**一、选择题**

**1、HTML5新特性**

1. 用于绘画的 canvas 元素
2. 用于媒介回放的 video 和 audio 元素
3. 对本地离线存储的更好的支持，localStorage/sessionStorage
4. 新的特殊内容元素，比如 article、footer、header、nav、section
5. 新的表单控件，比如 calendar、date、time、email、url、search
6. 文档类型doctype
7. 引入css、js文件不用再加type属性了

**2、CSS3新特性**

1. CSS3 选择器:tbody: nth-child(even), nth-child(odd),E:last-child 等等
2. @Font-face 特性：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
3. Word-wrap & Text-overflow 样式
4. 文字渲染（Text-decoration）
5. CSS3 的多列布局（multi-column layout）
6. 边框和颜色（color, border）
7. CSS3 的渐变效果（Gradient）
8. CSS3 的阴影（Shadow）和反射（Reflect）效果
9. CSS3 的背景效果
10. CSS3 的 Transitions, Transforms 和 Animation
11. 在CSS3中唯一引入的伪元素是::selection

**3、JS的基本数据类型**

Undefined、Null、Boolean、Number、String

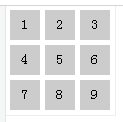
**二、简答题**

**1、cookies、sessionStorage、localStorage的区别**

共同点：都是保存在浏览器端，且同源的。

区别：

1. cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递。而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下。
2. 存储大小限制也不同，cookie数据不能超过4k，同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如会话标识。sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。
3. 数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭。
4. 作用域不同，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的。
5. Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。
6. Web Storage 的 api 接口使用更方便。
7. **写一个九宫格布局**



css:

\*{list-style: none;padding: 0;margin: 0;}

ul{

width: 105px;

overflow: hidden;

border:1px solid #eee;

padding: 0 0 5px 5px;

}

li{

width: 30px;

height: 30px;

text-align: center;;

line-height: 30px;

background-color: #ccc;

float: left;

margin-right: 5px;

margin-top: 5px;

}

html:  
 <ul>

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

<li>5</li>

<li>6</li>

<li>7</li>

<li>8</li>

<li>9</li>

</ul>

1. **把var str = "www.zhenai.com" 变成 var strResult = "moc.ianehz.www"**

var strResult = str.split("").reverse().join("")

**4、如果需要手动写动画，你认为最小时的间隔是多久，为什么？**

多数显示器默认频率是60Hz，即1秒刷新60次，所以理论上最小间隔为1/60＊1000ms ＝ 16.7ms

**5、你如何对网站的文件和资源进行优化（可列出用到的自动化工具）**

1. 文件合并（目的是减少http请求）：Web性能优化最佳实践中最重要的一条是减少HTTP 请求，它也是YSlow中比重最大的一条规 则。减少HTTP请求的方案主要有合并JavaScript和CSS文件、CSS Sprites、图像映射 （Image Map）和使用Data URI来编码图片。
2. 文件压缩：目的是直接减少文件下载的体积；常用的工具是YUI Compressor。参考http://www.cnblogs.com/Darren\_code/archive/2011/12/31/property.html
3. 使用 CDN （内容分发网络）来托管资源；"其基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。"
4. 缓存的使用（并且多个域名来提供缓存）
5. GZIP 压缩你的 JS 和 CSS 文件
6. 注意控制Cookie大小和污染
7. **ajax请求的时候get和post的区别，以及如何解决跨域问题**

get和post的区别:

1.使用Get请求时,参数在URL中显示,而使用Post方式,则不会显示出来

2.使用Get请求发送数据量小,Post请求发送数据量大

3.get安全性非常低，post安全性较高。

如何解决跨域问题:

1. JSONP(只支持GET请求，不支持POST请求)
2. 添加响应头，允许跨域

addHeader(‘Access-Control-Allow-Origin:\*’);//允许所有来源访问

addHeader(‘Access-Control-Allow-Method:POST,GET’);//允许访问的方式

1. 代理的方式
2. **前端优化、用过的框架、自己的作品、自己的优势**前端优化：

一）内容层面

1、DNS解析优化（DNS缓存、减少DNS查找、keep-alive、适当的主机域名）

2、避免重定向（/还是需要的）

3、切分到多个域名

4、杜绝404

二）网络传输阶段

1、减少传输过程中实体的大小

1）缓存

2）cookie优化

3）文件压缩（Accept-Encoding：g-zip）

2、减少请求的次数

1）文件适当的合并

2）雪碧图

3、异步加载（并发,requirejs）

4、预加载、延后加载、按需加载

三）渲染阶段

1、js放底部，css放顶部

2、减少重绘和回流

3、合理使用Viewport 等meta头部

4、减少dom节点

5、BigPipe

四）脚本执行阶段

1、缓存节点，尽量减少节点的查找

2、减少节点的操作（innerHTML）

3、避免无谓的循环，break、continue、return的适当使用

4、事件委托

用过的框架：vue/jq/bootstrap/element/vue-bootstrap/react-native/swiper/animate

**8、如何判断一个对象是空对象**

JSON.stringify(obj)===”{}”

**9、如何在跨域请求的时候让服务端能拿到你的cookie信息**

前端要配置一个ajax参数：xhrFields:{withCredentials:true}

后台要在响应头中绑定"Access-Control-ALLOW-Credentials",值为"true"