一、dianhua

1. CSS盒模型

## **基本概念**

在 CSS 盒子模型 (Box Model) 规定了元素处理元素的几种方式： **内容、内边距、边框、外边距**。

在 CSS的**标准盒子模型**中，**width 和 height 指的是内容区域**的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。**IE盒子模型**中，**width 和 height 指的是内容区域+border+padding**的宽度和高度。

## **css如何设置两种模型**

这里用到了CSS3 的属性 box-sizing

/\* 标准模型 \*/

box-sizing:content-box;

/\*IE模型\*/

box-sizing:border-box;

## **JS获取宽高**

通过JS获取盒模型对应的宽和高，有以下几种方法：

为了方便书写，以下用dom来表示获取的HTML的节点。

1.  dom.style.width/height

　　这种方式只能取到dom元素内联样式所设置的宽高，也就是说如果该节点的样式是在style标签中或外联的CSS文件中设置的话，通过这种方法是获取不到dom的宽高的。

 2. dom.currentStyle.width/height

　　这种方式获取的是在页面渲染完成后的结果，就是说不管是哪种方式设置的样式，都能获取到。

　　但这种方式只有IE浏览器支持。

 3. window.getComputedStyle(dom).width/height

　　这种方式的原理和2是一样的，这个可以兼容更多的浏览器，通用性好一些。

 4. dom.getBoundingClientRect().width/height

这种方式是根据元素在视窗中的绝对位置来获取宽高的

空边框盒（译者注：没有内容的边框）会被忽略。如果所有的元素边框都是空边框，那么这个矩形给该元素返回的 width、height 值为0，left、top值为第一个css盒子（按内容顺序）的top-left值。

当计算边界矩形时，会考虑视口区域（或其他可滚动元素）内的滚动操作，也就是说，当滚动位置发生了改变，top和left属性值就会随之立即发生变化（因此，它们的值是相对于视口的，而不是绝对的）。如果你需要获得相对于整个网页左上角定位的属性值，那么只要给top、left属性值加上当前的滚动位置（通过window.scrollX和window.scrollY），这样就可以获取与当前的滚动位置无关的值。

 5.dom.offsetWidth/offsetHeight

　　这个就没什么好说的了，最常用的，也是兼容最好的。

1. box-sizing的应用场

### 1）常规的布局思维是我定好整体的宽高之后，里面元素的整体尺寸会确定下来，后面才是考虑这个元素的内边距以及边框，如果我已经确定好了大体的布局，再去微调边框而导致了整体布局的崩塌，这显然是不科学的，所以这也是我极力推崇前端布局使用box-sizing：border;的原因。

### 2）box-sizing 布局三栏目案例

div{

height:700px;

float:left;

}div.left{

width:25%;

background:red;

}div.cent{

width:50%;

box-sizing:border-box;/\*可以改变元素以使其宽度包含填充\*/

/\* 现在整个元素,包括填充在内,占页面总宽度的50%,所以元素的组合宽度为100%,这全程它们很好地适应于它们的容器.\*/

background:yellow;

padding:0 20px;/\*加了这个会使盒子内容溢出 但是box-sizing很好的自适应了\*/

}div.right{

width:25%;

background:blue;

}

<div class="left"></div>

<div class="cent"></div>

<div class="right"></div>

1. 弹性flex布局
2. 未知宽高元素上下左右垂直居中
3. 原型链、对象、构造函数之间的联系
4. DOM事件绑定的几种方式
5. 项目中用到的技术栈，以及觉得得意和出色的点，以及头疼点，怎么解决的
6. 了解过http2.0,websocket,https？说一下你的理解以及所了解的特性
7. Jishu
8. webpack的入口文件怎么配置，多个入口怎么分割
9. Babel插件transform-runtime以及stage-2，它们的作用
10. Webpac.optimize.UglifyJsPlugin这个插件，有没有压缩速度慢？什么办法提升速度
11. http协议，200和304的理解和区别
12. DOM事件中target和currentTarget的区别
13. 平时怎么解决跨域，以及JSONP的原理和实现，以及cors怎么设置
14. 深拷贝的实现原理
15. 项目中觉得可以改进的地方以及做的很优秀的地方
16. jishu2
17. 自己写过webpack的loader？原理？
18. 有没有去研究webpack的原理和机制，怎么实现的
19. Babel把ES6转成ES5或ES3之类的原理？
20. Git大型项目的团队合作，以及持续集成
21. 什么是函数柯里化？JS的API有哪些用到了函数柯里化的实现
22. ES6箭头函数this问题，以及拓展运算符
23. JS模块化Commonjs、UMD、CMD规范的了解，以及ES6的模块化跟其他几种的区别，以及出现的意义
24. Vue实现双向数据绑定的原理，以及vue.js和react.js异同点，如果让你选框架，你怎么权衡这两个框架，分析一下
25. 写博客，草稿的交互细节以及实现原理
26. jishu3
27. 自我介绍，项目技术栈，项目遇到的问题
28. 一个业务场景，面对产品不断迭代，以及需求变动该怎么应对，具体技术方案实现
29. 代码的可维护性和可扩展性
30. 怎么获取一个元素到视图顶部的距离
31. getBoundingClientRect获取的top和offsetTop获取的top区别
32. 事件委托
33. 百度的服务不想让阿里使用，如何识别到是阿里的请求，然后跳转到404或者拒绝服务之类的
34. http协议头Referer
35. 二分查找的时间复杂度怎么求，是多少
36. XSS是什么，攻击原理，怎么预防
37. 跨站脚本攻
38. 线性存储结构和链式存储结构有什么区别？以及优缺点
39. 移动端日历、PC端日历以及桌面日历的一些不同和需要注意的地方
40. 用最简单的方式实现数组去重
41. ES6实现

[...new Set([])]

1. ES5实现

[].filter(function(item,index,array){

return index===array.indexOf(item)

})

1. 怎么实现草稿，多终端同步，以及冲突问题