1、 什么是盒子模型

CSS 盒子模型就是在网页设计中经常用到的 CSS 技术所使用的一种思维模型。

网页设计中常听的属性名：内容(content)、填充(padding)、边框(border)、边界(margin)， CSS 盒子模式都具备这些属性。

这些属性可以把它转移到日常生活中的盒子（箱子）上来理解，日常生活中所见的盒子也就是能装东西的一种箱子，也具有这些属性，所以叫它盒子模式。

2、 行内元素有哪些，块元素有哪些，空元素有哪些？

（1） 行内元素有：a span img input select strong

（2） 块级元素有： div ul ol li dl dt dd h1 h2 h4...p

（3） 常见的空元素
<hr><img><input><link><meta>

3、 css 垂直水平居中

宽度固定

.parent { position: relative; } .child { width: 300px; height: 100px; padding: 20px; position: absolute; top: 50%; left: 50%; margin: -70px 0 0 -170px; }

宽度不固定

.parent { position: relative; } .child { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); }

弹性布局实现

.parent { display: flex; justify-content: center; align-items: center; }

{ position: absolute; margin: auto; top: 0; left: 0; right: 0; bottom: 0; }

4、 什么是 css Hack

CSS hack 是通过在 CSS 样式中加入一些特殊的符号，让不同的浏览器识别不同的符号（什么样的浏览器识别什么样的符号是有标准的，CSS hack 就是让你记住这个标准），以达到应用不同的 CSS 样式的目的，

5、 简述同步和异步的区别？

同步，是所有的操作都做完，才返回给用户结果。即写完数据库之后，再响应用户，用户体验不好。

异步，不用等所有操作等做完，就相应用户请求。即先响应用户请求，然后慢慢去写数据库，用户体验较好

6、 js 怎么样添加、移除、移动、复制、创建和查找节点？

appendChild() 添加
removeChild() 删除
replaceChild() 替换
insertBefore() 之前插入
insertAfter() 之后插入
cloneNode() 复制
createElement()创建
getElementBy 查找

7、 var val=”smtg”;console.log(“Value is”+(val===”smtg”)?’Something’:’Nothing’)
输出什么？
输出 Something 优先级问题 如果没有小括号 就输出 Nothing

8、var name='World';

```
(function (){
    if(name==="World"){
        var name="Jack";
        console.log("jack");
    }else{
        console.log("Rose");
    }
})();
```

打印出 Rose 因为 var 声明提前
如果条件变成 typeOf name===”undefine”; 就打印 Jack 因为声明提前

8、 如何消除一个数组里面的重复元素
循环、for……in 循环 、var set1 = Array.from(new Set([1,1,2,2,33,'33',44,'44'])) es6 新属性

9、[]==[]、'5'+3、'5'-3 分别输出什么？

false、53、2

[] 数组，在 Javascript 中是对象，对象使用 == 比较都是比较的引用。简单的说，就是，如果是同一个对象，就相等，如果不是同一个对象，就不等。每次使用 [] 都是新建一个数组对象，所以 [] == [] 这个语句里建了两个数据对象，它们不等

10、请描述一下 cookie、sessionStorage、localStorage 的区别？

1.存储大小

- cookie 数据大小不能超过 4k。
- 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。

2.有效时间

- localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；
- sessionStorage 数据在当前浏览器窗口关闭后自动删除。
- cookie 设置的 cookie 过期时间之前一直有效，即使窗口或浏览器关闭

3. 数据与服务器之间的交互方式

- cookie 的数据会自动的传递到服务器，服务器端也可以写 cookie 到客户端
- sessionStorage 和 localStorage 不会自动把数据发给服务器，仅在本地保存

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：在所有同源窗口都是共享的；
- cookie：也是在所有同源窗口中共享的
- webStorage 支持事件通知机制，可以将数据更新的通知发生给监听者
- webStorage 的 API 借口使用更方便 。setItem getItem clearItem

11、创建方法打印重复的字符串。

```
String.prototype.native=function(num){
        return this.repeat(num);
}
console.log("abc".native(3));
```

12、谈谈你对前端性能优化的理解？

1、减少 http 请求，合理设置 HTTP 缓存

2、使用浏览器缓存

3、启用压缩

4、CSS Sprites

5、LazyLoad Images

6、CSS 放在页面最上部，javascript 放在页面最下面

7、异步请求 Callback（就是将一些行为样式提取出来，慢慢的加载信息的内容）

8、减少 cookie 传输

13、前端 mv*框架的意义；

jQuery 的思维方式是：以 DOM 操作为中心

MV*框架的思维方式是：以模型为中心，DOM 操作只是附

如果是页面型产品，多数确实不太需要它，因为页面中的 JavaScript 代码，处理交互的绝对远远超过处理模型的，但是如果是应用软件类产品，这就太需要了。

14、线程和进程的区别

所以运行某个软件，相当于开了一个进程。在这个软件运行的过程里（在这个进程里），多个工作支撑的完成 QQ 的运行，那么这“多个工作”分别有一个线程。

所以一个进程管着多个线程。

通俗的讲：“进程是爹妈，管着众多的线程儿子”...

15、前端页面有哪三层构成？分别是什么 ？作用是什么？

结构层 html 搭建文档结构

表示层 css 设置文档呈现效果

行为层 js 和 dom 实现文档行为

16、减少页面请求时间的方法

1.减少 Http 的请求

2. 服务器开启 gzip 压缩

3.css 样式的定义放置在文件头部 Javascript 脚本放在文件末尾

4 压缩 Javascript、CSS 代码

5. Ajax 采用缓存调用

6. 尽可能减少 DOM 元素

7．使用多域名负载网页内的多个文件、图片，节约主域名的连接数，从而提高客户端网络带宽的利用率，优化页面响应。

17、获取 js 函数参数

Arguments

18、js 代码优化

1.尽量使用原生

2.减少 dom 操作

3.计时器的选择

4．减少循环

5.if 和 switch 的选择

19、apply, call 和 bind 有什么区别?

参考答案：三者都可以把一个函数应用到其他对象上，注意不是自身对象．apply,call 是直接执行函数调用，bind 是绑定，执行需要再次调用．apply 和 call 的区别是 apply 接受数组作为参数，而 call 是接受逗号分隔的无限多个参数列表，

20、闭包——函数，更多字嵌套函数的父子自我引用关系

21、

```
let bar=foo.call();
console.log(bar);
function foo(){
    return "Hello World".split("").sort().join("");
}
```

22、js 清除浏览器缓存

meta 方法

jquery ajax 清除浏览器缓存

用随机数、随机时间，随机数也是避免缓存的一种很不错的方法！

23、

```
var val;
console.log('Value is'+(val !='0')?1:2);     1
```

24、

```
function test(){
    var n=666;
    function add(){
        n++;
        console.log(n);
    }
    return {n:n,add:add};
}
var r1=test();
var r2=test();
r1.add();   667
r1.add();   668
console.log(r1.n);666
```

r2.add();667

25、call 和 apply 和 bind 的区别

Call 和 apply 都是自动执行　bind 需要手动调用，call 的参数是逗号隔开的 apply 参数是数组格式。

26、对于设计模式的理解

Mvc　mvp　mvvm

27、webpack 配置，开发环境，运行环境，插件

28、<keep-alive>是 Vue 的内置组件，能在组件切换过程中将状态保留在内存中，防止重复渲染 DOM

29、前端如何实现模块化？

Es5 之前，requireJs　<script data-main="js/main" src="xxx/xxx/require.js"></script>　使用 RequireJS，你只需要引入一个 require.js 即可。你的页面上只需要通过\<script\>标签引入这一个 js 即可。然后这个页面的所有业务逻辑只需要在 main.js 里面写.

es6 原生支持模块化了，通过 import 导入模块,export 导出模块。

30、AMD 和 CMD 有什么区别？

1、区域以来的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.
2、CMD 推崇依赖就近，AMD 推崇依赖前置。

```
//CMD

define(function(require,exports,module){

    var a = require("./a");

    a.doSomethis();

    var b = require("./b")//依赖可以就近书写

    b.doSomething()

})

//AMD
```

```
define(['./a,./b'],function(a,b){//依赖必须一开始就写好
    a.dosomething()
    b.dosomething()
      })
```