面试题总结

- 此版本用于助教老师（助教老师根据实际情况考虑给不给学员答案）
- 使用于带班期间晚自习或者强制仔细学生演讲时使用
- 总结有常用面试基础题和学员们从面试经历中带回来的
- **标注星星的为末尾学员必须掌握的，达到讲过以后随时随地问都可以流利回答**

一、基础班模块

基础部分

1. 什么是HTML?

   答：

   > HTML并不是真正的的程序语言，他是一种 标 记 语 言 ，用来结构化和含义化你想要放 在web 网站上的那些内容。它由一系列的元素（elements）所组成，这些元素可以用来 封装你的内容中担任不同工作的各部分和各个角色。

2. 什么是CSS?

   答：

   > 就像 HTML，CSS 也不是真正的编程语言。它是样式表语言，也就是说，它允许你有 选择性的为 HTML 文档的元素添加样式。

3. 行内元素和块级元素的具体区别是什么？行内元素的padding和margin可设置吗?

   答：

   > 块级元素(block)特性：
   >
   > 总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示;
   >
   > 宽度(width)、高度(height)、内边距(padding)和外边距(margin)都可控制;
   >
   > 内联元素(inline)特性：
   >
   > 和相邻的内联元素在同一行;
   >
   > 宽度(width)、高度(height)、内边距的top/bottom(padding-top/padding-bottom)和外边距的top/bottom(margin-top/margin-bottom)都不可改变（也就是padding和margin的left和right是可以设置的），就是里面文字或图片的大小。

4. 简述一下你对HTML语义化的理解?

答:

> 1. HTML语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析；
> 2. 即使在没有样式CSS 的情况下也能以一种文档格式显示，并且是容易阅读的；
> 3. 搜索引擎的爬虫也依赖于HTML标记来确定上下文和各个关键字的权重，有利于SEO；
> 4. 使阅读源代码的人更容易将网站分块，便于阅读、维护和理解；

5. rgba() 和 opacity 设置透明度的区别是什么?

答:

> rgba()和opacity都能实现透明效果，但最大的不同是opacity作用于元素，以及元素内的所有内容的透明度，而rgba()只作用于元素的颜色或其背景色。（设置rgba透明的元素的子元素不会继承透明效果！）

6. DOCTYPE的作用?

答:

> 1. <!DOCTYPE> 声明位于文档中的最前面，处于 标签之前。告知浏览器以
>
> 何种模式来渲染文档。
>
> 2. 严格模式的排版和 JS 运作模式是以该浏览器支持的最高标准运行。
> 3. 在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站
>
> 点无法工作。
>
> 4. DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。

7. 介绍一下你对浏览器内核的理解？都有哪些常见的浏览器内核?

答:

> 要或者说核心的部分是“Rendering Engine”，可大概译为“渲染引擎”，不过我们 一般习惯将之称为“浏览器内核”。负责对网页语法的解释（如标准通用标记语言下的一个 应用 HTML、JavaScript）并渲染（显示）网页。 所以，通常所谓的浏览器内核也就是浏 览器所采用的渲染引擎，渲染引擎决定了浏览器如何显示网页的内容以及页面的格式信息。 不同的浏览器内核对网页编写语法的解释也有不同，因此同一网页在不同的内核的浏览器里 的渲染（显示）效果也可能不同，这也是网页编写者需要在不同内核的浏览器中测试网页显 示效果的原因。 常见浏览器内核： Trident 内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称 MSHTML] Gecko内核：

> Netscape6 及以上版本，FF,MozillaSuite/SeaMonkey 等。 Presto 内核：Opera7及以上。 [Opera内核原为：Presto，现为、Blink;] Webkit 内核：Safari,Chrome 等。 [Chrome 的：Blink（WebKit 的分支）] EdgeHTML内核：Microsoft Edge。 [此内核其实是从 MSHTML fork 而来，删掉了几 乎所有的 IE私有特性

8. CSS选择器权重如何计算?

   答:

   - [ ]

   > 页面显示样式的优先级取决于其“特殊性”，特殊性越高，就显示最高的，当特殊性相等 时，显示后者 特殊性表述为 4个部分：0,0,0,0 一个选择器的特殊性如下:
   >
   > 对于选择器是#id的属性值,特殊性值为：0,1,0,0 对于属性选择器，class或伪类，特殊性值为：0,0,1,0 对于标签选择器或伪元素，特殊性值为：0,0,0,1 通配符‘*’特殊性值为：0,0,0,0 内联样式特殊性值为：1,0,0,0

9. 对WEB标准以及W3C的理解与认识?

   答:

   - [ ]

   > 1. 标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外链 css 和 js 脚
   >   本、结构行为表现的分离，
   > 2. 文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所
   >   访问、更少的代码和组件，
   > 3. 容易维护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提
   >   高网站易用性。

10. CSS中优雅降级和渐进增强有什么区别?

    答:

    - [ ]

    > 优雅降级和渐进增强印象中是随着css3 流出来的一个概念。由于低级浏览器不支持css3，但 css3 的效果又太优秀不忍放弃，所以在高级浏览中使用 css3 而低级浏览器只保证 最基本的功能。咋一看两个概念差不多，都是在关注不同浏览器下的不同体验，关键的区别 是他们所侧重的内容，以及这种不同造成的工作流程的差异。
    >
    > “优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。 “渐进增强”观点则认为应关注于内容本身。

11. 对BFC规范的理解有哪些?

    答:

1. 定义：

BFC(Block formatting context)直译为"块级格式化上下文"。它是一个独立的渲染区域， 只有 Block-level box 参与，它规定了内部的Block-level Box 如何布局，并且与这个区域 外部毫不相干。 布局规则：

A. 内部的Box 会在垂直方向，一个接一个地放置。 B. Box垂直方向的距离由margin决定。属于同一个 BFC的两个相邻 Box的 margin 会发 生重叠。 C. 每个元素的 margin box 的左边，与包含块border box 的左边相接触(对于从左往右的 格式化，否则相反)。即使存在浮动也是如此。 D. BFC 的区域不会与 float box 重叠。 E. BFC 就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反 之也如此。 F. 计算 BFC 的高度时，浮动元素也参与计算。

2. 哪些元素会生成 BFC：

A. 根元素 B. float 属性不为 none C. position为 absolute 或 fixed D. display为 inline-block， table-cell， table-caption， flex， inline-flex F. overflow 不为 visible

12. 有多少种清除浮动的方法?

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专有属性) 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；}
```

实际工作部分

1. HTML常见兼容性问题?

   答:

   1. 双边距BUG float引起的 使用display
   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. Iez-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:0.08line-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

2. 描述一个"reset"的CSS文件并如何使用它。知道`normalize.css`吗？你了解他们的不同之处?

   答:

   重置样式非常多，凡是一个前端开发人员肯定有一个常用的重置CSS文件并知道如何使用它们。他们是盲目的在做还是知道为什么这么做呢？原因是不同的浏览器对一些元素有不同的默认样式，如果你不处理，在不同的浏览器下会存在必要的风险，或者更有戏剧性的性发生。

   你可能会用Normalize来代替你的重置样式文件。它没有重置所有的样式风格，但仅提供了一套合理的默认样式值。既能让众多浏览器达到一致和合理，但又不扰乱其他的东西（如粗体的标题）。

   在这一方面，无法做每一个复位重置。它也确实有些超过一个重置，它处理了你永远都不用考虑的怪癖，像HTML的audio元素不一致或line-height不一致。

3. BFC是什么?

答：

- [ ] BFC（块级格式化上下文），一个创建了新的BFC的盒子是独立布局的，盒子内元素的布局不会影响盒子外面的元素。在同一个BFC中的两个相邻的盒子在垂直方向发生margin重叠的问题

  BFC是指浏览器中创建了一个独立的渲染区域，该区域内所有元素的布局不会影响到区域外元素的布局，这个渲染区域只对块级元素起作用

4. 怎样实现三栏布局，两边宽度固定，中间自适应?

答：

- [ ] 圣杯布局 双飞翼布局

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title></title>
    <style type="text/css">
        * {
            margin: 0;
            padding: 0;
        }

        #left {
            width: 200px;
            height: 200px;
            float: left;
            background-color: red;
        }

        #right {
            width: 150px;
            height: 200px;
            float: right;
            background-color: mistyrose;
        }

        #middle {
            height: 200px;
            margin: 0 150px 0 200px;
            background-color: saddlebrown;
            word-break: break-word;
```

```
            }
        </style>
    </head>
    <body>
    <div id="content">
        <div id="left">我是左侧内容我是左侧内容我是左侧内容我是左侧
    内容我是左侧内容</div>
        <div id="right">我是右侧内容我是右侧内容我是右侧内容我是右侧
    内容我是右侧内容我是右侧内容</div>
        <div id="middle">我是中间内容我是中间内容我是中间内容我是中
    间内容我是中间内容我是中间内容我是中间内容我是中间内容我是中间内容我
    是中间内容</div>
    </div>
    </body>
    </html>
```

5. 精灵图(CSS Sprites)的优点和缺点

   答：

精灵图是一种网页图片应用处理方式。就是把网页中很多小背景图片整合到一张图片文件中，再利用CSS的“background-image”，“background-repeat”，“background-position”的组合进行背景图显示及定位，达到显示某一部分背景图的效果。

精灵图的优点：

1. 减少图片的体积，因为每个图片都有一个头部信息，把多个图片放到一个图片里，就会共用同一个头部信息，从而减少了字节数。
2. 减少了网页的http请求次数，从而加快了网页加载速度，提高用户体验。
3. 解决了网页设计师在图片命名上的困扰，只需对一张集合的图片上命名就可以了，不需要对每一个小元素进行命名，从而提高了网页的制作效率。
4. 更换风格方便，只需要在一张或少张图片上修改图片的颜色或样式，整个网页的风格就可以改变。维护起来更加方便。

精灵图的缺点：

1. 在图片合并的时候，你要把多张图片有序的合理的合并成一张图片，还要留好足够的空间，防止板块内出现不必要的背景；这些还好，最痛苦的是在宽屏，高分辨率的屏幕下的自适应页面，你的图片如果不够宽，很容易出现背景断裂；
2. 在开发的时候比较麻烦，你要通过photoshop或其他工具测量计算每一个背景单元的精确位置，这是针线活，没什么难度，但是很繁琐；
3. 在维护的时候比较麻烦，如果页面背景有少许改动，一般就要改这张合并的图片，无需改的地方最好不要动，这样避免改动更多的css，如果在原来的地方放不下，又只能（最好）往下加图片，这样图片的字节就增加了，还要改动css。
4. 精灵图不能随意改变大小和颜色。精灵图改变大小会失真模糊，降低用户体验，css3新属性可以改变精灵图颜色，但是比较麻烦，并且新属性有兼容问题。现在

> 一般都是用web字体(图标字体)来代替精灵图。

二、JavaScript基础模块

基础部分

1. JS中有哪些数据类型?

   答:

   - [ ] 简单数据类型：Undefined、Null、Boolean、Number 和String。 复杂数据类型：Object

2. "==" 和 "===" 的区别?

   答:

   - [ ] 前者会自动转换类型,而后者不会。

     前者比较的是值，后者比较的是值和类型。

3. JS中的常用内置对象有哪些？并列举该对象的常用方法?

   答:

   - [ ] 1. Arguments 函数参数集合

     arguments[ ] 函数参数的数组

     Arguments 一个函数的参数和其他属性

     Arguments.callee 当前正在运行的函数

     Arguments.length 传递给函数的参数的个数

     2. Array 数组

     length属性 动态获取数组长度

     join() 将一个数组转成字符串。返回一个字符串。

     reverse() 将数组中各元素颠倒顺序

     delete运算符 只能删除数组元素的值，而所占空间还在，总长度没变(arr.length)。

     shift() 删除数组中第一个元素，返回删除的那个值，并将长度减 1。

     pop() 删除数组中最后一个元素，返回删除的那个值，并将长度减1。

     unshift() 往数组前面添加一个或多个数组元素，长度要改变。

     push() 往数组结尾添加一个或多个数组元素，长度要改变。

     concat( ) 连接数组

slice( ) 返回数组的一部分

sort( ) 对数组元素进行排序

splice( ) 插入、删除或替换数组的元素

toLocaleString( ) 把数组转换成局部字符串

toString( ) 将数组转换成一个字符串

3. Boolean 布尔对象

   Boolean.toString( ) 将布尔值转换成字符串

   Boolean.valueOf( ) Boolean对象的布尔值

4. Error 异常对象

   Error.message 可以读取的错误消息

   Error.name 错误的类型

   Error.toString( ) 把Error 对象转换成字符串

   EvalError 在不正确使用 eval()时抛出

   SyntaxError 抛出该错误用来通知语法错误

   RangeError 在数字超出合法范围时抛出

   ReferenceError 在读取不存在的变量时抛出

   TypeError 当一个值的类型错误时，抛出该异常

   URIError 由URI的编码和解码方法抛出

5. Function 函数构造器

   Function 函数构造器

   Function.apply( ) 将函数作为一个对象的方法调用

   Function.arguments[] 传递给函数的参数

   Function.call( ) 将函数作为对象的方法调用

   Function.caller 调用当前函数的函数

   Function.length 已声明的参数的个数

   Function.prototype 对象类的原型

   Function.toString( ) 把函数转换成字符串

6. Math 数学对象

   Math对象是一个静态对象

   Math.PI 圆周率。

   Math.abs() 绝对值。

   Math.ceil() 向上取整(整数加 1，小数去掉)。

   Math.floor() 向下取整(直接去掉小数)。

   Math.round() 四舍五入。

Math.pow(x，y) 求 x的y次方。

Math.sqrt() 求平方根。

7. Number 数值对象

   Number.MAX_VALUE 最大数值

   Number.MIN_VALUE 最小数值

   Number.NaN 特殊的非数字值

   Number.NEGATIVE_INFINITY 负无穷大

   Number.POSITIVE_INFINITY 正无穷大

   Number.toExponential( ) 用指数计数法格式化数字

   Number.toFixed( ) 采用定点计数法格式化数字

   Number.toLocaleString( ) 把数字转换成本地格式的字符串

   Number.toPrecision( ) 格式化数字的有效位

   Number.toString( ) 将一个数字转换成字符串

   Number.valueOf( ) 返回原始数值

8. Object 基础对象

   Object 含有所有 JavaScript 对象的特性的超类

   Object.constructor 对象的构造函数

   Object.hasOwnProperty( ) 检查属性是否被继承

   Object.isPrototypeOf( ) 一个对象是否是另一个对象的原型

   Object.propertyIsEnumerable( ) 是否可以通过 for/in循环看到属性

   Object.toLocaleString( ) 返回对象的本地字符串表示

   Object.toString( ) 定义一个对象的字符串表示

   Object.valueOf( ) 指定对象的原始值

9. RegExp 正则表达式对象

   RegExp.exec( ) 通用的匹配模式

   RegExp.global 正则表达式是否全局匹配

   RegExp.ignoreCase 正则表达式是否区分大小写

   RegExp.lastIndex 下次匹配的起始位置

   RegExp.source 正则表达式的文本

   RegExp.test( ) 检测一个字符串是否匹配某个模式

   RegExp.toString( ) 把正则表达式转换成字符串

10. String 字符串对象

    Length 获取字符串的长度。

    toLowerCase() 将字符串中的字母转成全小写。

> toUpperCase() 将字符串中的字母转成全大写。
>
> charAt(index) 返回指定下标位置的一个字符。如果没有找到，则返回空字符串。
>
> substr() 在原始字符串，返回一个子字符串
>
> substring() 在原始字符串，返回一个子字符串。
>
> split() 将一个字符串转成数组。
>
> charCodeAt( ) 返回字符串中的第 n个字符的代码
>
> concat( ) 连接字符串
>
> fromCharCode( ) 从字符编码创建一个字符串
>
> indexOf( ) 返回一个子字符串在原始字符串中的索引值(查找顺序从左往右查找)。如果没 有找到，则返回-1。
>
> lastIndexOf( ) 从后向前检索一个字符串
>
> localeCompare( ) 用本地特定的顺序来比较两个字符串
>
> match( ) 找到一个或多个正则表达式的匹配
>
> replace( ) 替换一个与正则表达式匹配的子串
>
> search( ) 检索与正则表达式相匹配的子串
>
> slice( ) 抽取一个子串
>
> toLocaleLowerCase( ) 把字符串转换小写
>
> toLocaleUpperCase( ) 将字符串转换成大写
>
> toLowerCase( ) 将字符串转换成小写
>
> toString( ) 返回字符串
>
> toUpperCase( ) 将字符串转换成大写
>
> valueOf( ) 返回字符串

4. 什么是闭包?

   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。

5. 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null，这样也就形成了原型链。

6. 有哪些方式继承?

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。
> 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。
> 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。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。

7. 字符创的常用方法有哪些?

答：

> charCodeAt 方法返回一个整数，代表指定位置字符的 Unicode 编码；
>
> charAt方法返回指定索引位置处的字符。如果超出有效范围的索引值返回空字符串；
>
> slice方法返回字符串的片段；

> substring方法返回位于String 对象中指定位置的子字符串。
>
> substr方法返回一个从指定位置开始的指定长度的子字符串。
>
> indexOf方法返回 String 对象内第一次出现子字符串位置。如果没有找到子字符串，则返回-1；
>
> lastIndexOf方法返回 String对象中字符串最后出现的位置。如果没有匹配到子字符串，则返回-1；
>
> search方法返回与正则表达式查找内容匹配的第一个字符串的位置。
>
> concat 方法返回字符串值，该值包含了两个或多个提供的字符串的连接；
>
> split 将一个字符串分割为子字符串，然后将结果作为字符串数组返回；

8. DOM节点的增删改查?

   答：

> 1. 创建节点、追加节点
>
>   createElement（标签名）创建一个元素节点（具体的一个元素）。
>
>   createTextNode（节点文本内容）创建一个文本节点。
>
>   createDocumentFragment() //创建一个 DOM 片段。
>
>   appendChild（节点）追加一个节点。
> 2. 插入节点
>
>   appendChild（节点）也是一种插入节点的方式，还可以添加已经存在的元素，会将其 元素从原来的位置移到新的位置。
>
>   insertBefore（a,b）是参照节点，意思是 a节点会插入 b节点的前面。
> 3. 删除、移除节点
>
>   removeChild(节点) 删除一个节点，用于移除删除一个参数（节点）。其返回的被移除 的节点，被移除的节点仍在文档中，只是文档中已没有其位置了。
> 4. 复制节点
>
>   cloneNode() 方法，用于复制节点， 接受一个布尔值参数， true 表示深复制（复制节点 及其所有子节点）， false 表示浅复制（复制节点本身，不复制子节点）。
> 5. 替换节点
>
>   replaceChild(插入的节点，被替换的节点)，用于替换节点，接受两个参数，第一参数 是要插入的节点，第二个是要被替换的节点。返回的是被替换的节点。
> 6. 查找节点
>
>   getElementsByTagName() //通过标签名称 getElementsByName() //通过元素的 Name 属性的值(IE容错能力较强，会得到一 个数组，其中包括 id等于 name值的) getElementById() //通过元素 Id，唯一性

9. 什么是预解析?

答:

> 在代码整体执行之前，先解析一部分。
>
> 预解析之后，代码才会从上往下依次整体执行，但是预解析执行过的代码不会 重复执行。
>
> js预解析干了什么事：js 中预解析会把声明部分的代码预先执行。
>
> 声明相关的代码可以分为两部分：
>
> 1、 变量声明 通过 var关键字定义的变量。
>
> 2、函数声明 通过 function关键字声明的函数
>
> 预解析时如果遇到重复的变量声明，那么忽略。
>
> 预解析时如果遇到重复的函数声明，保留后面的函数。
>
> 预解析时如果遇到变量与函数重名的情况，保留函数。

10. 什么是变量名提升?

答:

> 使用 var关键字定义的变量，被称为变量声明；
>
> 函数声明提升的特点是，在函数声明的前面，可以调用这个函数。

11. JS中的typeof关键字能返回哪些数据类型?

答:

> typeof一般判断基本数据类型。是一个操作符而不是函数，圆括号可有可无。
>
> typeof 返回值有：string，number，boolean，undefined，object ，function，
>
> 基本数据类型：Boolean、Number、String、Undefined、Null
>
> 基本数据类型中数字，字符串，布尔类型返回其对类型 undefined返回 undefined
>
> 九大内置构造函数及其他所有函数返回function；
>
> 其他所有复杂类型对象和null返回 object

12. 简述创建函数的几种方式?

答:

> 1. 函数声明
>
>   function sum1(num1,num2){

```
    return num1+num2;
   }
```

2. 函数表达式

```
   var sum2 = function(num1,num2){
    return num1+num2;
   }
```

3. 函数对象方式

```
   var sum3 = new Function("num1","num2","return num1+num2");
```

13. 代码实现数组排序并去重

答:

```
function fn(arr){
  for(var i = 0; i < arr.length-1; i++){
    for(var j = 0; j < arr.length-1-i; j++){
      if(arr[j]<arr[j+1]){
        var temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
      }
    }
  }
  for(var k = 0; k < arr.length; k++){
    var c = arr[k];
    for(var l = k+1; l < arr.length; l++){
      if(arr[l] == c){
        arr.splice(l, 1);
        l--;
      }
    }
  }
  return arr
}
var arr = [1, 2, 5, 6, 8, 9, 10, 6, 5, 7, 4, 3, 5]
console.log(fn(arr))
```

14.写出下面代码输出的结果

A. console.log(undefined || 1); --> 1

B. console.log(null || NaN); --> NaN

C. console.log(0 && 1); --> 0

D. console.log(0 && 1 || 0); --> 0

15. 下列代码将会输出什么?

```
var foo = 1;
function fn() {
  console.log( foo );   -->  undefined
  var foo = 2;
  console.log( foo );   -->  2
}
fn();
```

实际工作部分

1. 什么是短路表达式?

   答:

   - [ ] 短路表达式只是一种简写形式，也就是用 && 和 || 来赋值或者执行函数的形式

     例如:

     var foo = foo1 || foo2;

     意思是如果foo1是真的，那么就把foo1的值赋给foo，否则把foo2的值赋给foo。

     foo && foo()

     当foo存在的时候，我们就执行foo函数，如果这个时候foo不是一个函数，就会报错，所以这个只是一种简写形式而已。

2. 控制台中使用哪些部分调试?

   答:

   - [ ] 主要用console来进行调试

     1. console.log 用于输出普通信息
     2. console.info 用于输出提示性信息
     3. console.error用于输出错误信息
     4. console.warn用于输出警示信息
     5. console.debug用于输出调试信息

3.

三、Web API模块

基础部分

1. 要你出一套适应不同分辨率，不同终端的前端实现方案你有什么思路?

   答:

   1. 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。

   2. 响应式开发:

   那么 EthanMarcotte 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。

2. px em rem 取用选择依据?

   答:

   1. px 像素（Pixel）。绝对单位。像素px是相对于显示器屏幕分辨率而言的，是一个虚拟长度单位，是计算机系统的数字化图像长度单位，如果 px要换算成物理长度，需要 指定精度 DPI。
   2. em是相对长度单位，相对于当前对象内文本的字体尺寸。如当前对行内文本的字

   体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。它会继承父级元素的字体大小，因 此并不是一个固定的值。

   3. rem是 CSS3新增的一个相对单位（root em，根em），使用 rem 为元素设定字体大小时，仍然是相对大小，但相对的只是HTML根元素。
   4. 区别：IE无法调整那些使用 px作为单位的字体大小，而em 和rem可以缩放，rem

   相对的只是HTML根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既 可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁 反应。目前，除了 IE8及更早版本外，所有浏览器均已支持rem。

3. Zepto和jQuery的区别?

   答:

> Zepto相对jQuery更加轻量，主要用在移动端，jQuery也有对应的jQuerymobile移动端框架。

实际工作部分

1. 移动端touch事件判断滑屏手势的方向?

   答:

> 当开始一个touchstart事件的时候，获取此刻手指的横坐标startX和纵坐标startY；当触发touchmove事件时，在获取此时手指的横坐标moveEndX和纵坐标moveEndY;最后，通过这两次获取的坐标差值来判断手指在手机屏幕上的滑动方向。 思路：用touchmove的最后坐标减去touchstart的起始坐标，X的结果如果正数，则说明手指是从左往右划动；X的结果如果负数，则说明手指是从右往左划动；Y的结果如果正数，则说明手指是从上往下划动；Y的结果如果负数，则说明手指是从下往上划动。
>
> 具体代码如下:

```
var mybody = document.getElementsByTagName('body')[0];

//滑动处理
var startX, startY, moveEndX, moveEndY, X, Y;
mybody.addEventListener('touchstart', function(e) {
    e.preventDefault();
    startX = e.touches[0].pageX;
    startY = e.touches[0].pageY;
});
mybody.addEventListener('touchmove', function(e) {
    e.preventDefault();
    moveEndX = e.changedTouches[0].pageX;
    moveEndY = e.changedTouches[0].pageY;
    X = moveEndX - startX;
    Y = moveEndY - startY;
    if ( X > 0 ) {alert(‘向右’);}
    else if ( X < 0 ) {alert(‘向左’);}
    else if ( Y > 0) {alert(‘向下’);}
    else if ( Y < 0 ) { alert(‘向上’);}
    else{alert(‘没滑动’); }
});
```

2. 移动端对图片优化有哪些方式，怎么实现?

   答:

> 懒加载，使用CSS Sprites合并为一张大图，首先从图片格式方面着手，webp(google官方网址)是谷歌推出的一种图片格式，优点在于同等画面质量下，体积比jpg、png少了25%以上，去掉无意义的修饰，使用矢量图替代位图。 按照HTTP协议设置合理的缓存。 详见链接 http://web.jobbole.com/81766/

3. rem布局中的尺寸是怎样计算的，实际举例说明一下?

   答:

> 其实rem布局的本质是等比缩放，一般是基于宽度，试想一下如果UE图能够等比缩放，假设我们将屏幕宽度平均分成100份，每一份的宽度用x表示，x = 屏幕宽度 / 100，如果将x作为单位，x前面的数值就代表屏幕宽度的百分比。

4.

四、JavaScript高级模块

基础部分

1. 说说你对this关键字的理解?

   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

2. 表单验证传输的什么数据？明文还是暗文==加密？如何加密？是每一次传 输数据，都是加密之后才传输吗?

   答:

> 1. 概述： GET是从服务器上请求数据，POST 是发送数据到服务器。事实上，GET 方法是把数 据参数队列（query string）加到一个URL上，值和表单是一一对应的。比如说， name=John。在队列里，值和表单用一个&符号分开，空格用+号替换，特殊的符号转换 成十六进制的代码。因为这一队列在 URL里边，这样队列的参数就能看得到，可以被记录 下来，或更改。通常GET 方法还限制字符的大小（大概是 256 字节 ） 。事实上POST 方法可以没有时间限制的传递数据到服务器，用户在浏览器端是看不到 这一过程的，所以 POST方法比较适合用于发送一个保密的（比如信用 卡号）或者比较大 量的数据到服务器。

2. 区别： Post 是允许传输大量数据的方法，而 Get 方法会将所要传输的数据附在网址后面，然 后一起送达服务器，因此传送的数据量就会受到限制，但是执行效率却比 Post 方法好。
3. 总结：

   1、get 方式的安全性较Post 方式要差些，包含机密信息的话，建议用 Post 数据提交 方式；

   2、在做数据查询时，建议用 Get 方式；而在做数据添加、修改或删除时，建议用Post 方式；
4. 所以： 表达如果是向服务器传输数据(如帐号密码等)都是加密数据(post)，如果只是单单想要 从服务器获得数据或者传输的数据并不重要， 可以直接使用明文方式传输(get)

3. 如何实现跨域?

答：

- [ ] 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，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。

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

4. 说说事件委托机制？这样做有什么好处?

答：

- [ ] 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。

优点：

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。

> 新添加的元素还会有之前的事件；

5. call和apply的区别?

   答：

> 它们的共同之处：都“可以用来代替另一个对象调用一个方法，将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。”
>
> 它们的不同之处：
>
> Apply：最多只能有两个参数——新this 对象和一个数组 argArray。如果给该方法传递多 个参数，则把参数都写进这个数组里面，当然，即使只有一个参数，也要写进数组里面。如 果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被 用作 thisObj， 并且无法被传递任何参数
>
> Call：则是直接的参数列表，主要用在js 对象各方法互相调用的时候，使当前 this 实例指 针保持一致，或在特殊情况下需要改变this指针。如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

6. 在JS的计时器运行原理是怎样的，为什么可以触发计时效果? 计时器是多线程吗?

   答：

> 1. javascript引擎只有一个线程，强迫异步事件排队等待被执行。
> 2. setTimeout 和setInterval 本质上不同的地方是他们如何执行异步代码的。
> 3. 如果一个定时器正在执行的时候被阻塞了，那么它将会被推迟到下一个可能的执行点，这既是使得延迟时间有可能会超过声明定时器时设置的值。
> 4. Interval 如果有足够的时间来执行（大于制定的延迟），那么它将会无延迟的一个紧 接着一个执行。
> 5. 原理： 计时器通过设定一定的时间段（毫秒）来异步的执行一段代码。因为 Javascript 是一 个单线程语言，计时器提供了一种绕过这种语言限制来执行代码的能力。
> 6. 总结： 计时器是单线程的， 需要等待上一个执行完， 如果上一个没有执行完，下一个需要 延迟执行， 直到上一个执行完。

7. 什么是事件的冒泡和捕获?

   答：

> 事件冒泡：子元素事件的触发会影响父元素事件；
>
> 开关事件冒泡：
>
> A，开启事件冒泡：element.addEventListener(eventName，handler，false)；

> B，关闭事件冒泡：假设传统方式事件的返回值为 e，就可以通过 e.stopPropagation()来关闭事件冒泡；
>
> 事件捕获：父元素的事件会影响子元素的事件；
>
> 开启事件捕获：element.addEventListener(eventName，handler，true)

8. 面向对象和类的区别?

   答：

- [ ]

> 简单的说类是对象的模版。 在 js 中没有类， 所以在js 中所谓的 类 就是构造函数，对象就是由构造函数创建 出来的实例对象。面向对象就是使用面向对象的方式处理问题， 面向对象是对面向过程进 行封装。 面向对象有三大特性 抽象性， 需要通过核心数据和特定环境才能描述对象的具体意义
>
> 封装性， 封装就是将数据和功能组合到一起， 在js 中对象就是键值对的集合， 对象 将属性和方法封装起来， 方法将过程封装起来
>
> 继承性，将别人的属性和方法成为自己的，传统继承基于模板(类)，js 中继承基于构 造函数

实际工作部分

1. JavaScript 中的垃圾回收机制?

   答：

- [ ]

> 在Javascript 中，如果一个对象不再被引用，那么这个对象就会被GC 回收。如 果两个对象互相引用，而不再被第3者所引用，那么这两个互相引用的对象也会被回收。因为函数 a被b引用，b又被 a外的 c引用，这就是为什么 函数 a 执行后不会被回收的原 因。

2. 列出3条以上 FF 和 IE 的脚本兼容问题

   答：

- [ ]

> 1. window.event：
>
>   表示当前的事件对象，IE有这个对象，FF没有，FF通过给事件处理函数传递事件对象
> 2. 获取事件源
>
>   IE用srcElement获取事件源，而FF用target获取事件源
> 3. 添加，去除事件
>
>   IE：element.attachEvent(“onclick”,function) element.detachEvent(“onclick”,function)

FF：element.addEventListener(“click”,function, true)
element.removeEventListener(“click”, function, true)

4. 获取标签的自定义属性

IE：div1.value或div1[“value”]

FF：可用div1.getAttribute(“value”)

3. 用正则表达式，写出由字母开头，其余由数字、字母、下划线组成的6~30的字符串?

答：

```
/^[a-zA-Z]{1}[\w]{5,29}$/
```

4. 下列JavaScript代码执行后，iNum的值是多少?

```
var iNum = 0;
for(var i = 1; i < 10; i++){
 if(i% 5 == 0){
 continue;
 }
 iNum++;
}
```

答：8

5. 程序中捕获异常的方法?

答：

```
window.error
try{}catch(error){}finally{}
```

6. 正则匹配邮箱?

答：

```
/^([a-zA-Z0-9-])+@([a-zA-Z0-9-])+((.[a-zA-Z0-9_-]{2,3}){1,2})$/;
```

7. Javascript中callee和caller的作用?

答：

caller是返回一个对函数的引用，该函数调用了当前函数；

callee是返回正在被执行的function函数，也就是所指定的function对象的正文。

8. 下列代码的输出结果?

```
function f1(){
  var tmp = 1;
  this.x = 3;
  console.log( tmp );
  console.log( this.x );
}
var obj = new f1();
console.log( obj.x );
console.log( f1() );
```

答:

首先看代码（1），这里实例话化了f1这个类。相当于执行了 f1函数。所以**这个时候 A 会输出 1**， 而 B 这个时候的 this 代表的是实例化的当前对象 obj ,**B 输出 3**。

代码（2）毋庸置疑**会输出 3**,

重点 代码（3）首先这里将**不再是一个类**，它**只是一个函数**。那么 **A输出 1**， B呢? 这里的this 代表的其实就是window对象，那么this.x 就是一个全局变量 相当于在外部 的一个全局变量。所以**B 输出 3**。最后代码由于**f没有返回值**那么一个函数如果没返回值的话，将会**返回 underfined** ，所以答案就是： 1， 3， 3， 1， 3， underfined 。

9. 下面代码输出结果?

```
function changeObjectProperty(o){
  o.siteUrl = "http://www.csser.com/";
  o = new Object();
  o.siteUrl = "http://www.popcg.com/";
}
var CSSer = new Object();
changeObjectProperty( CSSer );
console.log( CSSer.siteUrl );
```

答:

`"http://www.popcg.com/"`

10.

五、jQuery模块

基础部分

1. 谈谈你对jQuery的理解?

   答:

   > JQuery 是继 prototype 之后又一个优秀的 Javascript 库。它是轻量级的js 库 ， 它兼容 CSS3，还兼容各种浏览器（IE 6.0+，FF1.5+，Safari 2.0+，Opera 9.0+），jQuery2.0 及后续版本将不再支持 IE6/7/8 浏览器。jQuery 使用户能更方便地处理 HTML（标准通用 标记语言下的一个应用）、events、实现动画效果，并且方便地为网站提供 AJAX 交互。 jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细， 同时还有许多成熟的插件可供选择。jQuery 能够使用户的 html页面保持代码和 html 内容 分离，也就是说，不用再在 html里面插入一堆 js 来调用命令了，只需要定义 id即可。 jQuery 是一个兼容多浏览器的 javascript 库，核心理念是write less，do more(写 得更少，做得更多)。 jQuery是免费、开源的，使用 MIT 许可协议。jQuery 的语法设 计可以使开发更加便捷，例如操作文档对象、选择 DOM 元素、制作动画效果、事件处理、 使用 Ajax 以及其他功能。除此以外，jQuery 提供 API让开发者编写插件。其模块化的使 用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

2. 、原生JS的window.onload与Jquery的$(document).ready(function () {})，$(function () {})有什么不同?

   答:

   > 1.执行时间 window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。 $(document).ready()是 DOM 结构绘制完毕后就执行，不必等到加载完毕。
   >
   > 2.编写个数不同 window.onload不能同时编写多个，如果有多个 window.onload 方法，只会执 行一个 $(document).ready()可以同时编写多个，并且都可以得到执行
   >
   > 3.简化写法 window.onload没有简化写法
   >
   > $(document).ready(function (){})
   >
   > 可以简写成$(function(){})；

3. jQuery一个对象可以同时绑定多个事件，是如何实现的?

   答:

- [ ] jQuery可以给一个对象同时绑定多个事件，低层实现方式是使用addEventListner或attachEvent兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

4. jQuery.fn的init方法返回的this指的是什么对象？为什么要返回this?

   答：

- [ ] this执行init构造函数自身，其实就是jQuery实例对象，返回this是为了实现jQuery的链式操作

5. jQuery.extend和jQuery.fn.extend有什么区别?

   答：

- [ ] Jquery.extend用来扩展jQuery对象本身；jquery.fn.extend用来扩展jQuery实例

实际工作部分

1. jQuery框架中$.ajax()的常用参数有哪些？写一个post请求并带有发送数据和返回数据的样例?

   答：

- [ ] async是否异步
  url请求地址
  contentType发送信息至服务器时内容编码类型
  data发送到服务器的数据
  dataType预期服务器返回的数据类型
  type请求类型
  success请求成功回调函数
  error请求失败回调函数
  $.ajax({
  url: "/jquery/test1.txt",
  type: 'post',
  data: { id: 1 },
  success: function (data) { alert(data); }
  })

2. 举一下jquery中的函数，这些函数实现链式编程的原理?

   答：

```
toggle (fn, fn)

$ (“td”) .toggle (
function () {
$ (this) .addClass (“selected”) ;
},
function () {
$ (this) .removeClass (“selected”) ;
)
```

实现函数链式编程的原理：返回自身，其他过程在函数内部实现，其好处是：节约js代码，返回的是同一个对象，提高代码的效率。

3.

六、PHP后台开发模块

基础部分

1. php inset 和 empty 的区别?

   答：

   1、empty函数

   用途：检测变量是否为空

   判断：如果 var 是非空或非零的值，则 empty() 返回 FALSE。换句话说，""、0、"0"、NULL、FALSE、array()、var $var; 以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 TRUE。注意：empty() 只检测变量，检测任何非变量的东西都将导致解析错误。换句话说，后边的语句将不会起作用;

   2、isset函数

   用途：检测变量是否设置

   判断：检测变量是否设置，并且不是 NULL。如果已经使用 unset() 释放了一个变量之后，它将不再是 isset()。若使用 isset() 测试一个被设置成 NULL 的变量，将返回 FALSE。同时要注意的是一个NULL 字节（"\0"）并不等同于 PHP 的 NULL 常数。

2. php 中$_SERVER变量中是如何得到当前执行脚本路径的?

七、Ajax模块

基础部分

1. ajax是什么?

   答:

> Ajax并不算是一种新的技术，全称是asychronous javascript and xml，可以说是已有技术的组合，主要用来实现客户端与服务器端的异步通信效果，实现页面的局部刷新，早期的浏览器并不能原生支持ajax，可以使用隐藏帧（iframe）方式变相实现异步效果，后来的浏览器提供了对ajax的原生支持
>
> 使用ajax原生方式发送请求主要通过XMLHttpRequest(标准浏览器)、ActiveXObject(IE浏览器)对象实现异步通信效果

2. 同步和异步执行代码的区别?

   答:

> 同步：阻塞的
>
> =张三叫李四去吃饭，李四一直忙得不停，张三一直等着，直到李四忙完两个人一块去吃饭
>
> =浏览器向服务器请求数据，服务器比较忙，浏览器一直等着（页面白屏），直到服务器返回数据，浏览器才能显示页面
>
> 异步：非阻塞的
>
> =张三叫李四去吃饭，李四在忙，张三说了一声然后自己就去吃饭了，李四忙完后自己去吃
>
> =浏览器向服务器请求数据，服务器比较忙，浏览器可以自如的干原来的事情（显示页面），服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新

3. 页面编码和被请求的资源编码不一样如何处理?

   答:

> 对于ajax请求传递的参数，如果是get请求方式，参数如果传递中文，在有些浏览器会乱码，不同的浏览器对参数编码的处理方式不同，所以对于get请求的参数需要使用 encodeURIComponent函数对参数进行编码处理，后台开发语言都有相应的解码api。对于post请求不需要进行编码

4. 简述ajax的过程?

   答:

> 1. 创建XMLHttpRequest对象,也就是创建一个异步调用对象
> 2. 创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息

3. 设置响应HTTP请求状态变化的函数
4. 发送HTTP请求
5. 获取异步调用返回的数据
6. 使用JavaScript和DOM实现局部刷新

5. 请解释一下JavaScript的同源策略?

   答:

- [ ] 同源策略在什么情况下会起作用呢？ 当web 页面使用多个