1、jquery的一些接口

添加类

删除类

ajax，里面的参数是怎么样的

http中get和post的区别

2、高级程序设计

基本类型

数据类型识别（typeof）

array、date是object吗?

基本数据类型和引用类型在存储上的差别（栈和堆）

==和===的差别

构造函数、继承

绑定事件，第三个参数，true是捕获，false是冒泡。事件委托的时候，绑定事件的元素上事件会被触发几次？（一次，因为绑定的时候，一定要指定是冒泡还是捕获，捕获的时候就不冒泡，冒泡的时候就不捕获。）

dom时间流的过程：捕获阶段、目标阶段、冒泡阶段

不使用代理事件绑定在自身元素上的时候，没有冒泡还是捕获只说，就是在目标上触发。

闭包的概念（为什么外层函数可以访问其他函数内部的变量）

跨域的方法

3、其他知识：

计算机网络：网络结构

面向对象：什么是封装

js高级程序、css基础

网易实习校招现场面

周红伟

jsonp优缺点

优点：

第一个优点当然是能跨域了。一个访问不再受限于域名了，这个代表什么呢？代表我可以提供一个公共的webservice了，这个服务可以给你服务，也可以给他服务，你们不需要一定是在我的域名下的。

其次的优点是将回调方法的权限给了调用方。这个就相当于将controller层和view层终于分开了。我提供的jsonp服务只提供纯服务的数据，至于提供服务以后的页面渲染和后续view操作都由调用者来自己定义就好了。如果有两个页面需要渲染同一份数据，你们只需要有不同的渲染逻辑就可以了，逻辑都可以使用同一个jsonp服务。

还有一个优点是它甚至不需要浏览器能支持XMLHTTPRequest，就是说所有的浏览器都可以使用这个技术。

缺点：

首先的缺点是安全性。万一假如提供jsonp的服务存在页面注入漏洞，即它返回的javascript的内容被人控制的。那么结果是什么？所有调用这个jsonp的网站都会存在漏洞。于是无法把危险控制在一个域名下...所以在使用jsonp的时候必须要保证使用的jsonp服务必须是安全可信的。

其次是错误处理，jsonp在调用失败的时候不会返回各种HTTP状态码。只有200，没有404，没有500等状态码让你来标识是否要重新调用。

它只能支持GET，而不能支持POST请求，所以它的参数一定是带在HTTP头中的，会受到一些参数的限制，比如长度限制。

轮播图实现原理

jquery插件编写

前端生态圈

标签页之间的通信方式

访问网页的过程

http协议

canvas、svg

这么防止js注入

xss攻击，对输入的内容进行过滤。
项目里你收获了什么
js继承有哪些方法
var a=[1,2];var b=[1,2];a==b?
==和===
jquery用过过什么（选择器）
jquery好的地方
谷歌调试断点有哪些
新型技术知道什么
看过哪些书
为什么选择前端
你喜欢或者想做什么产品
组员不配合你工作，怎么办
假如你有很多offer，你怎么选择
快排算法

王兴菲
mysql的增删改查
Linux命令，例如显示当前目录，显示详细信息
最满意的项目，在其中的角色，详细描述项目中负责的部分
前端页面时怎么搭建的，用的bootstrap的哪些东西。详细介绍一下栅格系统，是怎么布局的。css饰怎么实现的，12列能增加吗。写一下，左边一个div，右边一个div，是怎么布局的，html和css都写。先写定宽的再写定宽的。col-xs是什么意思，是多少像素，如果写成col-sm，两个div会怎么显示，如果小于768会怎么样？992 1200
绑定事件，
事件流是什么过程
有一个input，有一个focus事件，他会有冒泡事件吗
平时怎么记录学习的东西，说一下纪录的内容。创建对象，为什么叫工厂模式，知道工厂的概念吗
看过什么书
用的新技术，讲一讲新技术的好处，为什么会方便
多页单页
DNS解析过程

jquery用过哪些

me

作用域

字符串中替换子字符串：

两个标签页如何进行通信：

1、利用localStorage的**storage事件**：当同一个域中的某个标签页设置了localStorage，同一个域中其他的打开的标签页都会触发storage事件。

storage事件的使用方法：

```
window.addEventListener("storage", function(e){
  console.log(e);
  document.write("oldValue: "+ e.oldValue + " newValue:" + e.newValue)
});
```

storage的events对象的属性常用的如下：

oldValue：更新前的值。如果该键为新增加，则这个属性为null。

newValue：更新后的值。如果该键被删除，则这个属性为null。

url：原始触发storage事件的那个网页的网址。

key：存储store的key名；

例子：

标签页1:

```
(function(D){
  var val = D.getElementsByTagName("input")[0],
    btn = D.getElementsByTagName("button")[0];
  btn.onclick = function(){
    var value = val.value;
    if(!value) return;
    localStorage.setItem("key", val.value);
  };
  window.addEventListener("storage", function(e){
    console.log(e);
  });
})(document);
```

标签页2:

```
window.addEventListener("storage", function(e){
  console.log(e);
  document.write("oldValue: "+ e.oldValue + " newValue:" + e.newValue)
});
```

2、利用**cookie+setInterval**：将要传递的信息存储在cookie中，每隔一定时间读取cookie信息，即可随时获取要传递的信息。

例子：

标签页1:

```
<input id="name">
<input type="button" id="btn" value="提交">
<script type="text/javascript">
    $(function(){
        $("#btn").click(function(){
            var name=$("#name").val();
            document.cookie="name="+name;
        });
    });
</script>
```

标签页2:

```
<script type="text/javascript">
    $(function(){
        function getCookie(key) {
            return JSON.parse("{\"" + document.cookie.replace(/;\s+/gim,"\",\"").replace(/=/gim, "\":\"") + "\"}")
        }
        setInterval(function(){
            console.log("name=" + getCookie("name"));
        }, 10000);
    });
</script>
```

3、webWorker的postMessage（前端面试经典题目集合）

nodes

es6

闭包 他写让你判断结果

继承 手写

上下两栏布局，下面自适应，不用position

```
<html style="height: 100%">
<head>
    <title></title>
</head>
<body style="height: 100%">
<div style="height: 100%;background-color: yellow;box-sizing: border-box;">
    <div style="height: 200px;background-color: red;float: left;width:
```

100%">3s
sdf

</body>
</html>
块级元素和行内元素的区别
dom插入节点
position:relative会不会挤掉后面的元素？不会
获取某个类名的元素
typeof识别array object，Object.toString.call(array)
<!DOCTYPE html>是用来干嘛的
有没有用过现在比较新的一些框架
h5新特征
css3新属性
数组去重
jquery好的地方
轮播图的实现
webpack（不知道是不是这么写的）
前端生态圈知道吗
对闭包的理解
介绍你的项目，功能，负责的部分
最近浏览了什么知识
对自己的评价
有没有投其他地方
平时怎么巩固自己，看什么书

- **box-sizing实现上下布局**

函数提升（定义了b = 1,function b(){} log(b)，输出1），函数内部对arguments赋值
函数的提升：

用function(){}声明的函数会被提升到最前面（整个函数），用var a = 1定义的变量，只是声明提升，赋值行为不会提升.
在函数内部改变arguments类数组，也会将传入的参数改变。

不new能不能继承
对于原型链继承来说，一定要new，否则只是执行了一下父函数，毫无意义。

标签语意化
根据内容的结构化（内容语义化），选择合适的标签（代码语义化）便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好地解析。
应该尽可能少的使用无语意的标签，例如div、span。
以前的主要的语意化的标签：li：list、H1～H6:head、strong：加粗、em：斜体、table、thead、tbody、tfoot。
h5新增语意化标签：article、nav、aside、section、header、footer、hgroup、address、mark：加背景等。

获取class类的元素
element.getElementbyClassName()、
element.querySelector/element.querySelectorAll(“.className”)

哪些事件不冒泡
abort　在图像加载被中断时发生，当用户在图像完成载入之前放弃图像的装载（如单击了 stop 按钮）时，就会调用该句柄。

blur
error
focus
load
mouseenter
mouseleave
resize
unload
冒泡的事件：
beforeinput
click
compositionstart
compositionupdate
dblclick
focusin
focusout
input
keydown
keyup
mousedown
mousemove
mouseout
mouseover
mouseup
scroll
select
wheel
keypress

h5新特征：异步、local/session storage、多线程
(1)新的语义标签和属性
(2)表单新特性

(3)视频和音频
(4)Canvas绘图
(5)SVG绘图
(6)地理定位
(7)拖放API
(8)WebWorker　创建线程
(9)WebStorage
(10)WebSocket　建立TCP连接

H5新事件：

1、针对window 对象触发的事件（应用到<body>标签）：

onafterprint：文档打印之后
onbeforeprint：文档打印之前
onbeforeunload：文档卸载之前
onerror：在错误发生时
onhaschange：当文档已经改变时
onmessage：在消息被触发时
onoffline：当文档离线时
ononline：当文档上线时
onpagehide：当窗口隐藏时
onpageshow：当窗口成为可见时
onpopstate：当窗口历史记录改变时
onredo：当文档执行撤销（redo）时
onresize：当浏览器窗口被调整大小时触发
onstorage：在 Web Storage 区域更新后
onundo：在文档执行 undo 时运行的脚本

2、form事件：

oncontextmenu：当上下文菜单被触发
onformchange：在表单改变时
onforminput：当表单获得用户输入时
oninput：当元素获得用户输入时
oninvalid：当元素无效时

3、mouse事件

ondrag：元素被拖动时
ondragend：在拖动操作末端运行
ondragenter：当元素已被拖动到有效拖动区域时
ondragleave：当元素离开有效拖动目标时
ondragover：当元素在有效拖放目标上正在被拖动时
ondragstart：在拖动操作开端时
ondrop：当被拖元素正在被拖放时
onmousewheel：当鼠标滚轮正在被滚动
onscroll：当元素滚动掉被滚动
4、media事件
oncanplay：当文件就绪可以开始播放时
oncanplaythrough：当媒介能够无需因缓冲二停止即可播放至结尾时
ondurationchange：当媒介长度改变时
onemptied：当发生故障并且文件突然不可用时
onended：当媒介已到达结尾时
onerror：当在文件加载期间发生错误时
onloadeddata：当媒介数据已加载时
onloadedmetadata：当元数据（比如分辨率和时长）被加载时
onloadstart：在文件开始加载且未实际加载任何数据前
onpause：当媒介被用户或程序暂停时
onplay：当媒介已就绪可以开始播放时
onplaying：当媒介已开始播放时
onprogress：当浏览器正在获取媒介数据时
onratechange：每当回放速率改变时
onreadystatechange：每当就绪状态改变时运行的脚本（就绪状态监测媒介数据的状态）
onseeked：当 seeking 属性设置为 false（指示定位已结束）时
onseeking：当 seeking 属性设置为 true（指示定位是活动的）时
onstalled：在浏览器不论何种原因未能取回媒介数据时
onsuspend：在媒介数据完全加载之前不论何种原因终止取回媒介数据时
ontimeupdate：当播放位置改变时（比如当用户快进到媒介中一个不同的位置时）
onvolumechange：每当音量改变时（包括将音量设置为静音）时

表单新特征：

(1)新的input type

(2)新的表单标签

(3)表单标签的新属性

新的input type：

HTML5之前已有的input type：

text、password、radio、checkbox、file、submit、reset、button、image、hidden

HTML5新增加的input type：

email：邮件输入域，在表单提交时提供简单的邮箱格式验证，并弹出一个提示窗口。基本上只能验证出需要有@。

url：URL地址输入，待表单提交时提供简单的URL地址格式验证，并弹出提示窗口。基本上只要有xxx:xxx就可以通过。

number：数字输入域。在表单提交时提供简单的数字格式验证，并弹出提示窗口。<input type=“number” min=“” max=“” step=“”>

tel：电话号码输入域，在手机浏览器中弹出数字输入键盘。

search：搜索输入域，在手机浏览器中右下角呈现搜索按键。

range：范围选择控件，帮助用户在一定范围内选择一个数字。<input type=“range” min=“” max=“” step=“”>

color：颜色选择控件，浏览器并未自己实现颜色选择框，而是使用操作系统自带的颜色选择控件。

date：日期选择器。FF没有实现

month：月份选择器，FF没有实现

week：星期选择器，FF没有实现

新的表单元素：

HTML5之前from有的标签，用于数据提交：

input、textarea、select/option、button

HTML新增表单元素，用于信息提交，不能用于数据提交：

datalist：数据列表，配合option使用，本身为不可见元素，为普通的input提供输入建议列表

<datalist id=“l”><option>XXX</option></datalist>

<input type=“text” list=“l”>

progress：进度条，未指定value属性则显示为“进行中”样式；若指定了

value（默认在0～1之间）就可以控制其现实的进度。<progress value=“0.5”>
</progress>

meter：刻度尺/度量衡，用红黄绿三色表示出一个数值所处的范围：不可接受/可以接受/最优范围。<meter min=“最小可能值” max=“最大的可能值” low=“合理的下限” high=“合理的上限” optimum=“最优值” value=“实际值“>
</meter>（如果optimum位于low和high之间，且value位于low和high之间，显示绿色；如果上述都不满足显示红色；如果满足一项，显示黄色。）

output：输出，用于描述表单中的计算结果，语意标签，样式与span无异。

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
  <input type="range" id="a" value="50">100
  +<input type="number" id="b" value="50">
  =<output name="x" for="a b"></output>
 </form>
```

表单元素新的属性

HTML5之前表单元素可用的属性：

id、class、title、style、type、name、value、checked、selected、disabled、readonly、

HTML5表单元素新增的属性：

autocomplete：**on**/off，自动补全，是否自动纪录之前提交的数据，以用于下一次输入建议。

placeholder：用于 在输入框中显示提示性文字，与value不同，不能被提交

autofocus：false/true，自动获得输入焦点

multiple：false/true，是否允许多个输入值，若声明了该属性，输入框中就允许输入用逗号分隔的多个值。

form：为一个元素指定form属性，值为某个表单的ID，则此输入域可以放到表单的外部。

－－－－－－－－上面五个是新的通用属性－－－－－－－－－－－－－

required：false/true，必需的/必填项。在表单提交时会验证是否有输入，没有输入则弹出提示消息。

maxlength：最大长度，在有输入的情况下，限定输入域中字符的个数。

minlength：最小长度，在有输入的情况下，限定输入域中字符的个数，不是HTML5标准属性，仅部分浏览器支持。

min：限定输入的数字的最小值

max：限定输入的数字的最大值
step：限定输入的数字的步长，与min属性连用
pattern：指定一个正则表达式，对输入进行验证。<input pattern="1[3578]\d{9}">（对电话的匹配）

HTML5地理定位：
HTML5 Geolocation API 用于获得用户的地理位置。
鉴于该特性可能侵犯用户的隐私，除非用户同意，否则用户位置信息是不可用的。
支持地理位置的API的浏览器会定义navigator.geolocation
navigator.geolocation.getCurrentPosition(function(postion){},function(error){})：获取用户当前位置
该方法的第一个参数用来返回地理位置。position.coords.latitude(纬度)，position.coords.longitude(经度)。
第二个参数用于处理错误，它规定当获取用户位置失败时运行的函数。
error.code有四个值，PERMISSIO_DENIED：用户不允许地理定位；POSITION_UNAVAILABLE：无法获取当前位置；TIMEOUT：操作超时；UNKNOWN_ERROR：未知错误。
navigator.geolocation.watchPosition()：获取并不断监视当前位置，一旦有更改就会触发指定函数
navigator.geolocation.clearWatch()：停止监听用户位置

HTML5拖放API：

HTML5 webWorker：
在以前的javascript中我们常用回调函数，来实现单线程异步。在H5中webworker解决了以前单线程性能。webwork是实现javascript真正的多线程机制。
webworker是运行在后台的javascript，独立于其他脚本，不会影响页面的性能。你可以继续做任何愿意做的事：点击、选取内容等等，而此时webworker在

后台运行。与setTimeout的单线程延迟异步不同，webworker是真正的多线程机制。

webworker不能做的事：

不能访问document对象

不能访问window对象

不能访问parent对象

Worker本质是一个线程，在UI主线程之外并发执行的线程；用于执行耗时的JS任务；缺陷：不能操作BOM和DOM，只能和UI主线程发消息。

通过postMessage()来发送数据，通过onmessage()来接收数据，无论是UI中的主线程还是Worker线程都可以用这两个方法来发送和接收数据。

```
var result = document.getElementById('result'),
    start = document.getElementById('start'),
    end = document.getElementById('end');

start.addEventListener('click',function (event) {
    if (typeof Worker !== 'undefined') {
        w = new Worker('demo_worker.js');
        w.onmessage = function(e){
            result.innerHTML = e.data;
        }
    }
});

end.addEventListener('click',function(event){
    w.terminate();
});
```

```
var i=0;

function timedCount()
{
    i=i+1;
    postMessage(i);
    setTimeout("timedCount()",500);
}

timedCount();
```

postMessage方法和onmessage事件不仅只有Worker线程拥有。

在页面中，一些Window对象拥有postMessage方法，例如parent（Window {stop: function, open: function, alert: function, confirm: function, prompt: function...}）、（iframe）.contentWindow。这些对象可以直接调用postMessage方法。可以

写：`document.getElementById('child').contentWindow.postMessage(docume`

`nt.getElementById('message').value,'/');`
同时，window对象上有onmessage事件。可以
写：`window.addEventListener('message', function(event){})`
postMessage会被onmessage检测到。
在这里，谁来调用postMessage，就代表发给哪个子窗口。

HTML5 WebStorage：
localStorage和sessionStorage
localStorage在本地永久性存储数据，除非显式将其删除或清空；
sessionStorage存储的数据只在会话期间有效，关闭浏览器则自动删除。
两个对象都有共同的API。
1、length:唯一的属性，只读，用来获取storage内的键值对数量。
2、key：根据index获取storage的键名
3、getItem：根据key获取storage内的对应value
4、setItem：为storage内添加键值对
5、removeItem：根据键名，删除键值对
6、clear：清空storage对象

HTML5 WebSocket：
WebSocket是建立了一个TCP连接，双向通信。Session只是保存了上次通信的所有相关信息，为了弥补http无状态连接的机制。
WebSocket 协议本质上是一个基于 TCP 的协议。为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息”Upgrade: WebSocket”表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。
检查浏览器是否支持WebSocket：

```
window.WebSocket = window.WebSocket || window.MozWebSocket;
if (!window.WebSocket){
   alert("WebSocket not supported by this browser");
   return;
```

}

创建WebSocket的过程：

1、构造：

var websocket = new WebSocket(“ws://127.0.0.1:8080/alarm/alarmServer");

第一个参数是请求地址，第二个参数选填，表示协议名

2、事件：

onopen：连接打开时　websocket.onopen = function (evt) { onOpen(evt) };

onmessage：接收到后台消息时　websocket.onmessage = function (evt) { onMessage(evt){console.log(evt.data)} };

onclose：后台关闭时　websocket.onclose = function (evt) { onClose(evt) };

onerror：出现异常时　websocket.onerror = function (evt) { onError(evt) {console.log(ev.data)} };

例如：

```
websocket.onmessage = function(evt){
   var data = evt.data;
}
```

3、方法：

send：WebSocket.send(data)

close：WebSocket.close(optional code, optional reason)

两栏布局，不用float

数组去重，不用遍历

1、ES6：

```
Array.prototype.uniq = function(){
return Array.from(new Set(this));
}
```

2、underscore

```
Array.prototype.uniq = function(){
return _.unique(this);
}
```

不用let，原生实现块级作用域

函数自执行

多重继承

1、多次superx.call(this)

2、对多个superx里的prototype循环，赋值给sub

jsonp能不能保证只访问某些js

用URL控制

bootstrap中的栅格的实现

@media来决定所占宽度，再对所占宽度等分成12份，利用 百分比

单页应用和多页应用

路由#

VUE、VUEX、VUE-Cli例子

分页：单页－显示区域为单独一个模块，只更新那个区域。（还会使用模版吗？）

多页－整个页面为一个模块，更新了整个页面。

都是在后端得到一共多少条数据，根据浏览器传递过去的（当前页码、每页几条）返回要显示的条目以及分页器的结构信息。

深拷贝的实现：用递归（学习内容－js－深拷贝的实现）