面试准备：页面布局、CSS盒模型、DOM事件、HTTP协议、面向对象、原型链、通信、安全、算法。



一、页面布局

题目：假设高度已知，请写出三栏布局，其中左栏、右栏各为300px，中间自适应。

(常见方案：浮动布局、绝对布局、flexbox布局、表格布局、网格布局)

延伸：1.这5种方案各自有什么优点和缺点？

2.如果高度未知，使每栏高度随内容自动撑开，且高度一致，该5种方案还有哪几个还可以适用？哪几个已经不能用了？

3.这5种方案的兼容性如何？如果在业务开发中给你选择，你的最优方案是哪种？

1.答案：

(1)浮动布局 缺点：浮动后元素脱离文档流，如果处理不好会带来很多问题。

优点：如果做好清除浮动及处理好浮动周边的元素的关系，兼容性比较好。

(2)绝对布局 缺点：布局已经脱离文档流，意味着下面的子元素也脱离了文档流，这导致有效性及可使用性比较差。

优点：快捷，如果配合js使用，非常快，也不容易出问题。

(3)flex布局 为解决上述两种方案产生的不足而设计的，相对来说比较完美，在移动端上基本都是flex布局。

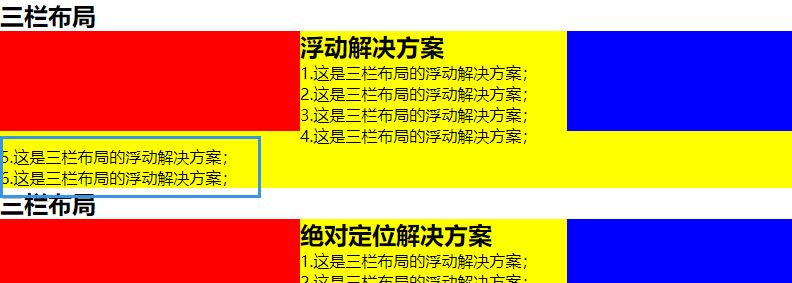
(4)表格布局 缺点：比较麻烦；操作比较繁琐；对SEO不够友好；当其中一个单元格的高度超出的时候，其他两个单元格也需要跟着相应改变，在某些业务场景中是不需要自动增高的。

优点：兼容性比较好，如果在PC上需要兼容比较低版本的IE8，flex布局不能实现，表格布局可以实现

(5)网格布局 网格布局可以做很多复杂的事情，但是代码量要简化很多，网格布局也是一种新的技术

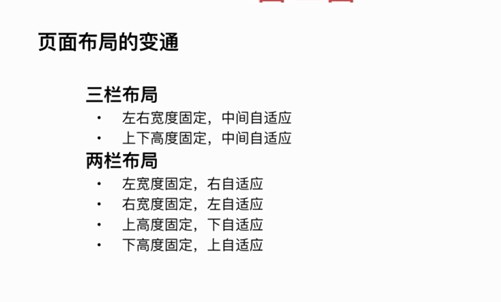
2.答案：flex布局和表格布局还适用。其他三种均超出。

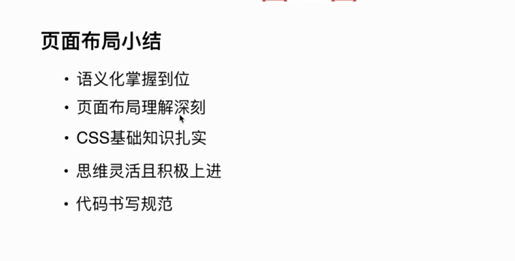
延伸：(1)浮动布局中，为什么中间部分的内容超出后会在左栏下面显示？



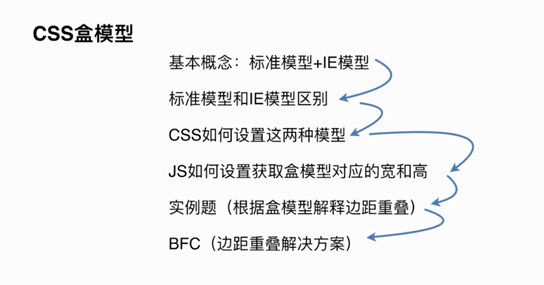
原因：这是因为浮动的基本原理。中间的内容向左浮动的时候，被左侧的块挡住了，所以文字在中间显示；当内容超出以后，发现左侧没有遮挡物，所以文字会在左侧的块的下边显示。

延伸：如果需要让中间的内容即使超出后也只在中间部分展示，不能跑到左右侧下方，如何实现？ 答案：创建BFC



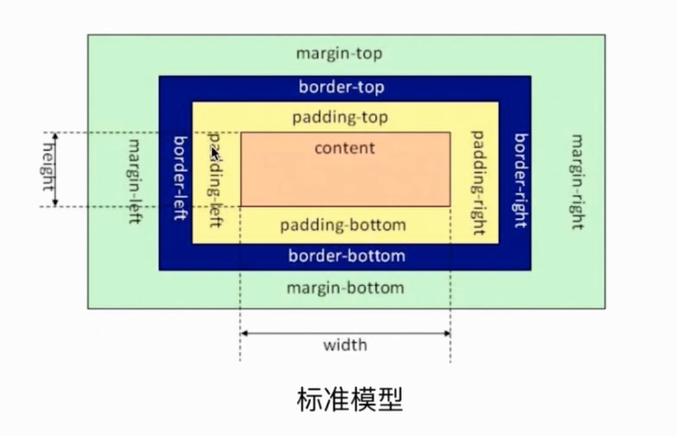


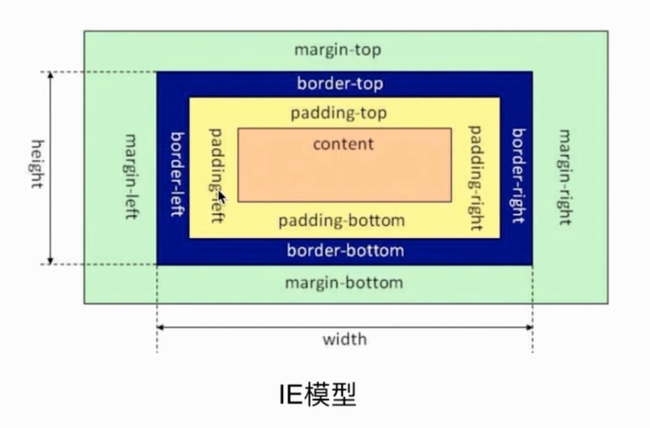
二、CSS盒模型



1.CSS模型有两种：标准模型和IE模型。

2.标准模型和IE模型区别？





3.CSS如何设置这两种模式？

box-sizing: content-box; 标准模式

box-sizing: border-box; IE模式

浏览器默认的是content-box，即标准模式

4.JS如何设置获取盒模型对应的宽和高

dom.style.width/height：只能取到内联样式的宽和高

dom.currentStyle.width/height：拿到的是渲染后的宽和高，相对比较准确，但是只有IE支持

window.getComputedStyle(dom).width/height：通用性更好，兼容火狐谷歌，不兼容IE

dom.getBoundingClientRect().width/height：也能拿到元素真实的宽和高，适用场景是计算元素的绝对位置。这个绝对位置是针对屏幕左上角(0,0)的绝对位置，该方法拿到的是4个值，left/top/width/height。

5.实例题(边距重叠)

(1)BFC的基本概念：块级格式化上下文

(2)BFC的原理(BFC的渲染规则)：

a.在BFC这个元素的垂直方向上的边距会发生重叠；

b.BFC的区域不会与浮动元素的box重叠；

c.BFC在页面上是一个独立的容器，外面的元素不会影响它里面的元素；反过来，里面的元素也不会影响外面的元素；

d.计算BFC高度的时候，其内部的浮动元素也会参与计算。

(3)如何创建BFC？

a.float值不为none；

b.position的值不为static或者relative；

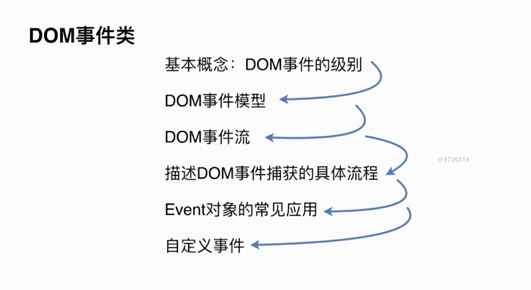
c.display的值是inline-block、table-cell、flex、table-caption或者inline-flex；

d.overflow不为visible。

(4)BFC的使用场景有哪些？

解决边距重叠、清除浮动

三、DOM事件



1.DOM事件的级别：实际上是DOM标准定义的级别。

DOM0 element.onclick = function(){}

DOM2 element.addEventListener('click', function(){}, false)

DOM3 element.addEventListener('keyup', function(){}, false)

说明：DOM1设计的时候，没有设计跟事件相关的东西，所以事件级别会直接从DOM0跳到DOM2。

2.DOM事件模型：捕获和冒泡

3.DOM事件流

浏览器在为当前这个页面与用户做交互的过程中，例如点击鼠标左键，这个动作是怎么传递到页面上，这就是事件流。

一个完整的事件流分三个阶段：捕获阶段，目标阶段，冒泡阶段。

4.描述DOM事件捕获的具体流程

window => document => html => body => ... => 目标元素

JS中获取html节点：document.documentElement

JS中获取body节点：document.body

5.Event对象的常见应用

event.preventDefault()

event.stopPropagation()

event.stopImmediatePropagation()

event.currentTarget

event.target

6.自定义事件(模拟事件)

使用场景：有一个按钮，不是一个常规的click事件，给这个按钮增加一个自己的事件，并且可以在其他地方触发这个事件，而不是用回调的方式触发，这个时候就可以用自定义事件。

var eve = new Event('custome'); // 将eve当成普通事件对象

ev.addEventListener('custome', function(){

console.log('custome');

});

ev.dispatchEvent(eve);

此时就完成了一个自定义事件和自定义触发事件的过程

四、HTTP协议类

1.HTTP协议的主要特点

简单快速：每个资源URI是固定的，想访问某个资源只需要输入URI就可以了。

灵活：在HTTP协议头部会有一个数据类型，通过一个HTTP协议就可以完成不同数据类型的传输。

无连接：连接一次就会断掉，不会保持连接。

无状态：客户端和服务端是两种身份，客户端下次请求的资源即使和上次一样，服务端也不会记住是哪个用户请求了哪些资源。

2.HTTP报文的组成部分

请求报文：请求行、请求头、空行、请求体。

请求行：HTTP方法、页面地址（请求URL）、HTTP协议、HTTP版本

请求头：一些key value值，来告诉服务端客户端需要哪些内容，要注意哪些类型。

空行：告诉服务端请求头到此结束，后面的内容是请求体。

请求体：

响应报文：状态行、响应头、空行、响应体。

响应报文的组成部分描述基本与请求报文一致。

3.HTTP方法

GET 获取资源

POST 传输资源

PUT 更新资源

DELETE 删除资源

HEAD 获得报文首部

4.POST和GET的区别

5.HTTP状态码

6.什么是持久连接

7.什么是管线化