前端的预处理器

less

sass

```
$bg-color: red;
.box {
  background: $bg-color;
}
```

stylus

```
$bg-color = 'red'
.box
  background $bg-color
```

sass里面的函数声明

@function funNanme（）{}

$变量

Less用@来定义变量

盒模型差异

IE盒模型：margin 、 content（包含border、padding）

W3C盒模型： margin 、border、 padding、 content

CSS3中的box-sizing的属性就是为了这两种模式

border-box（IE盒模型）和content-box（W3C）

CSS清除浮动的几种方法（至少两种）

- 浮动元素末尾添加一个空的标签例如 ，其他标签br等亦可；
- 通过设置父元素overflow值设置为hidden/auto(在支持BFC的浏览器触发BFC)；在IE6,IE7中还需要触发 hasLayout ，例如 zoom：1
- 万能清除法

```
clearfix:after {content:"."; display:block; height:0; visibility:hidden; clear:both; }
clearfix { *zoom:1; }
```

父元素也设置浮动

父元素设置display:table

使用 br标签和其自身的 html属性

<br clear="all" />

bootstrap12删格系统的不同终端尺寸

- 1200. 992. 768

for in 和 for of的区别

for in 遍历对象

for of 遍历数组

for...in：主要用于遍历普通对象，也可以遍历数组

forEach：没有返回值，替换for循环

for of 循环（遍历的最好方法）：Es6 中新增的语句，用来替代 for in 和 forEach，可以遍历 Arrays（数组）, Strings字符串

一行代码去重

```
let array = Array.from(new Set([1, 1, 1, 2, 3, 2, 4]));
console.log(array);
// => [1, 2, 3, 4]
```

冒泡排序

```
//1.冒泡排序
                        //冒泡排序算法的原理如下：
                        //1比较相邻的元素。如果第一个比第二个大，就交换他们两个。
                        //2对每一对相邻元素做同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。
                        //3针对所有的元素重复以上的步骤，除了最后一个。
                        //4持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。
function bubblesort(arr){
                                if(Array.isArray(arr)){
                                        for(var i=0;i<arr.length-1;i++){//次数/轮数
                                                for(var j=0;j<arr.length-i-1;j++){//相邻的元素两两比较。
                                                        if(arr[j]>arr[j+1]){
                                                                var temp=arr[j];
                                                                arr[j]=arr[j+1];
                                                                arr[j+1]=temp;
```

```
                                        }
                                    }
                                }
                                return arr;
                            }else{
                                //alert('请输入数组');
                                throw new Error('你输入的不是一个数组');//利用错误对象抛错
                            }
                        }
    console.log(bubblesort(12,5,896,35,4000,-450,-6,7,2,100,48,35,4));
```

谈谈This对象的理解

this 是 javascript 里面的关键字,存在于函数内部,
this 指向有一个总的原则,谁调用这个函数,this 指向谁,

根据函数调用的方式不同 this 的值也不同:

1.全局作用域中，this 指向 window
2.以函数（普通函数，定时器，回调函数）的形式调用，this 指向 window
3.以方法（obj）的形式调用，this 指向调用方法的对象
4.以构造函数（new）的形式调用，this 指向新创建的那个对象
5.事件处理函数的 this 指向当前操作元素对象
6.箭头函数 this 指向定义时的对象
7.严格模式下 this 指向 undefined
8.使用 call 和 apply、blind 调用的函数，可以改变 this 的指向，第一个参数就是 this 指向；call 从第二个参数开始代表函数的参数，用逗号分隔；apply 第二个参数代表函数的参数，将参数放入数组中；blind 参数和 call 一样,返回的是函数体,需要再次触发.

注意:
当调用方式不同时，this 指代的含义不同，得到的结果也不同。
this 本身不具备任何含义。

typeof/instanceof的区别

- typeof 是一个操作符，其右侧跟一个一元表达式，并返回这个表达式的数据类型。

返回的结果用该类型的字符串(全小写字母)形式表示，包括以下 6 种：
number、boolean、string、object、undefined、function

- instanceof 是用来判断 A 是否为 B 的实例，表达式为：A instanceof B，如果 A 是 B 的实例，则返回 true,否则返回 false。 在这里需要特别注意的是：instanceof 检测的是原型

call()和apply()区别

```
/*apply()方法*/
function.apply(thisObj[, argArray])
/*call()方法*/
function.call(thisObj[, arg1[, arg2[, [,...argN]]]]);
```

共同之处：
都“可以用来代替另一个对象调用一个方法，将一个函数的对象上下文从初始的上下文改变为由thisObj指定的新对象”。
不同之处：
apply：最多只能有两个参数——新this对象和一个数组argArray。如果给该方法传递多个参数，则把参数都写进这个数组里面，当然，即使只有一个参数，也要写进数组里。如果argArray不是一个有效的数组或arguments对象，那么将导致一个TypeError。如果没有提供argArray和thisObj任何一个参数，那么Global对象将被用作thisObj，并且无法被传递任何参数。
call：它可以接受多个参数，第一个参数与apply一样，后面则是一串参数列表。这个方法主要用在js对象各方法相互调用的时候，使当前this实例指针保持一致，或者在特殊情况下需要改变this指针。如果没有提供thisObj参数，那么 Global 对象被用作thisObj。
实际上，apply和call的功能是一样的，只是传入的参数列表形式不同。

面向对象继承的几种方法

1. 原型继承
2. 原型链继承
3. 混合继承 call apply
4. 超类继承 super

node.js创建服务？

引入模块require(“http”)

```
const http=require(“http”);
http.createServer((req,res)=>{})).listen(“8080”,(req,res)=>{})
```

闭包、常用场景、优缺点

闭包的工作原理

因为闭包只有在被调用时才执行操作，所以它可以被用来定义控制结构。多个函数可以使用同一个环境，这使得他们可以通过改变那个环境相互交流。

闭包就是能够读取其他函数内部变量的函数，可以简单理解成“定义在一个函数内部的函数”

主要应用闭包场合主要是为了：设计私有的方法和变量，有些方法和属性只是运算逻辑过程中的使用的，不想让外部修改这些属性，因此就可以设计一个闭包来只提供方法获取。

使用场景

（1）采用函数引用方式的setTimeout调用。

（2）将函数关联到对象的实例方法。

（3）封装相关的功能集。

优点：

（1）逻辑连续，当闭包作为另一个函数调用参数时，避免脱离当前逻辑而单独编写额外逻辑。

（2）方便调用上下文的局部变量。

（3）加强封装性，可以达到对变量的保护作用

缺点：

（1）由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

（2）闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

伪类与伪元素的特性及其区别：

CSS3中伪类和伪元素的语法不同；　　伪类　:link　:hover　　伪元素　::before　　::after

其中伪类和伪元素的根本区别在于：它们是否创造了新的元素，　这个新创造的元素就叫"伪无素"　，伪元素本质上是创建了一个有内容的虚拟容器；

伪元素/伪对象：不存在在DOM文档中，是虚拟的元素，是创建新元素。 这个新元素(伪元素)　是某个元素的子元素，这个子元素虽然在逻辑上存在，但却并不实际存在于文档树中；

伪类本质上是为了弥补常规CSS选择器的不足，以便获取到更多信息，可以同时使用多个伪类，而只能同时使用一个伪元素；

伪类：存在DOM文档中，(无标签,找不到，　只有符合触发条件时才能看到)，　逻辑上存在但在文档树中却无须标识的“幽灵”分类。

因为伪类是类似于添加类所以可以是多个，而伪元素在一个选择器中只能出现一次，并且只能出现在末尾

W3C中对于二者应用的描述(描述太模糊，不容易理解)：

- 伪类：用于向某些选择器添加特殊的效果
- 伪元素：用于将特殊的效果添加到某些选择器(标签)