不同浏览器

1) 流式布局

2) 宽度不写死，用百分比

3) 多媒体查询器: @media

4) em设置字体大小，需要提前设定好body的字体大小

5) flex伸缩盒模型布局

## 4.6 一个页面上有大量图片的时候加载很慢，你有哪些方法优化这些图片的加载，让用户的体验更好？

1. 图片懒加载，在页面上的未可视区域可以添加一个滚动条事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。
2. 如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。
3. 如果图片为css图片，可以使用CSSsprite，SVGsprite，Iconfont、Base64等技术。
4. 如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。
5. 如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

## 4.7 什么是ajax和json，优缺点

### 4.7.1 ajax

1) 定义：一种不用刷新整个页面就可以实现浏览器端和服务器通信的技术，它不是新的技术而是已有技术的整合，重点XMLHttpRequest对象

2) 优点：无刷新更新数据 异步与服务器通信 前后端负载均衡

3) 缺点： ajax干掉了Back和history功能，对浏览器机制的破坏 2）对搜索引擎支持较弱 3）违背了URI和资源定位的初衷

### 4.7.2 json

1) 定义：JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。

2) 优点：

1、轻量级的数据交换格式  
2、人们读写更加容易  
3、易于机器的解析和生成  
4、能够通过JavaScript中eval()函数解析JSON  
5、JSON支持多语言。包括：ActionScript, C, C#, ColdFusion, E, Java, JavaScript, ML, Objective CAML, Perl, PHP, **[Python](http://lib.csdn.net/base/11" \t "https://blog.csdn.net/ssisse/article/details/_blank" \o "undefined)**, Rebol, Ruby, and Lua.

3) 缺点：

JSON的非冗长性也不例外，为此JSON丢失了XML具有的一些特性。命名空间允许不同上下文中的相同的信息段彼此混合，然而，显然在JSON中已经找不到了命名空间。JSON与XML的另一个差别是属性的差异，由于JSON采用冒号赋值，这将导致当XML转化为JSON时，在标识符（XML CDATA）与实际属性值之间很难区分谁应该被当作文本考虑。   
　　另外，JSON片段的创建和验证过程比一般的XML稍显复杂。从这一点来看，XML在开发工具方面领先于JSON。

# 看山科技

## 手写三列布局，左右固定，中间自适应

### 方案一：定位position

|  |
| --- |
| **\*** {  **margin**:0;  **padding**:0; } **#container**{  **position**: **relative**;  **width**:100%; }  **#left** {  **position**: **absolute**;  **left**:0;  **height**:200**px**;  **width**:200**px**;  **background**: **red**; } **#right** {  **position**: **absolute**;  **right**:0;  **top**:0;  **height**:200**px**;  **width**:200**px**;  **background**: **green**; }  **#center** {  **height**:200**px**;  **margin**:0 200**px**;  **background**: **yellow**; } |
| <**div id="container"**>  <**div id="left"**></**div**>  <**div id="center"**></**div**>  <**div id="right"**></**div**> </**div**> |

### 方案二： 浮动

|  |
| --- |
| **\*** {  **margin**:0;  **padding**:0; } **#container**{  **width**:100%; }  **#left** {  **float**: **left**;  **height**:200**px**;  **width**:200**px**;  **background**: **red**; } **#right** {  **float**: **right**;  **height**:200**px**;  **width**:200**px**;  **background**: **green**; }  **#center** {  **height**:200**px**;  **margin**:0 200**px**;  **background**: **yellow**; } |
| <**div id="container"**>  <**div id="left"**></**div**>  <**div id="right"**></**div**>  <**div id="center"**></**div**> </**div**> |

### 方案三：圣杯布局，利用margin-left为负值 + 浮动

|  |
| --- |
| **\*** {  **margin**:0;  **padding**:0; } **#container**{  **width**:100%; }  **#left** {  **float**: **left**;  **height**:200**px**;  **width**:200**px**;  **background**: **red**;  **margin-left**: -100%; } **#right** {  **float**: **left**;  **height**:200**px**;  **width**:200**px**;  **background**: **green**;  **margin-left**: -200**px**; }  **#centerContainer** {  **float**: **left**;  **width**:100%;  **height**:200**px**; } **#center** {  **height**:200**px**;  **margin**:0 200**px**;  **background**: **yellow**; } |
| <**div id="container"**>  <**div id="centerContainer"**>  <**div id="center"**></**div**>  </**div**>  <**div id="left"**></**div**>  <**div id="right"**></**div**> </**div**> |

## 列出position的相关属性，并说出其作用

position用来设定元素的定位类型，有absolute（[绝对定位](https://www.baidu.com/s?wd=%E7%BB%9D%E5%AF%B9%E5%AE%9A%E4%BD%8D&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4rHPhPjK-nWbzn1T1m1cL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10YPHR1PHmL" \t "https://zhidao.baidu.com/question/_blank)）、relative（[相对定位](https://www.baidu.com/s?wd=%E7%9B%B8%E5%AF%B9%E5%AE%9A%E4%BD%8D&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4rHPhPjK-nWbzn1T1m1cL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10YPHR1PHmL" \t "https://zhidao.baidu.com/question/_blank)）、static（静态定位，默认值）、fixed（固定定位）四种。  
static：默认。位置设置为 static 的元素，它始终会处于页面流给予的位置（static 元素会忽略任何 top、bottom、left 或 right 声明）。  
relative：位置被设置为 relative 的元素，可将其移至相对于其正常位置的地方，因此 "left:20" 会将元素移至元素正常位置左边 20 个像素的位置。  
absolute：位置设置为 absolute 的元素，可定位于相对于包含它的元素的指定坐标。此元素的位置可通过 "left"、"top"、"right" 以及 "bottom" 属性来规定。  
fixed：位置被设置为 fixed 的元素，可定位于相对于浏览器窗口的指定坐标。此元素的位置可通过 "left"、"top"、"right" 以及"bottom" 属性来规定。不论窗口滚动与否，元素都会留在那个位置。工作于 IE7（strict 模式）。

## 5.3 box-sizing的内容

box-sizing 属性允许您以特定的方式定义匹配某个区域的特定元素。

例如，假如您需要并排放置两个带边框的框，可通过将 box-sizing 设置为 "border-box"。这可令浏览器呈现出带有指定宽度和高度的框，并把边框和内边距放入框中。

## 5.4 判断数组[1,2,3] == [1,2,3], [1,2,3] === [1,2,3]

答案： false, false

## 5.5 js中不使用临时变量，交换两个变量的值

|  |
| --- |
| **var *a*** = 1; **var *b*** = 2; ***a*** = ***a*** + ***b***; ***b*** = ***a*** - ***b***; ***a*** = ***a*** - ***b***; **console**.log(***a***, ***b***); |

## 5.6 完成功能函数confirmEnding，判断给定的字符串是否已指定的字符串结尾

|  |
| --- |
| **function** *confirmEnding*(str, target) {  **return** str.endsWith(target);} |

## 5.7 计算整数阶乘的方法

理解： 什么是整数阶乘？

一个正整数的阶乘（factorial）是所有小于及等于该数的[正整数](https://baike.baidu.com/item/%E6%AD%A3%E6%95%B4%E6%95%B0" \t "https://baike.baidu.com/item/%E9%98%B6%E4%B9%98/_blank)的[积](https://baike.baidu.com/item/%E7%A7%AF" \t "https://baike.baidu.com/item/%E9%98%B6%E4%B9%98/_blank)，并且0的阶乘为1。自然数n的阶乘写作n!。1808年，基斯顿·卡曼引进这个表示法。

亦即n!=1×2×3×...×n。阶乘亦可以[递归](https://baike.baidu.com/item/%E9%80%92%E5%BD%92" \t "https://baike.baidu.com/item/%E9%98%B6%E4%B9%98/_blank)方式定义：0!=1，n!=(n-1)!×n。

|  |
| --- |
| *// n! = 1 \* 2 \* 3 \* ...\* (n - 1) \* n* **function** *test*(num) {  **return** num > 1? num \* *test*(num - 1): 1 } **console**.log(*test*(6)); |

## 5.8 按要求输出数组的内容

let arr = ['html', 'css', 'js', 'angular', 'vue'];  
// 输出 ['js', 'css', 'html', 'vue', angular]

|  |
| --- |
| **let** [***a***, ***b***, ***c***, ***d***, ***e***] = ***arr***; **console**.log([***c***, ***b***, ***a***, ***d***, ***e***]); |

# 商壤集团

## 6.1 在一个方法中定义变量不写var什么结果

## 6.2 浏览器渲染页面的机制

## 6.3 ajax同步异步的区别

## 6.4 div和table布局比较

## 6.5 怎么优化代码

## 6.6 npm包管理

# 7. 索为科技

## 7.1 都做过哪些项目，公司有几个前端，你所担任的职责。

## 7.2 是否独立地搭建过项目，如何搭建一个前端项目。

## 7.3 公司如何发送ajax请求

## 7.4 commonJs，与ES6规范的区别

## 7.5 vue的主线程是什么？

## 7.6 不使用服务器代理，前台与后台如何实现联调。

# 8. 用友科技

## 8.1 清除浮动的方案

## 8.2 同源策略，及解决跨域的常见方案和优缺点

## 8.3 移动端布局自适应(适配)方案

## 8.4 cookie localStorage sessionStorage三者的区别和应用场景

## 8.5 jQuery中$(function(){})代表什么事件，和window.onload的区别是什么？

## 8.6 事件点透的原因及解决方案

# 9. 中科软

## 1. XHTML和HTML的区别

## 2. 行内元素和块级元素的区别

## 3. css引入的方式有哪些， link和@import的区别

## 4. title和alt的区别

## 5. 描述css的reset的作用

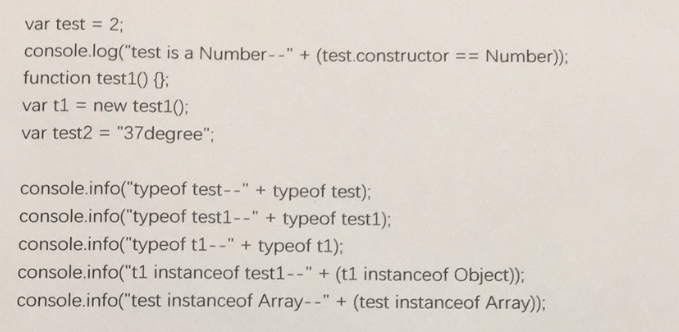
## 6. split和join()的区别

## 7. 写一个获取非行间样式的方法

## 8. HTML如何嵌入音频

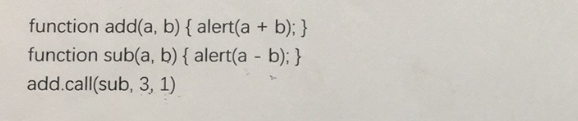
# 凯特伟业

## 判断代码输出结果



答案: true “number” “function” “object” true false

## 解释call与apply的区别,并写出下面代码输出的结果:



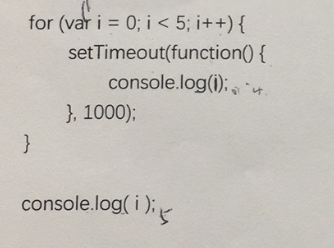
区别:

call第一个参数是修改原函数中this的指向,第二个参数开始是依次传入原函数所需要的参数,用逗号隔开

apply第一个参数是修改原函数中this的指向,第二个参数要求传入一个数组,数组中存储这原函数所需要的参数

结果: 4

## 3.阅读代码:

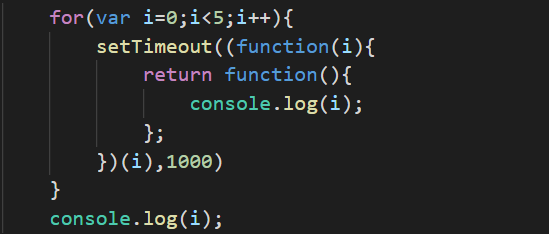


### 3.1 写出输出结果

答案:6个5

### 3.2 如果期望代码的输出变成:5,0,1,2,3,4,该如何改造代码?

答案:使用闭包来解决

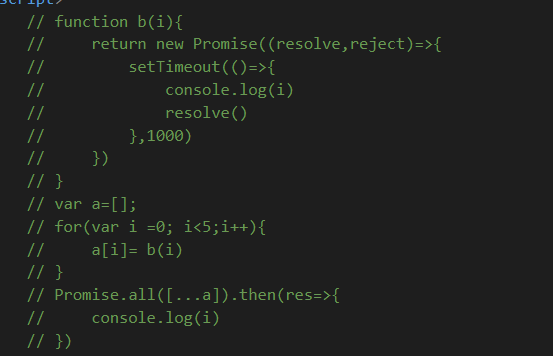


### 3.3 如果期望代码的输出变成:0,1,2,3,4,5,并且要求源代码的代码块中的循环和两处console.log不变,该怎么改造代码(建议使用:Promise)?

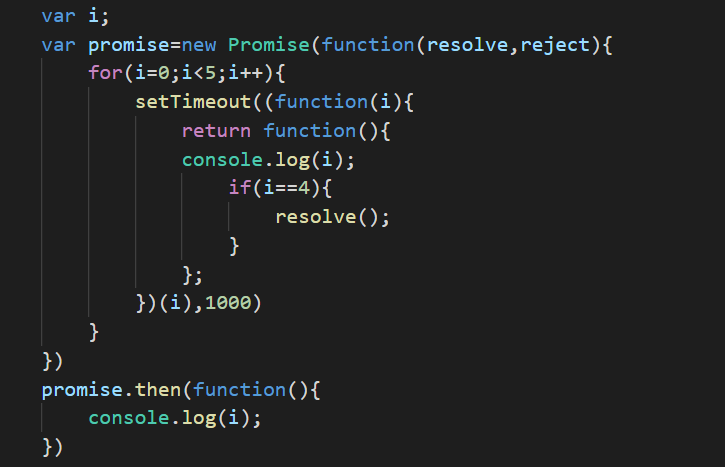
答案:

Promise版:

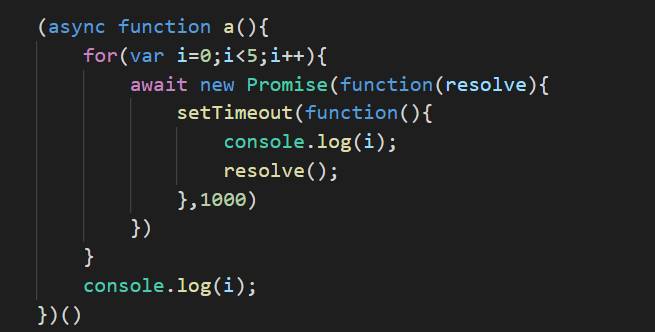
方案一:



方案二:



Async版:



## 设计:div隐藏滚动条,且内容可以滚动(不可用css3)

方法1:



方法2:



# 11.汉克时代

## 1.vue-router刷新的几种方式

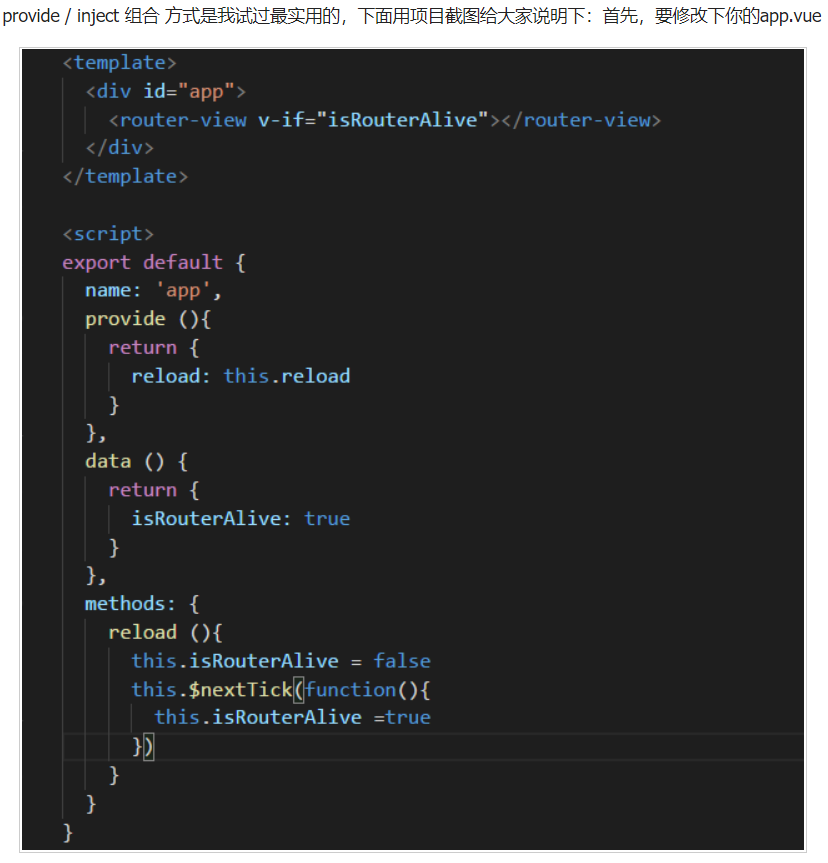
方案一:使用location.reload()

方案二:使用this.$router.go(0)

以上两种方案相当于ctrl+F5强制刷新,整个页面重新加载的时候会出现白屏的情况,体验不好

方案三:在项目新建一个空白页面,先跳转到这个空白页，然后再立马跳转回来,不会出现白屏现象

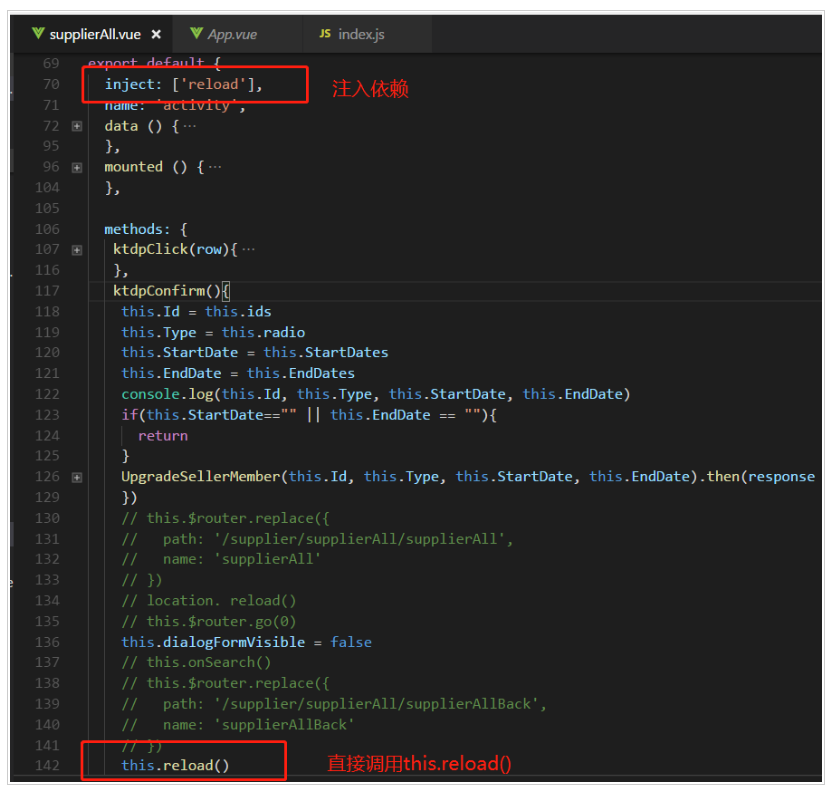
方案四:



通过声明reload方法，控制router-view的显示或隐藏，从而控制页面的再次加载，这边定义了

isRouterAlive //true or false 来控制

然后在需要当前页面刷新的页面中注入App.vue组件提供（provide）的 reload 依赖，然后直接用this.reload来调用就行



## 2.说出至少4中vue当中的指令和他的用法?

**1、v-text**

v-text是用于操作纯文本，它会替代显示对应的数据对象上的值。当绑定的数据对象上的值发生改变，插值处的内容也会随之更新。注意：此处为单向绑定，数据对象上的值改变，插值会发生变化；但是当插值发生变化并不会影响数据对象的值。其中：v-text可以简写为{{}},并且支持逻辑运算。例如：

<div id="app">

{{ message }}

</div>

var app = new Vue({

el : '#app',

data : {

message : 'hello world'

}

})

注：vue中有个指令叫做 v-once 可以通过v-once与v-text结合，实现仅执行一次性的插值

<span v-once>这个将不会随msg属性的改变而改变: {{ msg }}</span>

**2、v-html**

v-html用于输出html，它与v-text区别在于v-text输出的是纯文本，浏览器不会对其再进行html解析，但v-html会将其当html标签解析后输出。

<div id="app">

<p v-html="html"></p>

</div>

var app = new Vue({

el: "#app",

data: {

html: "<b style='color:red'>v-html</b>"

}

});

**3、v-model**

v-model通常用于表单组件的绑定，例如input，select等。它与v-text的区别在于它实现的表单组件的双向绑定，如果用于表单控件以外标签是没有用的。

<div id="app">

<input v-model="message " />

</div>

var app = new Vue({

el : '#app',

data : {

message : 'hello world'

}

})

4.v-if

v-if 是“真正的”条件渲染，因为它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建。  
v-show 就简单得多——不管初始条件是什么，元素总是会被渲染，并且只是简单地基于 CSS 进行切换。  
一般来说， v-if 有更高的切换开销，而 v-show 有更高的初始渲染开销。  
因此，如果需要非常频繁地切换，则使用 v-show 较好；如果在运行时条件不太可能改变，则使用 v-if 较好。

<template>

<div id="app">

<div v-if="isIf">

if

</div>

<div v-show="ifShow">

show

</div>

<button @click="toggleShow">toggle</button>

</div>

</template>

<script>

export default {

name: 'app',

data() {

return {

isIf : true,

ifShow : true,

loginType : "username"

}

},

methods: {

toggleShow : function(){

this.ifShow = this.ifShow ? false : true;

this.isIf = this.isIf ? false : true;

}

}

}

</script>

5.v-on

1、用于监听DOM事件，典型的就是v-on:click，处理的方法放在methods属性里会默认传一个参数。

<button @click="test">点击</button>

methods: {

test: function (evt) {

console.log(evt);

}

}

这里的evt，是标准的鼠标点击事件，类似jquery的click事件的回调函数中的参数。  
可以通过this来找到data属性里的值，例如：

data: {

items: "test"

},

methods: {

test: function (evt) {

console.log(this.items);

console.log(evt);

}

}

这里的this.items，就是data的items这个变量。

## vuex有哪几种属性?说出具体使用场景

1. state

Vuex使用单一状态树，即用一个对象就包含了全部的状态数据。state作为构造器选项，定义了所有我们需要的基本状态参数。

1. getters

基于state的getter计算属性

1. mutations

提交mutation是更改Vuex中的store中的状态的唯一方法

mutation必须是同步的，如果要异步需要使用action

1. actions

action 提交的是 mutation，而不是直接变更状态。

action 可以包含任意异步操作

1. modules

使用单一状态树，导致应用的所有状态集中到一个很大的对象。但是，当应用变得很大时，store 对象会变得臃肿不堪。

为了解决以上问题，Vuex 允许我们将 store 分割到多个模块（module）。每个模块拥有自己的 state、mutations、actions、getters

## 一个页面上有大量的图片(大型电商网站),加载很慢,你有哪些方法优化这些图片的加载,给用户更好的体验

图片懒加载，在页面上的未可视区域可以添加一个滚动条事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。

如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。

如果图片为css图片，可以使用CSSsprite，SVGsprite，Iconfont、Base64等技术。

如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。

如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

## Cookie,LocalStorage与SessionStorage三者的异同

### 生命周期

**Cookie**: 没有设置时间:则关闭浏览器失效设置 设置时间:则指定时间后失效

**LocalStorage**: 如果没有手动清除,永久保存

**SessionStorage**: 关闭设置SessionStorage的标签页便会失效,刷新该页面,不会失效,项目页面在一个新窗口打开或者关闭浏览器重新打开也会消失

### 作用范围

**Cookie**: 当前页面指向的文件目录及其子目录,当然设置setcookie函数的参数4,5,可以改变作用范围.同一浏览器的不同页面可以共享cookie,但是cookie无法跨域,这使得我在vue项目没有用它

**LocalStorage**: 同一个浏览器的不同页面可以共享信息,不同的浏览器之间数据不能共享

**SessionStorage**: 设置SessionStorage的标签页

### 可存储的数据大小

**Cookie**: 4KB左右

**LocalStorage**: 20MB

**SessionStorage**: 5MB

### 存储的位置

**Cookie**: 未设置保存时间:保存在运行内存中 设置保存时间:保存在硬盘里

**LocalStorage**: 保存在硬盘里

**SessionStorage**: 保存在运行内存中

### 数据的存储形式

**Cookie**: 纯纯的字符串:即“名称=值;expires=时间”读的时候也是这玩意,比较难用

**LocalStorage**: 也是字符串,以键值对的形式存储,常利用JSON对象的方法去进行读写数据时的涉及到的转化

**SessionStorage**: 同LocalStorage

### 与服务器端通信

**Cookie**: 每次都会携带在http头中,如果页面中有太多cookie会影响性能

**LocalStorage**: 不参合与服务器的通信,减少了网络请求

**SessionStorage**: 同LocalStorage

### 应用场景

**Cookie**: 与服务端的session技术配合判断用户是否登录

**LocalStorage**: 也可以记录网站用户名,显示搜索历史,h5游戏的游戏缓存,我前两天写的博客内容也被存下来了

**SessionStorage**: 根据需求

## 6.简单用代码实现父子组件参数,方法的传递,分别写出父传子,子传父方式

**父传子方法**:

将父组件中的状态或方法(假设名称为data),用标签属性的方式传入到子组件中

父组件:

<Child :data=”data” />

子组件:

props:[“data”]

**子传父方法**:

父组件中声明一个方法(假如名称为handleData),用$emit将子组件的状态(data)传入到父组件中

父组件:

methods:{

handleData(data){

this.data=data;

}

}

<Child v-on:handleData=”handleData” />

子组件:

methods:{

handleClick(){

this.$emit(“handleData”,this.data);

}

}

**任意组件之间传值方法**:

除此之外还可以使用公共的bus.js作为中间仓库来进行传值

## 7.设计一个权限控制方式,如何根据用权限实现菜单的动态加载(19-3-22)

后台数据库:

角色表: 包含了所有有权限菜单标识列表

用户表: 包含所属角色的ID

前台:

当用户登陆时, 会同时得到所属角色下的权限菜单标识列表

根据菜单标识列表动态生成有权限查看的菜单列表

## 8.怎么理解前后端分离

前端负责浏览器端（客户端）用户交互界面和逻辑等，显示数据；后端负责数据的处理和存储等，提供数据

好处:

1. 彻底解放前端

前端不再需要向后台提供模板或是后台在前端html中嵌入后台代码

1. 提高工作效率，分工更加明确

前后端分离的工作流程可以使前端只关注前端的事，后台只关心后台的活，两者开发可以同时进行

3. 局部性能提升

通过前端路由的配置，我们可以实现页面的按需加载，无需一开始加载首页便加载网站的所有的资源，服务器也不再需要解析前端页面，在页面交互及用户体验上有所提升

4. 降低维护成本

通过目前主流的前端MVC框架，我们可以非常快速的定位及发现问题的所在，客户端的问题不再需要后台人员参与及调试，代码重构及可维护性增强

## 开发过程中有遇到哪些兼容问题,怎么解决的

<https://blog.csdn.net/shunqixing/article/details/80265876>(从中选择3个左右自己能表达清楚的)

## 10.代码调试工具使用

1.如何开启调试模式

1). 添加语debugger句: 程序运行前 此方式用打包后才运行的项目

2). 添加(打)断点: 程序运行前或者过程中 此方式用运行源码js

2. 如何进行调试操作

resume: 恢复程序执行(可能执行完或者进入下一个断点处)

step ove: 单步跳转, 尝试执行完当前语句, 进入下一条(如果内部有断点, 自动进入内部断点处)

step into: 跳入, 进入当前调用函数内部

step out: 跳出, 一次性执行完当前函数后面所有语句,并出去

deactivate breakpoints: 使所有断点暂时失效

call stack: 显示是程序函数调用的过程

scop: 当前执行环境对应的作用域中包含的变量数据

breakpoints: 断点列表

# 12.北京博雅智学

## 1.写出下面代码运行后K的值:

for(i=0,j=0;i<10,j<6;i++,j++){

k=i+j;

}

答案:10

## 2.javascript的数据类型有哪些?

基本数据类型有6种:Number,String,Boolean,Undefined,Null,以及ES6新增加的Symbol

对象数据类型:Object, 2种特别的对象类型: Function和Array

## 3.请写出添加,删除,替换,插入到某个节点的方法

**原生js**:

添加:appendChild()

删除:removeChild()

替换:replaceChild(新节点,旧节点)

插入:insertBefore(插入节点，被插节点)

**jQ**：

添加:

append() （旧节点添加新节点后面）

appendTo() （新节点添加到旧节点后面）

prepend() （旧节点添加新节点的前面）

prependTo() （新节点添加到旧节点的前面）

删除:remove()

清空: empty()

插入:

after() \_\_\_\_\_\_// 旧节点后面插入 新节点

before() \_\_\_\_\_\_ //旧节点前面插入 新节点

insertAfter() \_\_\_\_\_\_\_\_//新节点插到旧节点后

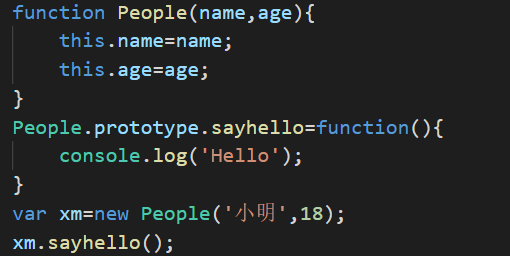
insertBefore() \_\_\_\_\_\_\_\_\_//新节点插到旧节点前

替换:replaceWith()

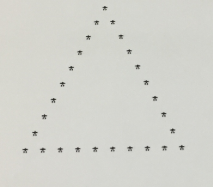
## 4.Javascript中的方法是否可以重载?并简述原因?

答案:不可以重载,因为在js中根本就不存在像C#中的那种方法重载，而有的只是方法的覆盖，当你在js中敲入两个或多个同名的方法的时候，不管方法 （函数）的参数个数怎么个不同，这个方法名只能属于最后定义的那个函数。也就是说后一个方法会把前面同名的方法给覆盖掉。

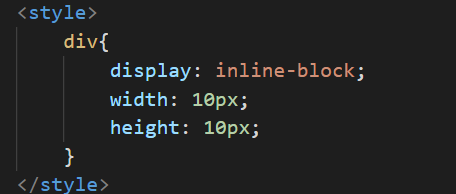
## 5.使用javascript脚本语言定义一个类people,定义两个属性name,age,然后对people类进行扩展,扩展一个sayhello()方法,实现打印Hello,最后创建一个name为小明,age为18岁的实例xm,并调用sayhello()方法



## 6.编写脚本实现打印如下效果,.要求写出完整代码



Css部分:



Js部分: 