# web 前端面试题精简附加版（仅针对前端二期）

## html+css 部分

1、怎么让一个不定宽高的 DIV，垂直水平居中？

答：1）使用 CSS 方法。
父盒子设置：display:table-cell; text-align:center;vertical-align:middle;
Div 设置：display:inline-block;vertical-align:middle;
2）使用 CSS3 transform。
父盒子设置：position:relative
Div 设置：transform: translate(-50%,-50%);position: absolute;top: 50%;left: 50%;

2、选择器优先级是怎样的？

答：！important>行内样式>id 选择器>类选择器>标签选择器>通配符>继承
权重算法：
（0，0，0，0）==》第一个 0 对应的是 important 的个数，第二个 0 对应的是 id 选择器的个数，第三个 0 对应的类选择器的个数，第四个 0 对应的是标签选择器的个数，就是当前选择器的权重。
比较：
先从第一个 0 开始比较，如果第一个 0 大，那么说明这个选择器的权重高，如果第一个相同，比较第二个，依次类推。

3、html 常见兼容性问题？

答：1.双边距 BUG float 引起的 使用 display:inline
2.3 像素问题 使用 float 引起的 使用 dislpay:inline -3px
3.超链接 hover 点击后失效 使用正确的书写顺序 link visited hover active
4.Ie z-index 问题 给父级添加 position:relative
5.Png 透明 使用 js 代码 改
6.Min-height 最小高度 ！Important 解决
7.select 在 ie6 下遮盖 使用 iframe 嵌套
8.为什么没有办法定义 1px 左右的宽度容器（IE6 默认的行高造成的，使用 over:hidden,zoom:0.08 line-height:1px）
9.IE5-8 不支持 opacity，解决办法：

```
.opacity {
    opacity: 0.4
    filter: alpha(opacity=60); /* for IE5-7 */
    -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; /* for IE 8*/
}
```

10. IE6 不支持 PNG 透明背景，解决办法：IE6 下使用 gif 图片或者用插件

## 4、CSS3 新特性有哪些？

1. 颜色:新增 RGBA,HSLA 模式
2. 文字阴影（text-shadow、）
3. 边框：圆角（border-radius）边框阴影：box-shadow
4. 盒子模型:box-sizing
5. 背景:background-size 设置背景图片的尺寸 background-origin 设置背景图片的原点，background-clip 设置背景图片的裁切区域,以”，”分隔可以设置多背景,用于自适应布局
6. 渐变:linear-gradient、radial-gradient
7. 过渡:transition,可实现动画
8. 自定义动画
9. 在 CSS3 中唯一引入的伪元素是 ::selection.
10. 媒体查询，多栏布局
11. border-image
12. 2D 转换:transform:translate(x,y) rotate(x,y) skew(x,y) scale(x,y)
13. 3D 转换

## 5、清除浮动的几种方式？

答：1、父级 div 定义 height

原理：父级 div 手动定义 height，就解决了父级 div 无法自动获取到高度的问题。 简单、代码少、容易掌握 ,但只适合高度固定的布局.

2、结尾处加空 div 标签 clear:both

原理：在浮动元素的后面添加一个空 div 兄弟元素，利用 css 提高的 clear:both 清除浮动，让父级 div 能自动获取到高度 ,如果页面浮动布局多，就要增加很多空 div，让人感觉很不好 .

3、父级 div 定义 伪类:after 和 zoom

```
/*清除浮动代码*/
.clearfix:after{
    content:"";
    display:block;
```

```
    visibility:hidden;
    height:0;
    line-height:0;
    clear:both;
}
.clearfix{zoom:1}
```

原理：IE8 以上和非 IE 浏览器才支持:after，原理和方法 2 有点类似，zoom(IE 转有属性)可解决 ie6,ie7 浮动问题 ,推荐使用，建议定义公共类，以减少 CSS 代码。

4、父级 div 定义 overflow:hidden

超出盒子部分会被隐藏,不推荐使用.

5、双伪元素法:

```
.clearfix:before,.clearfix:after {
        content: "";
        display: block;
        clear: both;
    }
    .clearfix {
        zoom: 1;
    }
```

6、如何实现左侧盒子固定宽度，右侧盒子根据屏幕大小自适应？

答：左侧盒子左浮动，给固定宽度，右侧盒子宽度 100%；

```
<div class="body">
    <div class="left"></div>
    <div class="right"></div>
</div>
.left{
    float: left;
    width:200px;
    height:300px;
}
.right{
    height:300px;
}
```

# js 部分

## 1、什么是预解析？

答：js 执行主要分两个阶段：预解析和执行期，在预解析阶段，会把变量名和函数声明提升。

## 2、谈谈 js 作用域和闭包？

答：简单的说，作用域是针对变量的，比如我们创建一个函数 a1，函数里面又包了一个子函数 a2。此时就存在三个作用域：
全局作用域—a1 作用域—a2 作用域；即全局作用域包含了 a1 的作用域，a2 的作用域包含了 a1 的作用域。
当 a1 在查找变量的时候会先从自身的作用域区查找，找不到再到上一级 a2 的作用域查找，如果还没找到就到全局作用域区查找，这样就形成了一个作用域链。

理解闭包首先要理解，js 垃圾回收机制，也就是当一个函数被执行完后，其作用域会被收回，如果形成了闭包，执行完后其作用域就不会被收回。
如果某个函数被他的父函数之外的一个变量引用，就会形成闭包。
闭包的作用，就是保存自己私有的变量，通过提供的接口（方法）给外部使用，但外部不能直接访问该变量。

## 3、什么是原型链？

答：Javascript 是面向对象的，每个实例对象都有一个__proto_属性，该属性指向它原型对象，这个实例对象的构造函数有一个原型属性 prototype，与实例的__proto__属性指向同一个对象。当一个对象在查找一个属性的时，自身没有就会根据__proto__ 向它的原型进行查找，如果都没有，则向它的原型的原型继续查找，直到查到 Object.prototype._proto_为 nul，这样也就形成了原型链。

## 4、实现继承的方法有什么？

答：
（1）借用构造函数。也叫伪造对象或经典继承。
思路：在子类构造函数的内部调用超类型构造函数。可以通过使用 apply（）和 call()方法在新创建的对象上执行构造函数。
缺点：方法都在构造函数中定义，函数的复用就无从谈起。在超类型的原型中定义的方法，对子类而言也是不可见的，结果所有的类型都只能使用构造函数模式。

```
function SuperType(){
    this.colors = ["red", "blue", "green"];
}

function SubType(){
    //继承了 SuperType
    SuperType.call(this);
}

var instance1 = new SubType();
instance1.colors.push("black");
alert(instance1.colors);    //"red,blue,green,black"

var instance2 = new SubType();
alert(instance2.colors);    //"red,blue,green"
```

（2）组合继承。也叫伪经典继承。指的是将原型链和借用构造函数的技术组合到一起，从而发挥二者之长。
思路：使用原型链实现对原型属性属性和方法的继承，通过借用构造函数来实现实例属性的继承。
优点：既通过在原型上定义方法实现了函数复用，又能保证每一个实例都有它自己的数组。组合继承避免了原型链和借用构造函数的缺陷，融合了他们的优点，成为 JavaScript 中常用的继承模式。

（3）原型链继承。
思路：借助原型可以基于已有的对象创建对象，同时还不必因此创建自定义类型。
在 object()函数内部，先创建一个临时的构造函数，然后将传入的对象作为这个构造函数的原型，最后返回了这个临时类型的一个新实例。

```
Function object(o){
Function    F(){};
F.prototype=o;
Return new F();
}
```

（4）寄生式继承。
思路：创建一个仅用于封装继承过程的函数，该函数在内部以某种方式来增强对象，最后再像真的是它做了所有的工作一样返回对象。

```
Function createAonter(original){
Var clone=object(original);//通过调用函数创建一个新对象
```

```
Clone.sayHi=function(){    //以某种方式来增强这个对象
Alert(“hi”);
}
Return clone; //返回这个对象
}
```

缺点：使用寄生式继承来为对象添加函数，会由于不能做到函数复用二降低效率，这一点和构造函数模式类似。

（5）寄生组合式继承。是 JavaScript 最常用的继承模式。
思路：通过借用构造函数来继承属性，通过原型链的混成形式来继承方法。
本质上，就是使用寄生式继承来继承超类型的原型，然后再将结果指定给子类型的原型。

开发人员普遍认为寄生组合式继承时引用类型最理想的继承范式。

Extend（）方法才用了这样的方式。

## 5、说说你对 this 的理解？

答：this 是一个关键字，它代表函数运行时，自动生成的一个内部对象，只能在函数内部使用。

1.作为纯粹的函数调用 this 指向全局对象
2.作为对象的方法调用 this 指向调用对象
3.作为构造函数被调用 this 指向新的对象（new 会改变 this 的指向）
4.apply 调用 this 指向 apply 方法的第一个参数

## 6、用 JavaScript 实现冒泡排序。数据为 23、45、18、37、92、13、24

答：

```
//外循环控制轮数(元素个数减一)
for(var i=0;i<arr.length-1;i++){
    //开闭原则中的开关（默认是开启的）
    var bool = true;

    //内循环控制次数(元素个数减一)
    for(var j=0;j<arr.length-1-i;j++){
        //注意：1.怎么比较？（相邻的两个数比较）    2.什么情况下交换位置。（前边数大）
        //如果判断条件和交换元素位置呢
        if(arr[j] > arr[j+1]){
            //交换位置
            var temp = arr[j];
```

```
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                bool = false;
            }
            n++;
        }
        m++;
        //内循环中的 if 语句没有被执行
        if(bool){
            break;
        }
    }
```

数组方法 sort

如果调用该方法时没有使用参数，将按字母顺序对数组中的元素进行排序，说得更精确点，是按照字符编码的顺序进行排序。

## 7、jquery 中常用的 api 有哪些？

答：addClass()，removeClass()，append()，appendTo()，prepend()，prependTo()，css()，show()，hide()，slideDown()，slideUp()，fadeIn()，fadeOut()，animate()等。

## 8、jquery.extend 与 jquery.fn.extend 的区别？

答：Jquery.extend 用来扩展 jQuery 对象本身；jquery.fn.extend 用来扩展 jQuery 实例。

## 9、AJAX 请求数据步骤是什么？传输的数据是用的暗文还是明文？

答：var xhr;
xhr = new XMLHttpRequest(); //创建一个异步对象
xhr.open("Get", "test.ashx", true);//Get 方式括号中的三个参数分别为：1.发送请求的方式 2.请求的页面 3.是否异步
//xhr.open("post","test.ashx",true);
//xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); Post 方式发送数据

//这个回调函数主要用来检测服务器是否把数据返回给异步对象
xhr.setRequestHeader("If-Modified-Since","0");//设置浏览器不使用缓存
xhr.onreadystatechange = function () {
if (xhr.readystate == 4) {
//readyState 属性指出了 XMLHttpRequest 对象在发送/接收数据过程中所处的几个状态。

XMLHttpRequest 对象会经历 5 种不同的状态。
//0：未初始化。对象已经创建，但还未初始化，即还没调用 open 方法；
//1：已打开。对象已经创建并初始化，但还未调用 send 方法；
//2：已发送。已经调用 send 方法，但该对象正在等待状态码和头的返回；
//3：正在接收。已经接收了部分数据，但还不能使用该对象的属性和方法，因为状态和响应头不完整；
//4：已加载。所有数据接收完毕

```
if(xhr.status==200){ //检测服务器返回的响应报文的状态码是否为 200
    alert(xhr.responseText);//服务器返回的 Response 数据
    //解析服务器返回的 jason 格式的数据
    var s=xhr.responseText;
    var json=eval("("+s+")");
    alert(jason.data);
    }
    };
};
xhr.send(null);//异步对象发送请求
//xhr.send("txtName=roger&txtPwd=123"); 以 post 方式发送数据
```

ajax 中 get 和 post 方式请求数据都是明文的。

## 10、怎么实现跨域问题？

答：

| URL | 说明 |
| --- | --- |
| http://www.a.com/b.js | 同一域名下 |
| http://www.a.com:8000/a.js | 同一域名，不同端口 |
| https://www.a.com/b.js | 同一域名，不同协议 |
| http://script.a.com/b.js | 主域相同，子域不同 |
| http://a.com/b.js | 同一域名，不同二级域名 |
| http://www.a.com/b.js | 不同域名 |

对于端口和协议的不同，只能通过后台来解决。我们要解决的是域名不同的问题。

1. 下面是用 php 进行的设置，“*”号表示允许任何域向我们的服务端提交请求：

```
header{"Access-Control-Allow-Origin: *"}
```

2.（二）JSONP(JSON with Padding 填充式 JSON 或参数式 JSON)

在 js 中，我们虽然不能直接用 XMLHttpRequest 请求不同域上的数据时，但是在页面上引入不同域上的 js 脚本文件却是可以的，jsonp 正是利用这个特性来实现的。

JSONP 由两部分组成：回调函数和数据。回调函数是当响应到来时应该在页面中调用的函数，而数据就是传入回调函数中的 JSON 数据。

```
<script type="text/javascript">
    function dosomething(jsondata){
        //处理获得的json数据
    }
</script>
<script src="http://example.com/data.php?callback=dosomething"></script>
```

首先第一个 script 便签定义了一个处理数据的函数;

然后第二个 script 标签载入一个 js 文件，http://example.com/data.php 是数据所在地址，但是因为是当做 js 来引入的，所以 http://example.com/data.php 返回的必须是一个能执行的 js 文件;

最后 js 文件载入成功后会执行我们在 url 参数中指定的函数，并且会把我们需要的 json 数据作为参数传入。所以 php 应该是这样的：

```
<?php
$callback = $_GET['callback'];//得到回调函数名
$data = array('a','b','c');//要返回的数据
echo $callback.'('.json_encode($data).')';//输出
?>
```

JSONP 的优缺点

优点:

它的兼容性更好，在更加古老的浏览器中都可以运行，不需要 XMLHttpRequest 或 ActiveX 的支持;

能够直接访问响应文本，支持在浏览器与服务器之间双向通信

缺点:

JSONP 是从其他域中加载代码执行。如果其他域不安全，很可能会在响应中夹带一些恶意代码，而此时除了完全放弃 JSONP 调用之外，没有办法追究。因此在使用不是你自己运维的 Web 服务时，一定得保证它安全可靠。

它只支持 GET 请求而不支持 POST 等其它类型的 HTTP 请求；它只支持跨域 HTTP 请求这种情况，不能解决不同域的两个页面之间如何进行 JavaScript 调用的问题

3. document.domain + iframe

## 11、请说说事件委托机制？这样做有什么好处？

答:

（1）事件委托，就是某个事件本来该自己干的，但是自己不干，交给别人来干。就叫事件委托。打个比方：一个 button 对象，本来自己需要监控自身的点击事件，但是自己不来监控这个点击事件，让自己的父节点来监控自己的点击事件。

（2）好处：

A,提高性能：列如，当有很多 li 同时需要注册事件的时候，如果使用传统方法来注册事件的话，需要给每一个 li 注册事件。然而如果使用委托事件的话，就只需要将事件委托给该一个元素即可。这样就能提高性能。

B,新添加的元素还会有之前的事件；

# 其他问题

## 1、谈谈你对模块化的理解？

答：

- 模块化就是为了减少系统耦合度，提高高内聚，减少资源循环依赖，增强系统框架设计。
- 让开发者便于维护,同时也让逻辑相同的部分可复用
- 模块化开发：针对 js、css，以功能或业务为单元组织代码。js 方面解决独立作用域、依赖管理、

api 暴露、按需加载与执行、安全合并等问题，css 方面解决依赖管理、组件内部样式管理等问题。

任何事物都有一个过程，那么模块化的过程通俗点讲就是：

模块化的过程就是：

- 1、拆分

将整个系统按功能,格式,加载顺序,继承关系分割为一个一个单独的部分.

注意:拆分的粒度问题,可复用问题,效率问题.如何这些问题处理的不好，就有可能出现不想要的后果。

将功能或特征相似的部分组合在一起,组成一个资源块.

将每个资源块按找需求,功能场景以及目录约束放到固定的地方以供调用.

模块的历程

模块化的发展也是从草根一步一步走过来的。从最开始到现在成熟方案：

1. namespace
2. sass,less
3. AMD&CMD
4. html 模版
5. grunt,gulp,webpack
6. FIS,YUI,KISSY

## 2、前端性能优化有哪些？

答：文件合并

文件最小化/文件压缩

使用 CDN 托管

缓存的使用

## 3、常用设计模式的理解？

答：
创建型
1. Factory Method（工厂方法）
2. Abstract Factory（抽象工厂）
3. Builder（建造者）
4. Prototype（原型）
5. Singleton（单例）

结构型
6. Adapter Class/Object（适配器）
7. Bridge（桥接）
8. Composite（组合）
9. Decorator（装饰）
10. Facade（外观）
11. Flyweight（享元）
12. Proxy（代理）

行为型
13. Interpreter（解释器）
14. Template Method（模板方法）
15. Chain of Responsibility（责任链）
16. Command（命令）
17. Iterator（迭代器）
18. Mediator（中介者）
19. Memento（备忘录）
20. Observer（观察者）
21. State（状态）
22. Strategy（策略）
23. Visitor（访问者）

## 4、常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

答：使用率较高的框架有 jQuery、YUI、Prototype、Dojo、Ext.js、Mootools 等。尤其是 jQuery，超过 91%。

轻量级框架有 Modernizr、underscore.js、backbone.js、Raphael.js 等。（理解这些框架的功能、性能、设计原理）

前端开发工具：Sublime Text 、Eclipse、Notepad、Hbuilder、Webstrom、Firebug、HttpWatch、Yslow。

开发过的插件：城市选择插件，汽车型号选择插件、幻灯片插件。弹出层。（写过开源程序，加载器，js 引擎更好）

## 5、常见的浏览器内核有哪些？

答：Ie(Ie 内核) 火狐（Gecko） 谷歌（webkit） opear(Presto)

## 6、js 内存泄漏了解吗？

答：IE6 时代有 bug，闭包会造成内存泄漏，这个现在已经无须考虑了。
其次，闭包本身不会造成内存泄漏，但闭包过多很容易导致内存泄漏。
这句话很矛盾，技术上讲，闭包是不会造成内存泄漏的，浏览器的 bug 除外。但是，闭包会造成对象引用的生命周期脱离当前函数的上下文，因此，如果不仔细考虑闭包函数的生命周期，的确有可能出现意料之外的内存泄漏，当然，从严格意义上讲，这是程序员自己的 bug，而不是闭包的错。

## 7、能否简述一下如何使一套设计方案，适应不同的分辨率，有哪些方法可以实现？

答：流式布局：

使用非固定像素来定义网页内容，也就是百分比布局，通过盒子的宽度设置成百分比来根据屏幕的宽度来进行伸缩，不受固定像素的限制，内容向两侧填充。

这样的布局方式，就是移动 web 开发使用的常用布局方式。这样的布局可以适配移动端不同的分辨率设备。

响应式开发：

那么 Ethan Marcotte 在 2010 年 5 月份提出的一个概念，简而言之，就是一个网站能够兼容多个终端。越来越多的设计师也采用了这种设计。

CSS3 中的 Media Query（媒介查询），通过查询 screen 的宽度来指定某个宽度区间的网页布局。

- 超小屏幕（移动设备） 768px 以下
- 小屏设备 768px-992px
- 中等屏幕 992px-1200px
- 宽屏设备 1200px 以上

由于响应式开发显得繁琐些，一般使用第三方响应式框架来完成，比如 bootstrap 来完成一部分工作，当然也可以自己写响应式。

阐述下移动 web 和响应式的区别：

| 开发方式 | 移动web开发+PC开发 | 响应式开发 |
| --- | --- | --- |
| 应用场景 | 一般在已经有PC端的网站，开发移动站的时候，只需单独开发移动端。 | 针对新建站的一些网站，现在要求适配移动端，所以就一套页面兼容各种终端，灵活。 |
| 开发 | 针对性强，开发效率高。 | 兼容各种终端，效率低， |
| 适配 | 只适配 移动设备，pad上体验相对较差。 | 可以适配各种终端 |
| 效率 | 代码简洁，加载快。 | 代码相对复杂，加载慢。 |

8、Bootstrap 中最多可以分多少列？lg、md、sm、xs 这几个屏幕宽度的界限是多少？

答：12 列
.col-xs- 超小屏幕手机 (<768px)
.col-sm- 小屏幕平板 (≥768px)
.col-md- 中等屏幕桌面显示器 (≥992px)
.col-lg- 大屏幕大桌面显示器 (≥1200px)

9、为什么 angular 不推荐使用 dom 操作？

答：Angular 倡导以测试驱动开发，在的 service 或者 controller 中出现了 DOM 操作，那么也就意味着的测试是无法通过的

使用 Angular 的其中一个好处是啥，那就是双向数据绑定，这样就能专注于处理业务逻辑，无需关系一堆堆的 DOM 操作。如果在 Angular 的代码中还到处充斥着各种 DOM 操作，那为什么不直接使用 jquery 去开发呢。

## 10、谈谈你对 node. js 的认识？

答：Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。它使我们能够在本地运行 javascript。

## 11、服务器 Node. js 和浏览器 js 的区别是什么？Node. js 把 js 从客户端迁移了到服务端、主要做了哪些工作？为什么说 Node. js 适合做高并发的互联网应用？

答：Node 采用一系列“非阻塞”库来支持事件循环的方式。本质上就是为文件系统、数据库之类的资源提供接口。Node.js 使用事件驱动，非阻塞 I/O 模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用。