**面试问题总结**

1.常见的http状态码有哪些？

**1XX：通知**

**1XX系列响应代码仅在与HTTP服务器沟通时使用。**

100("Continue")

**2XX表示请求成功**

200("OK")

201("Created")

202("Accepted")

203("Non-Authoritative Information")

204("No Content")……

**3XX表示是重定向**

300("Multiple Choices")

301("Moved Permanently")

303("See Other")

304("Not Modified")

## 4XX表示一般是前端的问题 客户端错误

400("Bad Request")

401("Unauthorized")

404("Not Found")

**5XX表示服务器端错误**

500("Internal Server Error")

**2.如何进行网站性能优化？**

压缩代码  DOM操作优化、CSS选择符优化、图片及HTML结构优化等

减少http请求 ——最重要最有效的方法

减少dom操作

尽量使用精灵图

压缩图片和文本

按需加载 页面图片很多的，可以使用懒加载。只加载第一屏的图片，当用户滚动访问后面的内容时再加 载相应图片。方法是先用一张极小的占位图代替图片，占位图只下载一次，将原本图片的src存储在 另一 个属性中，判断当图片快进入可视区域就将路径赋值给src并下载图片进行展示

局部刷新ajax

**3.null和undefined的区别是什么？**

null表示没有对象，即该处不应该有值

1） 作为函数的参数，表示该函数的参数不是对象

2） 作为对象原型链的终点

undefined表示缺少值，即此处应该有值，但没有定义

1）定义了形参，没有传实参，显示undefined

2）对象属性名不存在时，显示undefined

3）函数没有写返回值，即没有写return，拿到的是undefined

4）写了return，但没有赋值，拿到的是undefined

null和undefined转换成number数据类型

null 默认转成 0 undefined 默认转成 NaN

**4.从你输入网址到出现页面，浏览器做了什么？**

1、用户打开url链接  
　　2、浏览器查询url的dns地址  
　　3、提交url请求到服务器端  
　　4、服务器端处理  
　　5、传输处理好的html文本内容到浏览器  
　　6、浏览器解析html，并加载css，js，图片等内容  
　　7、加载完成，用户看到完整的页面内容

5.请说出css占位的隐藏和不占位的隐藏

6.css的定位有哪几种？分别相对于什么定位？

**7.说出3种你遇到的兼容问题，你是如何解决的？**

1、不同浏览器的标签默认的外补丁( margin )和内补丁(padding)不同

解决方案： css 里增加通配符 \* { margin: 0; padding: 0; }

2、IE6双边距问题；在 IE6中设置了float , 同时又设置margin , 就会出现边距问题

解决方案：设置display:inline;

3、当标签的高度设置小于10px，在IE6、IE7中会超出自己设置的高度

解决方案：超出高度的标签设置overflow:hidden,或者设置line-height的值小于你的设置高度

4、图片默认有间距

解决方案：使用float 为img 布局

5、IE9一下浏览器不能使用opacity

解决方案：opacity: 0.5;filter: alpha(opacity = 50);filter: progid:DXImageTransform.Microsoft.Alpha(style = 0, opacity = 50);

6、边距重叠问题；当相邻两个元素都设置了margin 边距时，margin 将取最大值，舍弃最小值；

解决方案：为了不让边重叠，可以给子元素增加一个父级元素，并设置父级元素为 overflow:hidden；

7、cursor:hand 显示手型在safari 上不支持

解决方案：统一使用 cursor:pointer

8、两个块级元素，父元素设置了overflow:auto；子元素设置了position:relative ;且高度大于父元素，在 IE6、IE7会被隐藏而不是溢出；

解决方案：父级元素设置position:relative

**8.js的事件有哪些？请举例说明？如果一个登陆按钮要添加键盘事件要怎么添加？**

点击事件，键盘事件，焦点事件，加载事件

    var $btn = document.querySelector(".btn");

    $btn.denglu = function() {

      window.location.href = "http://www.jd.com";

    };

    document.onkeydown = function(e) {

      console.log(e.keyCode);

      if (e.keyCode == 13) {

        $btn.denglu();

      }

    };

**9.简单介绍一个什么是事件冒泡？什么是事件捕获？什么是事件委托？**

事件冒泡会从当前触发的事件目标一级一级往上传递，依次触发，直到document为止。  
 事件捕获会从document开始触发，一级一级往下传递，依次触发，直到真正事件目标为止。

事件委托的原理是事件冒泡

为什么要用事件委托？

事件委托，通过监听一个父元素，来给不同的子元素绑定事件，减少监听次数，从而提升速度。

**10.js的函数有哪几种？你如何理解构造函数？**

1. 普通函数

function ShowName(name) {

alert(name);

}

2. .匿名函数

//变量匿名函数，左侧可以为变量、事件等

var anonymousNormal = function (p1, p2) {

alert(p1+p2);

}

anonymousNormal(3,6);//输出9

3. 闭包函数

假设，函数A内部声明了个函数B，函数B引用了函数B之外的变量，并且函数A的返回值为函数B的引 用。那么函数B就是闭包函数。

构造函数有什么用？

当你需要大批量的写对象的时候，就需要用到构造函数，它可以方便创建多个对象的实例，并且创建 的对象可以被标识为特定的类型，可以通过继承扩展代码

2.构造函数的特点

a:构造函数的首字母必须大写，用来区分于普通函数

b:内部使用的this对象，来指向即将要生成的实例对象

c:使用New来生成实例对象

11.margin塌陷问题是怎么出现的？如何解决？

父元素加marin-top子元素跟着下移的问题

1.overflow：hidden

2.加padding

3.父级写border

12.怎么添加移除创建修改查找DOM节点？

移除：parentNode.removeChild(node);

添加：parentNode.appendChild(newNode);在父节点中插入一个子节点

document.insertBefore(newNode,referenceNode);在某个节点前插入一个节点

创建：document.createElement(eName);

修改：replaceChild()

查找：parentObj.getElementsByTagName(tagName) 父找子

childNode.parentNode 子找父

neighbourNode.previousSibling 上一个节点

neighbourNode.nextSibling 下一个节点

**1.正则怎么定义？是做什么的？**

a,利用RegExp()构造函数来创建RegExp对象, new RegExp()

b,直接量语法创建,正则表达式直接包裹在一对斜杠(/)之间, /^ $/

验证数据格式是否正确

**2.说出手机号码的正则**

var reg = /^1（3|4|5|7|8|9）\d{9}$/

**3.说出2-4位中文的正则**

/^[\u4e00-\u9fa5]{2,4}$/

**4.正则的 . 代表什么含义？**

**·**：匹配除 "n" 之外的任何单个字符

**5.正则的转义符是干嘛的？**

**在正则表达式中，有许多字符有特殊含义，比如+表示重复多次，如果想直接使用这些符号，而忽略它们的含义，就需要对这些字符进行转义**

**转义：JavaScript正则表达式也支持非字母的字符匹配，这些字符需要通过反斜杠（\）进行转义**1、\n：换行  
2、\t：制表符  
3、\v：垂直制表符  
4、\r：回车符  
5、\f：换页符

**6.使用正则时用的方法是什么？**

a. test() :在字符串中查找符合正则的内容，若查找到返回true,反之返回false.

b. search() ：在字符串搜索符合正则的内容，搜索到就返回出现的位置（从0开始，如果匹配的不只是一个字 母，那只会返回第一个字母的位置）， 如果搜索失败就返回 -1

c. match() 在字符串中搜索复合规则的内容，搜索成功就返回内容，格式为数组，失败就返回null。

例子： var str = 'haj123sdk54hask33dkhalsd879';

var re = /\d+/g;

// 每次匹配至少一个数字 且全局匹配 如果不是全局匹配，当找到数字123，它就会停止了。

// 就只会弹出123.加上全局匹配，就会从开始到结束一直去搜索符合规则的。如果没有加号，匹配的结果就是1，2， 3，5，4， 3，3，8，7，9并不是我们想要的，

// 有了加号，每次匹配的数字就是至少一个了。

console.log( str.match(re) ); // [123，54，33，879]

**8.说出ajax的5大步骤+**

1. 创建XMLHTTPRequest对象
2. 使用open方法设置和服务器的交互信息
3. 设置发送的数据，开始和服务器端交互
4. 注册事件
5. 更新界面

**9.前端在前后台交互的时候都会做什么？**

页面（html+css+js）

获取数据ajax

处理数据dom es5 es6

传递数据ajax

**10.什么情况下会跨域？为什么会跨域？怎么解决跨域？**

使用ajax调用后台接口的时候，有时候会发生跨域

因为浏览器有个同源策略（不是自己家的不让进门）

类似浏览器的防盗门

关闭同源策略相当于拆掉防盗门

解决跨域的简单方法：

1.关闭浏览器同源策略；

2.后台加响应头

**1.前端的排序方式有哪几种？**

一.冒泡排序

原理：简单来说就是相邻两个元素进行对比，按照你需要的排序方式（升序or降序）进行位置替换，替换时需 要额外一个变量当作中间变量去暂存值。

二：选择排序

原理：首先从原始数组中找到最小的元素，并把该元素放在数组的最前面，然后再从剩下的元素中寻找最小的 元素，放在之前最小元素的后面，直到排序完毕

三：插入排序

原理：一组数据分成两组，分别将其称为有序组与待插入组。每次从待插入组中取出一个元素，与有序组的元 素进行比较，并找到合适的位置，将该元素插到有序组当中。就这样，每次插入一个元素，有序组增加， 待插入组减少。直到待插入组元素个数为0。

四：快速排序

原理：选择一个基准，将比基准小的放左边，比基准小的放在右边（基准处在中间位置）

**2.post和get的区别是啥？**

### ****GET 和 POST 方法的区别****：

**1、发送的数据数量**

在 GET 中，只能发送有限数量的数据，因为数据是在 URL 中发送的。

在 POST 中，可以发送大量的数据，因为数据是在正文主体中发送的。

**2、安全性**

GET 方法发送的数据不受保护，因为数据在 URL 栏中公开，这增加了漏洞和黑客攻击的风险。

POST 方法发送的数据是安全的，因为数据未在 URL 栏中公开，还可以在其中使用多种编码技术，这使其具有弹性。

**3、加入书签中**

GET 查询的结果可以加入书签中，因为它以 URL 的形式存在；而 POST 查询的结果无法加入书签中。

**4、编码**

在表单中使用 GET 方法时，数据类型中只接受 ASCII 字符。

在表单提交时，POST 方法不绑定表单数据类型，并允许二进制和 ASCII 字符。

**5、可变大小**

GET 方法中的可变大小约为 2000 个字符。

POST 方法最多允许 8 Mb 的可变大小。

**6、缓存**

GET 方法的数据是可缓存的，而 POST 方法的数据是无法缓存的。

**7、主要作用**

GET 方法主要用于获取信息。而 POST 方法主要用于更新数据。

**3.什么是事件委托？为什么要用事件委托？**

事件委托是利用时间冒泡的原理实现的一种代码优化的方式

举例：10个li要绑定点击事件需要有10个点击事件，因为有事件冒泡点li时会冒泡到ul，直接把事件绑在ul上，通过e.target获取li元素，这样就只绑定一个点击事件了

**4.什么是ajax？他有什么作用？**

ajax 主要是实现页面和 web 服务器之间数据的异步传输。

**5.请说出前后台交互是如何进行的？以登录注册为例**

登录：1.利用ajax请求登录接口的同时传输用户输入的用户名和密码

2.后台查询用户名和密码是否存在select name pwd from...

3.后台把查询结果返回

4.前端接收，反馈给用户

注册：1.利用ajax请求注册接口的同时传输用户输入的所有信息（后台让用什么名称传就用什么名称）

2.后台接受数据，使用insert into（增加的sql语句入库）

3.把入库是否成功的信息反馈给前端

4.前端接收，反馈给用户

**BOM的**

## Location 对象

Location 对象包含有关当前 URL 的信息。

Location 对象是 window 对象的一部分，可通过 window.Location 属性对其进行访问。

## Location 对象属性 Location 对象方法

|  |  |
| --- | --- |
| **属性** | **描述** |
| [hash](https://www.runoob.com/jsref/prop-loc-hash.html) | 返回一个URL的锚部分 |
| [host](https://www.runoob.com/jsref/prop-loc-host.html) | 返回一个URL的主机名和端口 |
| [hostname](https://www.runoob.com/jsref/prop-loc-hostname.html) | 返回URL的主机名 |
| [href](https://www.runoob.com/jsref/prop-loc-href.html) | 返回完整的URL |
| [pathname](https://www.runoob.com/jsref/prop-loc-pathname.html) | 返回的URL路径名。 |
| [port](https://www.runoob.com/jsref/prop-loc-port.html) | 返回一个URL服务器使用的端口号 |
| [protocol](https://www.runoob.com/jsref/prop-loc-protocol.html) | 返回一个URL协议 |
| [search](https://www.runoob.com/jsref/prop-loc-search.html) | 返回一个URL的查询部分 |

|  |  |
| --- | --- |
| **方法** | **说明** |
| [assign()](https://www.runoob.com/jsref/met-loc-assign.html) | 载入一个新的文档 |
| [reload()](https://www.runoob.com/jsref/met-loc-reload.html) | 重新载入当前文档 |
| [replace()](https://www.runoob.com/jsref/met-loc-replace.html) | 用新的文档替换当前文档 |

|  |  |
| --- | --- |
| **方法** | **说明** |
| [back()](https://www.runoob.com/jsref/met-his-back.html) | 加载 history 列表中的前一个 URL |
| [forward()](https://www.runoob.com/jsref/met-his-forward.html) | 加载 history 列表中的下一个 URL |
| [go()](https://www.runoob.com/jsref/met-his-go.html) | 加载 history 列表中的某个具体页面 |

## History 对象

History 对象包含用户（在浏览器窗口中）访问过的 URL。

History 对象是 window 对象的一部分，可通过 window.history

属性对其进行访问。

## History 对象属性方法

|  |  |
| --- | --- |
| **属性** | **说明** |
| [length](https://www.runoob.com/jsref/prop-his-length.html) | 返回历史列表中的网址数 |

**window对象**

**Window 对象表示浏览器中打开的窗口。**

## Window 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [closed](https://www.runoob.com/jsref/prop-win-closed.html) | 返回窗口是否已被关闭。 |
| [defaultStatus](https://www.runoob.com/jsref/prop-win-defaultstatus.html) | 设置或返回窗口状态栏中的默认文本。 |
| [document](https://www.runoob.com/jsref/dom-obj-document.html) | 对 Document 对象的只读引用。(请参阅[对象](https://www.runoob.com/jsref/dom-obj-document.html)) |
| [frames](https://www.runoob.com/jsref/prop-win-frames.html) | 返回窗口中所有命名的框架。该集合是 Window 对象的数组，每个 Window 对象在窗口中含有一个框架。 |
| [history](https://www.runoob.com/jsref/obj-history.html) | 对 History 对象的只读引用。请参数 [History 对象](https://www.runoob.com/jsref/obj-history.html)。 |
| [innerHeight](https://www.runoob.com/jsref/prop-win-innerheight.html) | 返回窗口的文档显示区的高度。 |
| [innerWidth](https://www.runoob.com/jsref/prop-win-innerheight.html) | 返回窗口的文档显示区的宽度。 |
| [localStorage](https://www.runoob.com/jsref/prop-win-localstorage.html) | 在浏览器中存储 key/value 对。没有过期时间。 |
| [length](https://www.runoob.com/jsref/prop-win-length.html) | 设置或返回窗口中的框架数量。 |
| [location](https://www.runoob.com/jsref/obj-location.html) | 用于窗口或框架的 Location 对象。请参阅 [Location 对象](https://www.runoob.com/jsref/obj-location.html)。 |
| [name](https://www.runoob.com/jsref/prop-win-name.html) | 设置或返回窗口的名称。 |
| [navigator](https://www.runoob.com/jsref/obj-navigator.html) | 对 Navigator 对象的只读引用。请参数 [Navigator 对象](https://www.runoob.com/jsref/obj-navigator.html)。 |
| [opener](https://www.runoob.com/jsref/prop-win-opener.html) | 返回对创建此窗口的窗口的引用。 |
| [outerHeight](https://www.runoob.com/jsref/prop-win-outerheight.html) | 返回窗口的外部高度，包含工具条与滚动条。 |
| [outerWidth](https://www.runoob.com/jsref/prop-win-outerheight.html) | 返回窗口的外部宽度，包含工具条与滚动条。 |
| [pageXOffset](https://www.runoob.com/jsref/prop-win-pagexoffset.html) | 设置或返回当前页面相对于窗口显示区左上角的 X 位置。 |
| [pageYOffset](https://www.runoob.com/jsref/prop-win-pagexoffset.html) | 设置或返回当前页面相对于窗口显示区左上角的 Y 位置。 |
| [parent](https://www.runoob.com/jsref/prop-win-parent.html) | 返回父窗口。 |
| [screen](https://www.runoob.com/jsref/obj-screen.html) | 对 Screen 对象的只读引用。请参数 [Screen 对象](https://www.runoob.com/jsref/obj-screen.html)。 |
| [screenLeft](https://www.runoob.com/jsref/prop-win-screenleft.html) | 返回相对于屏幕窗口的x坐标 |
| [screenTop](https://www.runoob.com/jsref/prop-win-screenleft.html) | 返回相对于屏幕窗口的y坐标 |
| [screenX](https://www.runoob.com/jsref/prop-win-screenx.html) | 返回相对于屏幕窗口的x坐标 |
| [sessionStorage](https://www.runoob.com/jsref/prop-win-sessionstorage.html) | 在浏览器中存储 key/value 对。 在关闭窗口或标签页之后将会删除这些数据。 |
| [screenY](https://www.runoob.com/jsref/prop-win-screenx.html) | 返回相对于屏幕窗口的y坐标 |
| [self](https://www.runoob.com/jsref/prop-win-self.html) | 返回对当前窗口的引用。等价于 Window 属性。 |
| [status](https://www.runoob.com/jsref/prop-win-status.html) | 设置窗口状态栏的文本。 |
| [top](https://www.runoob.com/jsref/prop-win-top.html) | 返回最顶层的父窗口。 |

## Window 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [alert()](https://www.runoob.com/jsref/met-win-alert.html) | 显示带有一段消息和一个确认按钮的警告框。 |
| [atob()](https://www.runoob.com/jsref/met-win-atob.html) | 解码一个 base-64 编码的字符串。 |
| [btoa()](https://www.runoob.com/jsref/met-win-btoa.html) | 创建一个 base-64 编码的字符串。 |
| [blur()](https://www.runoob.com/jsref/met-win-blur.html) | 把键盘焦点从顶层窗口移开。 |
| [clearInterval()](https://www.runoob.com/jsref/met-win-clearinterval.html) | 取消由 setInterval() 设置的 timeout。 |
| [clearTimeout()](https://www.runoob.com/jsref/met-win-cleartimeout.html) | 取消由 setTimeout() 方法设置的 timeout。 |
| [close()](https://www.runoob.com/jsref/met-win-close.html) | 关闭浏览器窗口。 |
| [confirm()](https://www.runoob.com/jsref/met-win-confirm.html) | 显示带有一段消息以及确认按钮和取消按钮的对话框。 |
| [createPopup()](https://www.runoob.com/jsref/met-win-createpopup.html) | 创建一个 pop-up 窗口。 |
| [focus()](https://www.runoob.com/jsref/met-win-focus.html) | 把键盘焦点给予一个窗口。 |
| getSelection() | 返回一个 Selection 对象，表示用户选择的文本范围或光标的当前位置。 |
| [getComputedStyle()](https://www.runoob.com/jsref/jsref-getcomputedstyle.html) | 获取指定元素的 CSS 样式。 |
| [matchMedia()](https://www.runoob.com/jsref/met-win-matchmedia.html) | 该方法用来检查 media query 语句，它返回一个 MediaQueryList对象。 |
| [moveBy()](https://www.runoob.com/jsref/met-win-moveby.html) | 可相对窗口的当前坐标把它移动指定的像素。 |
| [moveTo()](https://www.runoob.com/jsref/met-win-moveto.html) | 把窗口的左上角移动到一个指定的坐标。 |
| [open()](https://www.runoob.com/jsref/met-win-open.html) | 打开一个新的浏览器窗口或查找一个已命名的窗口。 |
| [print()](https://www.runoob.com/jsref/met-win-print.html) | 打印当前窗口的内容。 |
| [prompt()](https://www.runoob.com/jsref/met-win-prompt.html) | 显示可提示用户输入的对话框。 |
| [resizeBy()](https://www.runoob.com/jsref/met-win-resizeby.html) | 按照指定的像素调整窗口的大小。 |
| [resizeTo()](https://www.runoob.com/jsref/met-win-resizeto.html) | 把窗口的大小调整到指定的宽度和高度。 |
| scroll() | 已废弃。 该方法已经使用了 [scrollTo()](https://www.runoob.com/jsref/met-win-scrollto.html) 方法来替代。 |
| [scrollBy()](https://www.runoob.com/jsref/met-win-scrollby.html) | 按照指定的像素值来滚动内容。 |
| [scrollTo()](https://www.runoob.com/jsref/met-win-scrollto.html) | 把内容滚动到指定的坐标。 |
| [setInterval()](https://www.runoob.com/jsref/met-win-setinterval.html) | 按照指定的周期（以毫秒计）来调用函数或计算表达式。 |
| [setTimeout()](https://www.runoob.com/jsref/met-win-settimeout.html) | 在指定的毫秒数后调用函数或计算表达式。 |
| [stop()](https://www.runoob.com/jsref/met-win-stop.html) | 停止页面载入。 |

## Screen 对象

Screen 对象包含有关客户端显示屏幕的信息。

## Screen 对象属性

|  |  |
| --- | --- |
| **属性** | **说明** |
| [availHeight](https://www.runoob.com/jsref/prop-screen-availheight.html) | 返回屏幕的高度（不包括Windows任务栏） |
| [availWidth](https://www.runoob.com/jsref/prop-screen-availwidth.html) | 返回屏幕的宽度（不包括Windows任务栏） |
| [colorDepth](https://www.runoob.com/jsref/prop-screen-colordepth.html) | 返回目标设备或缓冲器上的调色板的比特深度 |
| [height](https://www.runoob.com/jsref/prop-screen-height.html) | 返回屏幕的总高度 |
| [pixelDepth](https://www.runoob.com/jsref/prop-screen-pixeldepth.html) | 返回屏幕的颜色分辨率（每象素的位数） |
| [width](https://www.runoob.com/jsref/prop-screen-width.html) | 返回屏幕的总宽度 |

## Navigator 对象

Navigator 对象包含有关浏览器的信息。

## Navigator 对象属性 Navigator 对象方法

|  |  |
| --- | --- |
| **属性** | **说明** |
| [appCodeName](https://www.runoob.com/jsref/prop-nav-appcodename.html) | 返回浏览器的代码名 |
| [appName](https://www.runoob.com/jsref/prop-nav-appname.html) | 返回浏览器的名称 |
| [appVersion](https://www.runoob.com/jsref/prop-nav-appversion.html) | 返回浏览器的平台和版本信息 |
| [cookieEnabled](https://www.runoob.com/jsref/prop-nav-cookieenabled.html) | 返回指明浏览器中是否启用 cookie 的布尔值 |
| [platform](https://www.runoob.com/jsref/prop-nav-platform.html) | 返回运行浏览器的操作系统平台 |
| [userAgent](https://www.runoob.com/jsref/prop-nav-useragent.html) | 返回由客户机发送服务器的user-agent 头部的值 |

|  |  |
| --- | --- |
| **方法** | **描述** |
| [javaEnabled()](https://www.runoob.com/jsref/met-nav-javaenabled.html) | 指定是否在浏览器中启用Java |
| [taintEnabled()](https://www.runoob.com/jsref/met-nav-taintenabled.html) | 规定浏览器是否启用数据污点(data tainting) |

**查询1：什么是websocket？websocket能做什么？**

WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。

在 WebSocket API 中，浏览器和服务器只需要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。

创建 WebSocket 对象。

var Socket = new WebSocket(url, [protocol] );  
以上代码中的第一个参数 url, 指定连接的 URL。第二个参数 protocol 是可选的，指定了可接受的子协议。

WebSocket API是HTML5标准的一部分， 但这并不代表 WebSocket 一定要用在 HTML 中，或者只能在基于 浏览器的应用程序中使用。

**基于TCP全双工协议，与HTTP概念类似，体量较小，可以发送文本和二进制数据，解决了多次建立HTTP连接，消耗过多资源的问题。**

**查询2：webstorage和cookie还有session有什么区别？**

cookie是存于用户硬盘的一个文件，cookie是浏览器提供的一种机制，可以由js对其进行控制，而不是js本身的性质

session相当于程序在服务器上建立一份客户档案

Web Storage 包含sessionStorage和localStorage

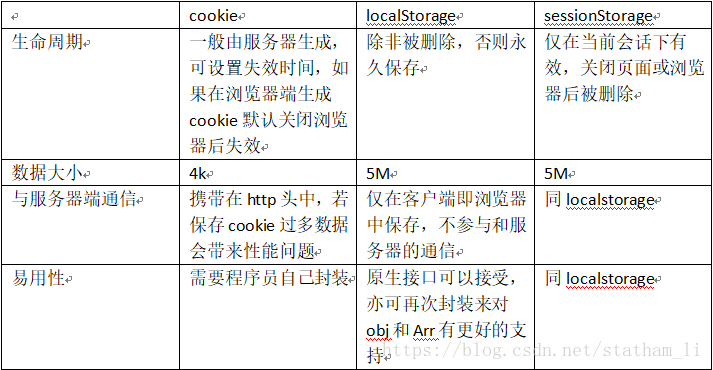
Web Storage API 提供机制， 使浏览器能以一种比使用Cookie更直观的方式存储键/值对。

区别：

session 保存在服务器，客户端不知道其中的信息；cookie 保存在客户端，服务器能够知道其中的信息。

session 中保存的是对象，cookie 中保存的是字符串。

安全性上WebStorage 并不作为 HTTP header 发送的浏览器，所以相对安全。



**查询3：echarts和highcharts有什么区别？**

1.highcharts是外国的公司的专业团队当做公司的重要产品去开发迭代的开源项目

echarts是国内百度公司的一个技术团队维护开的开源工具项目。

2. echarts底层为canvas，highcharts底层为svg

canvas特点：

    ①依赖分辨率

    ②不支持事件处理器

    ③弱的文本渲染能力

    ④能够以.jpg、.png格式保存结果图像

⑤最适合图像密集型的游戏，其中的许

多对象会被频繁重绘

基本上是一个用js操作的位图

svg特点：

①不依赖分辨率

②支持事件处理器

③最适合带有大型渲染区域的应用程序（如谷歌地图）

④复杂度高会减慢渲染速度（任何过度使用DOM的应用都不快）

 ⑤不适合游戏应用

**1.html5的新特性有哪些？**

1.语义化标签

2.增强型表单

3.新增视频 <video> 和音频 <audio> 标签

4.Canvas绘图

5.SVG绘图

6.地理定位

7.拖放API

8.Web Worker

9.Web Storage

10.WebSocket

**2.css3的新特性有哪些？**

1.边框圆角，渐变，阴影之类的样式

2.过渡效果和动画

3.弹性盒子

4.@media

5.CSS3的选择器

1）E:last-child 匹配父元素的最后一个子元素E。

2）E:nth-child(n)匹配父元素的第n个子元素E。

3）E:nth-last-child(n) CSS3 匹配父元素的倒数第n个子元素E。

6. @Font-face 特性

Font-face 可以用来加载字体样式，而且它还能够加载服务器端的字体文件，让客户端显示客户端所没有 安装的字体。

7. 边框圆角

8. 多列布局 （multi-column layout）

9.阴影（Shadow）

10.CSS3 的渐变效果

11.css弹性盒子模型

12. CSS3制作特效

Transition 对象变换时的过渡效果

transition-property 对象参与过渡的属性  
transition-duration 过渡的持续时间  
transition-timing-function 过渡的类型  
transition-delay 延迟过渡的时间

13.Transforms 2D转换效果

14.Animation动画特效

**6.jquery相对于原生js有哪些优点？**

封装：dom ajax 一部分bom 三大家族

优点：write less do more

缺点：不能向后兼容

**8.前端的 布局方式有哪几种？**

移动端常用：弹性盒子，百分比布局

pc端常用：

官网（介绍）平台（带支付功能等）：

静态布局（基本没有兼容问题），国家官网

响应式布局（做官网的主流）bootstrap 移动端优先

问题：汇总

自适应布局

后台系统：（一般用于内部使用，不对外开放）一般用ui框架搭建

流式布局（三栏布局）

**12.bootstrap的实现原理？可以兼容到ie几？**

bootstrap的实现原理

通过定义容器的大小，平分为12份（也有24份，23份的，取12份是一打，表示进阶完美的意思），再调整内外边距，最后结合媒体查询，制作出了强大的响应式网格系统。

插件：swiper iscoll jquery-ui animate.css

13.你常用的ui框架有哪些？你觉得哪个好用？为什么？

bootrap layui (element-ui mint-ui)饿了么团队

**gulp和babel的作用和用法总结**

node.js可以理解为环境，gulp和babel可以理解为可以实现不同功能的软件

bable作用：因为es6兼容性差，通过bable可以将es6编译为es5文件

gulp作用：可以实现编译scss，压缩css，压缩js，实时刷新等各种优化功能

**let const，var的区别**

1，var定义的变量，作用域是整个封闭函数，是全域的； var创建的变量可以覆盖

let定义的变量，作用域是在块级或者字块中； let创建的变量只可以创建一次，不允许覆盖  
2，变量提升：不论通过var声明的变量处于当前作用于的第几行，都会提升到作用域的最顶部。  
      而let声明的变量不会在顶部初始化，凡是在let声明之前使用该变量都会报错（引用错误ReferenceError）；  
3，只要块级作用域内存在let，它所声明的变量就会绑定在这个区域；  
4，let不允许在相同作用域内重复声明（报错同时使用var和let，两个let）。  
  const用来专门声明一个常量，它跟let一样作用于块级作用域，没有变量提升，重复声明会报错，不同的是const 声明的常量不可改变，声明时必须初始化（赋值）

const一般用来声明常量，且声明的常量是不允许改变的，只读属性，因此就要在声明的同时赋值

**箭头函数和普通函数的区别**

区别主要在于this指向这个地方

1. 箭头函数是匿名函数，不能作为构造函数，不能使用new

2.箭头函数的this永远指向其上下文的this，没有办改变其指向，普通函数的this指向调用它的对象

3.箭头函数不绑定this，会捕获其所在的上下文的this值，作为自己的this值

**symbol的特点**

ES6 引入了一种新的原始数据类型 Symbol，表示独一无二的值

特点：

1. Symbol 值通过 Symbol 函数生成，使用 typeof，结果为 "symbol"

2. Symbol 函数前不能使用 new 命令，否则会报错。这是因为生成的 Symbol 是一个原始类型的值，不是对象。

3. instanceof 的结果为 false

4. Symbol 函数可以接受一个字符串作为参数，表示对 Symbol 实例的描述，主要是为了在控制台显示，或者转为字符串时，比较容易区分。

5. 如果 Symbol 的参数是一个对象，就会调用该对象的 toString 方法，将其转为字符串，然后才生成一个 Symbol 值。

6. Symbol 函数的参数只是表示对当前 Symbol 值的描述，相同参数的 Symbol 函数的返回值是不相等的

Symbol≠Symbol

7. Symbol 值不能与其他类型的值进行运算，会报错。

8. Symbol 值可以显式转为字符串。

9. 如果我们希望使用同一个 Symbol 值，可以使用 Symbol.for。它接受一个字符串作为参数，然后搜索有没有以该参数作为名称的 Symbol 值。如果有，就返回这个 Symbol 值，否则就新建并返回一个以该字符串为名称的 Symbol 值。

10. Symbol.keyFor 方法返回一个已登记的 Symbol 类型值的 key。

var s1 = Symbol.for("foo");

console.log(Symbol.keyFor(s1)); *// "foo"*

var s2 = Symbol("foo");

console.log(Symbol.keyFor(s2) ); *// undefined*

**es6新增的常用属性**

**兼容性：IE10以上版本**

**1. .声明变量 let/const**

let:用来声明变量，类似于var，但是使用let有以下特性：

（1）.不存在变量提升

（2）.暂时性死区

（3）.不允许重复声明，同一作用域内只能声明一次

（4）.新增的块级作用域

const：用来声明常量，使用规则除了声明之后不能被修改，其他特性和let一样，本质上是const是内存地址不得改动，而不是变量的值不能改动

（1）.const声明一个只读的常量。一旦声明，常量的值就不能改变。

(2). const声明的变量不得改变值，这意味着，const一旦声明变量，就必须立即初始化，不能留到以后赋值。

(3). const的作用域与let命令相同：只在声明所在的块级作用域内有效。

(4). const命令声明的常量也是不提升，同样存在暂时性死区，只能在声明的位置后面使用。

(5). const声明的常量，也与let一样不可重复声明。

**2. 字符串扩展**

(1).codePointAt()

ES6 提供了codePointAt()方法，能够正确处理 4 个字节储存的字符，返回一个字符的码点。codePointAt()方法会正确返回 32 位的 UTF-16 字符的码点。对于那些两个字节储存的常规字符，它的返回结果与charCodeAt()方法相同。

(2).fromCodePoint()

ES6 提供了String.fromCodePoint()方法，可以识别大于0xFFFF的字符，弥补了String.fromCharCode()方法的不足。在作用上，正好与上面的codePointAt()方法相反。注意，fromCodePoint方法定义在String对象上，而codePointAt方法定义在字符串的实例对象上。

(3).repeat()

repeat方法返回一个新字符串，表示将原字符串重复n次。

(4).indexof()补充includes() startswith() endswith()

includes()：返回布尔值，表示是否找到了参数字符串。

startsWith()：返回布尔值，表示参数字符串是否在原字符串的头部。

endsWith()：返回布尔值，表示参数字符串是否在原字符串的尾部。

**3.解构赋值**

ES6 允许按照一定模式，从数组和对象中提取值，对变量进行赋值，这被称为解构

ES6 允许写成下面这样。 let [a, b, c] = [1, 2, 3];

（1）. 对象的解构赋值

解构不仅可以用于数组，还可以用于对象。

let { foo, bar } = { foo: 'aaa', bar: 'bbb' };

foo // "aaa"

bar // "bbb"

对象的解构与数组有一个重要的不同。数组的元素是按次序排列的，变量的取值由它的位置决定；而对象的属性没有次序，变量必须与属性同名，才能取到正确的值

(2). 字符串的解构赋值

字符串也可以解构赋值。这是因为此时，字符串被转换成了一个类似数组的对象。

const [a, b, c, d, e] = 'hello';

a // "h"

b // "e"

c // "l"

d // "l"

e // "o"

类似数组的对象都有一个length属性，因此还可以对这个属性解构赋值。

let {length : len} = 'hello';

len // 5

(3). 数值和布尔值的解构赋值

解构赋值时，如果等号右边是数值和布尔值，则会先转为对象。

let {toString: s} = 123;

s === Number.prototype.toString // true

let {toString: s} = true;

s === Boolean.prototype.toString // true

上面代码中，数值和布尔值的包装对象都有toString属性，因此变量s都能取到值。

**gulp是什么？你都使用过gulp的哪些插件？**

定义：是一款基于node.js的自动化构建工具。

自动化构建工具可以做什么：编译，压缩，操作浏览器.

功能：1. 编译 sass 2.合并优化压缩 css 3.校验压缩 js 4.优化图片5.添加文件指纹（md5） 6.组件化头部底部（include html） 7.实时自动刷新… 8.压缩静态资源 9. 变更静态资源 10.给静态资源添加 md5（加密）11. 修改预处理样式后自动编译（SASS，Less）12.合并雪碧图13. 自动刷新浏览器15. Css Js 图片压缩16. Css Js 合并17. Css Js 内联18. Html的include功能19. Autoprefixer20. 去缓存

**什么是node.js？**

Nodejs 是基于 Chrome 的 V8 引擎开发的一个 C++ 程序，目的是提供一个 JS 的运行环境。

**你一般如何解决js的兼容问题？**

1.js使用jquery;jq可以兼容到IE6

2.es6用babel编译器编译成es5格式

**es6兼容性如何？兼容哪些浏览器？**

可以兼容主流浏览器（Chrome，Firefox，Safari）

IE：6-8支持部分属性 兼容10版本以上

**jquery兼容性如何？兼容哪些浏览器？**

可以兼容IE6以上及主流浏览器

**bootstrap兼容性如何？兼容哪些浏览器？**

可以兼容IE8（不含）以上及主流浏览器

**layui兼容性如何？兼容哪些浏览器？**

可以兼容（除IE6-7）以外的所有浏览器

**javascript编译器你知道哪些？**

把ES6转换为ES5，比较通用的工具有babel、jsx、traceur、es6-shim等

**举个简单示例说明什么是解构赋值？**

let { foo, bar } = { foo: 'aaa', bar: 'bbb' };

**es6的模块化如何使用？**

ES6的模块化的基本规则或特点：

　　　　1：每一个模块只加载一次， 每一个JS只执行一次， 如果下次再去加载同目录下同文件，直接从内存中 读取。 一个模块就是一个单例，或者说就是一个对象；

　　　　2：每一个模块内声明的变量都是局部变量， 不会污染全局作用域；

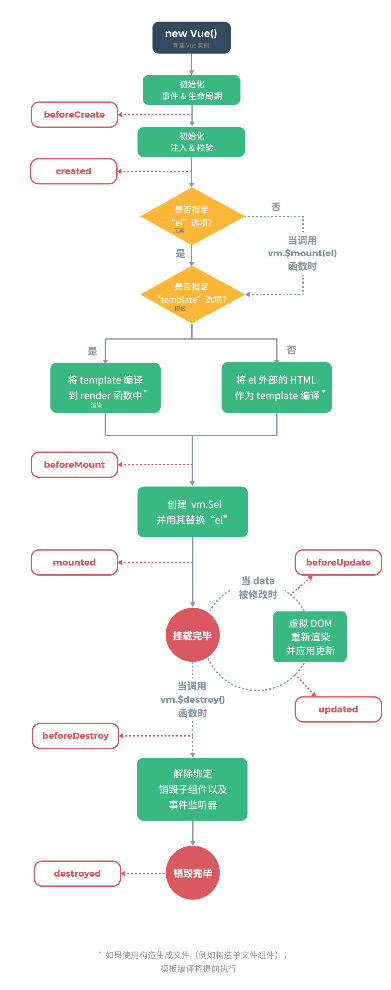
　　　　3：模块内部的变量或者函数可以通过export导出；

　　　　4：一个模块可以导入别的模块

使用：1. 使用export default导出 2. import可以导入另一个模块中的内容

Vue个人总结

**1.vue的生命周期？每一步具体干嘛的？**

**1.什么是vue生命周期？**  
答： Vue 实例从创建到销毁的过程，就是生命周期。从开始创建、初始化数据、编译模板、挂载Dom→渲染、更新→渲染、销毁等一系列过程，称之为 Vue 的生命周期。

**2.vue生命周期的作用是什么？**  
答：它的生命周期中有多个事件钩子，让我们在控制整个Vue实例的过程时更容易形成好的逻辑。

**3.第一次页面加载会触发哪几个钩子？**

答：beforeCreate， created， beforeMount， mounted

创建：前：**beforeCreate** 后：**created**

载入：前：**beforeMount** 后：**mounted**

更新：前：**beforeUpdate** 后：**updated**

销毁：前：**beforeDestroy** 后：**destroyed**

**2.vue的指令有哪些？把学过的说全面？**

v-model 数据绑定

v-for 循环 格式 v-for="字段名 in(of) 数组json"

v-show 显示 隐藏 传递的值为布尔值 true false 默认为false 原理：display:none

v-if 显示与隐藏 和v-show对比的区别 就是是否删除dom节点 默认值为false

v-else-if 必须和v-if连用

v-else 必须和v-if连用 不能单独使用 否则报错 模板编译错误

v-bind 动态绑定 作用： 及时对页面的数据进行更改

v-on 绑定事件 函数必须写在methods里面 @click 快捷方法

v-text 解析文本

v-html 解析html标签

v-bind:class 三种绑定方法 1、对象型 '{red:isred}' 2、三目型 'isred?"red":"blue"' 3、数组型

v-once 进入页面时 只渲染一次 不在进行渲染

v-cloak 防止闪烁

v-pre 把标签内部的元素原位输出

**3.vue的事件如何绑定？**

使用<v-on:事件名 = 函数名>来完成 函数写在method中

**4.vue的属性如何绑定？属性包含了哪些？**

语法： v-bind:属性名.修饰符="属性值"

简写： ：属性名.修饰符 = "属性值"

包含：class属性；style属性；监听属性：watch和computed

**5.vue的父子组件如何进行通讯？详细**

①父组件传给子组件：子组件通过props方法接受数据;

子组件通过两种方式接收：可以传递任何类型（数组，对象，各种数据类型等等）

props:[‘title’,‘likes’,‘isPublished’,‘author’]；

props:{title:String,likes:Number}  
②子组件传给父组件：$emit方法传递参数

子组件向父组件传值用 this.$emit(key,value) ，父组件接收的时候需要在父组件中创建的子组件的标签中绑定 Key，格式:@Key=“方法名”，父组件声明这个方法，方法带参数，这个参数就是子组件传递的 Value。

**6.vue的路由实现方式有几种？怎么实现？**

**①**<router-link to='需要跳转到的页面的路径>

**②**this.$router.push({ 路径’})

**③**this.$router.replace{ 路径 }

**7.什么叫单页面应用？实现单页面应用的框架有哪些?**

单页面应用：指一个系统只加载一次资源，然后下面的操作交互、数据交互是通过router、ajax来进行，页面并没有刷新； vue，React，angular 三大框架都可以实现单页面应用

**8.实现模块化的工具？**

Common.Js 是一种 JavaScript 语言的模块化规范

①.AMD（require.js）

②.CMD(sea.js)

**9.自动化构建工具有哪些？具体是做什么的？**

有哪些：Grunt、Gulp、Webpack

做什么：合并、压缩文件；编译预处理样式等…

**11.时间格式如何转化成时间戳？**

New Date();get Time();valueOf()

**12.什么是vue-cli？**

Vue脚手架

**13vue的axios是什么？怎么用？**

axios 是一个基于 promise 的 HTTP 库，可以发送 get，post 请求

发送get请求：this.$axios.get('/user?stu\_id=1002')

发送post请求：var params = new URLSearchParams(); params.append('name','孙悟空'); params.append('age',22);let that = this;this.$axios.post('http://localhost:8082/user',params)

**14.什么mock.js？你一般用来干嘛？**

在没有接口的时候，使用它来模拟接口

**15.什么是webpack？webpack可以做什么？**

webpack 是一个现代 JavaScript 应用程序的静态模块打包器

可以打包项目;可以实现模块化:可以使复杂的程序细化成为各个小的文件

预处理器 可以对Scss，less等CSS预先进行处理

**16.对于MVVM的理解？**

MVVM 是 Model-View-ViewModel 的缩写。

Model代表数据模型，也可以在Model中定义数据修改和操作的业务逻辑。

View 代表UI 组件，它负责将数据模型转化成UI 展现出来。

ViewModel 监听模型数据的改变和控制视图行为、处理用户交互，简单理解就是一个同步View 和 Model的对象，连接Model和View。

在MVVM架构下，View 和 Model 之间并没有直接的联系，而是通过ViewModel进行交互，Model 和 ViewModel 之间的交互是双向的， 因此View 数据的变化会同步到Model中，而Model 数据的变化也会立即反应到View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

**17. Vuex 是什么？哪种功能场景使用它？**

Vuex 是专门为 Vue.js 设计的状态管理模式，它采用集中式储存管理 Vue 应用中所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。

当项目庞大的时候使用它：

需要动态的注册响应式数据；需要命名空间 namespace 来管理组织我们的数据；希望通过插件，来更改记录；方便测试；以上这些需要和希望，都是我们 vuex 需要做的一些事情。

**18. Vuex 有哪几种属性？**

state：基本数据

getters：从基本数据派生的数据

mutations：提交更改数据的方法，同步！

actions：像一个装饰器，包裹

mutations，使之可以异步。

modules：模块化 Vuex。

**19. Vue 打包命令是什么？Vue 打包后会生成哪些文件？**

npm run build ：Vue 打包命令

Vue 打包后会在当前工作目录下生成一个 dist 文件夹，文件夹中会有 static 静态文件以及 index.html 初始页面。