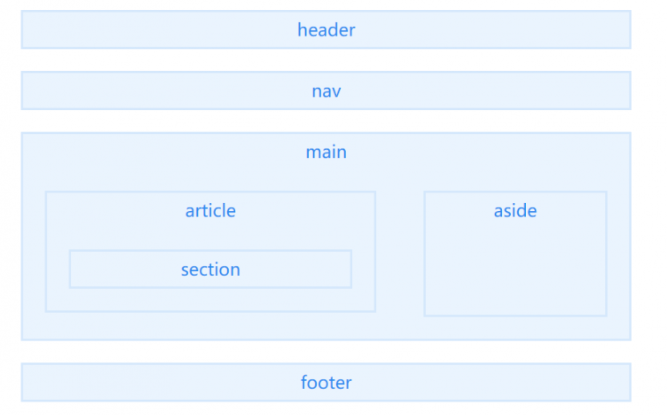
1. HTML语义化：根据内容结构化（内容语义化），选择合适的标签（代码语义化），便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好的解析
2. 为什么语义化：

* 在没有CSS的情况下，页面也能呈现出很好的内容结构、代码结构
* 用户体验：如title、alt、label等
* 有利于SEO（搜索引擎优化）：和搜索引擎建立良好的沟通，有利于爬虫抓取更多有效信息
* 方便其他设备（如盲人阅读器）解析渲染网页
* 便于团队开发和维护，语义化更具可读性，减少差异化

1. 语义化标签：



1. 弹性盒模型：当页面需要适应不同屏幕大小以及设备类型时确保元素拥有恰当的行为的布局方式

* 引入弹性盒模型目的：提供一种更加有效的方式对一个容器中的子元素进行排列、对齐、分配空白空间
* 弹性盒子由弹性容器和弹性子元素组成
* 弹性容器设置display属性值为flex或inline-flex将其定义为弹性容器
* 弹性盒子只定义了弹性子元素如何在弹性容器内部局

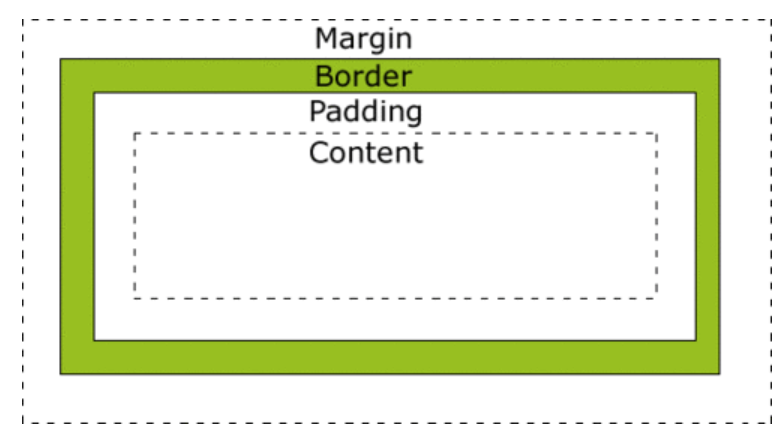
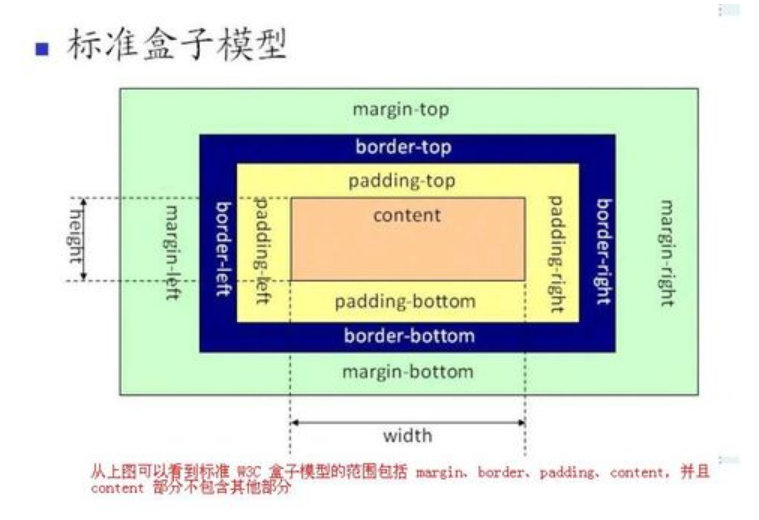
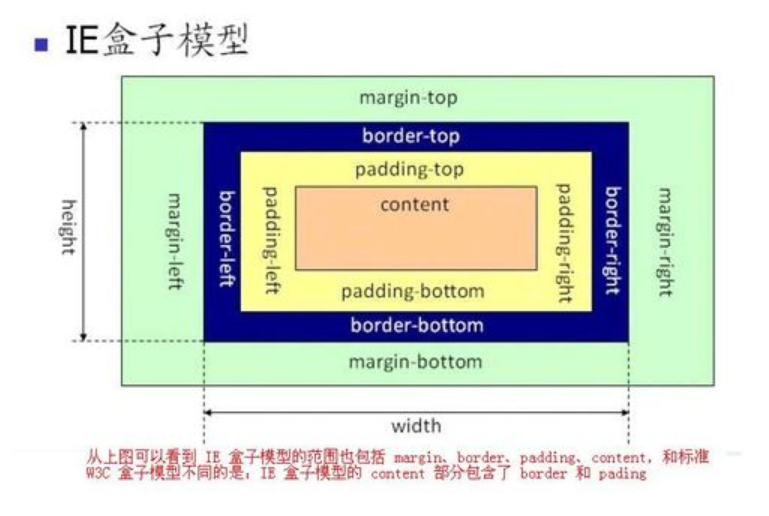
1. 响应式布局：根据设备环境及用户行为调整布局

实现方式：css3媒体查询（检测当前设备，来确定应用哪一个CSS样式表）、

媒体查询：@media 设备类型 and （设备特性） { 样式代码 }

@media screen and (max-width:960px) and(min-width:720px){}  
流式网格布局（对页面布局元素使用相对CSS比例而不是绝对大小）

1. CSS盒子模型：

* 
* W3C盒子模型（标准盒模型）和IE盒子模型（怪异盒模型）
* 
* 根据 W3C 的规范，元素内容占据的空间是由 width 属性设置的，而内容周围的 padding 和 border 值是另外计算的；即在标准模式下的盒模型，盒子实际内容（content）的width/height=我们设置的width/height;盒子总宽度/高度=width/height+padding+border+margin。
* 
* 在该模式下，浏览器的 width 属性不是内容的宽度，而是内容、内边距和边框的宽度的总和；即在怪异模式下的盒模型，盒子的（content）宽度+内边距padding+边框border宽度=我们设置的width(height也是如此)，盒子总宽度/高度=width/height + margin = 内容区宽度/高度 + padding + border + margin。
* box-sizing 属性允许您以特定的方式定义匹配某个区域的特定元素。
* box-sizing属性可以指定盒子模型种类，content-box指定盒子模型为W3C（标准盒模型），border-box为IE盒子模型（怪异盒模型）。

1. 取消两个span之间空格

<div class="span1">

<span>你好</span>

<span>你好</span>

</div>

.span1 span{

float: left;

}

1. var、let、const

* var无块作用域，可能造成变量污染；var可能造成变量的非期望共享
* let只在let所在的代码块{}内有效；let可有效避免变量共享缺陷
* const声明常量，也常用来声明不变的函数；const一旦声明就必须赋值；const作用域同let；const指向的对象引用不可变，但其属性是可变的
* var会状态提升，let、const不会
* let、const特性：暂时性死区特性  
  -只要块级作用域内存在let，它所声明的变量就“绑定”在这个区域，不再受外部影响  
  - let对这个块从一开始就形成了封闭的作用域，凡是在声明之前使用该变量，就会报错

1. 前后端分离

* 交互形式：后端只需负责按约定的数据格式向前端提供可调用的API服务；前后端通过HTTP请求交互；前端获取数据后，进行页面的组装和渲染，最终返回给浏览器
* 代码组织方式：前后端代码库分离，前端代码中有可以进行Mock测试(通过构造虚拟测试对象以简化测试环境的方法)的伪后端，能支持前端的独立开发和测试。而后端代码中除了功能实现外，还有着详细的测试用例，以保证**API**的可用性，降低集成风险。
* 开发模式：前端工程师只需要编写**HTML、CSS、JS**等前端资源，然后通 过HTTP请求调用后端提供的服务即可。除了开发期的分离，在运行期前后端资源也会进行分离部署。
* 数据接口规范过程：在开发期间前后端共同商定好**数据接口**的交互形式和数据格式。然后实现前后端的并行开发，其中前端工程师再开发完成之后可以独自进行mock测试，而后端也可以使用接口测试平台进行接口自测，然后前后端一起进行**功能联调**并**校验格式**，最终进行自动化测试。

# JS复习

1. JS是弱类型（动态更改变量类型）、动态类型（不用给变量指定特定数据类型）语言
2. ES5没有块作用域、函数式编程、闭包、基于原型链的继承方式、动态添加属性
3. JS数据类型：  
   基本（原始）类型：Number、String、Boolean、Undefined、Null  
   引用（对象）类型：Object（Array、Function、Date、Error等）
4. JS严格模式：

* 严格模式下全局变量需显式声明
* 一般严格模式下函数中this为undefined，非严格模式为全局变量
* 严格模式下禁止删除不可改变的属性和未定义的变量
* 禁止函数参数重名

1. arguments.length是实参个数；  
   函数名.length是形参个数
2. 函数定义3种方式：  
   函数声明（函数名）function add(a,b){}

函数表达式（可以是没有函数名的匿名函数）var add=function(a,b){}  
Function构造函数实例化 var add=new Function(‘a’,’b’,’return a+b’)

1. 函数调用4种：  
   直接调用、obj.fun()、objA.foo.call(objB)、new Person(‘Jack’)
2. ES5采用函数级作用域，没有块级作用域

* 无块级作用域问题：变量污染、变量共享
* 解决：IIFE:立即执行表达式 (function( ){ }( ))  
   建立函数作用域，解决作用域缺陷问题