# 前端面试分享

1. 讲一下es6有哪些特性？

答：（1）变量声明const和let:

const声明常量，let声明变量，两者都是块级作用域。

const声明的变量都会是常量，不可改的。如果const是一个对象，对象 所包含的值是可以修改的，对象所指向的地址没有改变就行。

1. 模板字符串：之前用“\”和“+”来构建模板，es6用${}和（``）
2. 箭头函数：使用括号包裹参数，跟随一个=>,接着是函数体
3. 函数的默认参数的定：

es6之前默认参数需要重新定义(a=a||’hello’)，es6可以直接赋值（a=’hello’）

1. Spread/Rest操作符：当被用于迭代器中时，它是一个 Spread 操作符，当被用于函数传参时，是一个 Rest 操作符

function foo(x,y,z){

console.log(x,y,z)

}

let arr = [1,2,3];foo(...arr) // 1 2 3

//传参

Founction foo(...args){console.log(args)}

Foo(1,2,3,4)//[1,2,3,4]

1. 二进制和八进制字面量：通过在数字前面添加0o或者0b将其转成八进制
2. 对象和数组结构:

对象：const student = {name:’jing’,age:’22’,sex:’nv’}

es5:const name = student.name,......c.log(name+’’+age+’’+sex)

es6:const {name,age,sex} = student;c.log(name+’’+age+’’+sex)

1. 对象超类：允许在对象中使用super方法
2. for ...of 和 for ...in

for...of 用于遍历一个迭代器，如数组

let letters = [‘a’,’b’,’c’];letter.size = 3;

for (let letter of letters ){c.log(letter)}//a,b,c

for ...in 用来遍历对象中的属性

let letters = [‘a’,’b’,’c’]；

For (let letter in letters){c.log(letters[letter])}//a,b,c

1. ES6中的类

ES6 中支持 class 语法，不过，ES6的class不是新的对象继承模型，它只是原型链的语法糖表现形式。函数中使用 static 关键词定义构造函数的的方法和属性：

class Student {

constructor() {

console.log("I'm a student.");

}

study() {

console.log('study!');

}

static read() {

console.log("Reading Now.");

}}

console.log(typeof Student); // functionlet stu = new Student(); // "I'm a student."

stu.study(); // "study!"

stu.read(); // "Reading Now."

类的继承和超集

class Phone {

constructor() {

console.log("I'm a phone.");

}}

class MI extends Phone {

constructor() {

super();

console.log("I'm a phone designed by xiaomi");

}}

let mi8 = new MI();

1. css动画特性可以用js实现，为什么还要用css实现？
2. 不占用js主线程
3. 可以利用硬件加速
4. 浏览器可以对动画做优化
5. Css3动画提供了2D和3D常规动画属性接口，3D动画在js中无法实现，2D矩阵动画效率高于js。
6. Ajax请求数据重新处理和拦截器是怎么做的？

请求拦截器会在你发送请求之前执行，每一次请求都要去判断是否有token（这个token一般是登录完成后，将用户的token通过localStorage或者cookie存在本地，用户每次进入页面的时候，会先从本地中读取token，如果token存说明已经登录过，则更新token状态，然后在每次请求接口时，都会在请求的header中携带token，服务器就可以根据你携带的token来判断你的登录是否过期，如果没有携带则没有登录）

说到这个的话还有一个相应拦截器：

响应拦截器很好理解，就是服务器返回给我们的数据，我们在拿到之前可以对他进行一些处理。例如：如果后台返回的状态码是200，则正常返回数据，否则的根据错误的状态码类型进行一些我们需要的错误，其实这里主要就是进行了错误的统一处理和没登录或登录过期后调整登录页的一个操作。

1. Bootstrap中栏栅的24个是怎么做到的？

24个col不加数字

1. 轮播图怎么考虑实现的？
2. get和post的区别？
3. H5的新特性和css3的新特性

H5的新特性：canvas，video，audio。

新的特殊内容元素：nav，header，section，article，footer。

新的表单控件：date，time，email，url，search

对本地离线存储有更好的支持。

CSS3的新特性：实现圆角border-radius，阴影box-shadow、box-image，对文字加特效text-shadow,线性渐变gradient，旋转transform，（transform：旋转rotate（9deg）、缩放scale（0.85.0.9）、定位translate（0px，-30px）、倾斜skew（-9deg，0deg））。

增加更多的css选择器,多背景，rgba

1. Let和var在全局定义的时候有什么区别?

var在全局对象windows中能找到，而let不行

提示：无关全局时var和let的区别

1. Var是在全局范围内有效，let代码块内有效
2. Var可以声明多次，let只能声明一次
3. Let不存在变量提升，var会变量提升
4. 数组里面有哪些遍历方式？
5. for(){}
6. forEach(){}
7. for...in
8. for...of
9. Map() \filter()
10. js的本地存储机制有？以及他们的区别

Cookies、sessionStorage、localStorage。

相同点：都存储在客户端

不同点：1.存储的大小不同

Cookie数据大小不能超过4k

sessionStorage和localStorage也有存储大小的限制，但比cookie大得多，可以达到5M。

1. 有效时间不同

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过期时间之前有效，即窗口或浏览器关闭

1. 数据与服务之间的交互方式不同

Cookie的数据会自动的传递到服务器，服务端也可以写cookie都客户端

sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

1. 为什么要清除浮动？怎么清除浮动？

产生原因：子盒子浮动导致父盒子内高度为0，父级盒子不能被撑开发生高度塌陷的情况，导致背景不能显示，边框不能撑开，margin和padding的值不能正确显示

清除浮动的方法：

1. 给父盒子设置合适的高度
2. 给父盒子添加样式overflow：hidden/auto,为了去除兼容性问题，会添加zoom：1
3. 在父盒子的子盒子后面添加一个子盒子，添加样式.clear{clear:both}
4. 采用伪元素，给父元素追加:after,给父元素添加一个类.clearfix{content:””;clear:both}
5. 对闭包的理解
6. Vue的slot插槽的作用和类型？
7. 深浅拷贝
8. 数组去重
9. 数组降维
10. 原型链
11. 链表
12. 讲述前端页面渲染的过程
13. 对vue生命周期的理解
14. 移动端1像素问题

1px变粗的原因是啥了， viewport的设置和屏幕物理分辨率是按比例而不是相同的. 移动端window对象有个devicePixelRatio属性, 它表示设备物理像素和css像素的比例, 在retina屏的iphone手机上, 这个值为2或3, css里写的1px长度映射到物理像素上就有2px或3px那么长。

如何解决1px问题：

1. 媒体查询利用设备像素比缩放，设置小数像素

iOS8下已经支持带小数的px值，media query对应devicePixeRatio有个查询词-webkit-min-device-pixel-ratio，css可以写成这样子

.border { border: 1px solid #999 }@media screen and (-webkit-min-device-pixel-ratio: 2) {

.border { border: 0.5px solid #999 }}@media screen and (-webkit-min-device-pixel-ratio: 3) {

.border { border: 0.333333px solid #999 }}

1. **Viewport+rem**
2. **设置border-image方案**

.border-image-1px {

border-width: 1px 0px;

-webkit-border-image: url("border.png") 2 0 stretch;

border-image: url("border.png") 2 0 stretch;}

Border-width指定边框宽度，可以设定四个值，分别为上右下左border-width：top right bottom left。

1. **Background-image渐变实现**

.border {

background-image:linear-gradient(180deg, red, red 50%, transparent 50%),

linear-gradient(270deg, red, red 50%, transparent 50%),

linear-gradient(0deg, red, red 50%, transparent 50%),

linear-gradient(90deg, red, red 50%, transparent 50%);

background-size: 100% 1px,1px 100% ,100% 1px, 1px 100%;

background-repeat: no-repeat;

background-position: top, right top, bottom, left top;

padding: 10px;

}

1. **box-shadow**

div {

-webkit-box-shadow: 0 1px 1px -1px rgba(0, 0, 0, 0.5);}

##### 6.Transform：scale（0.5）

1.在以上的用法种，无非逃不开一种思想，就是将1px缩小为0.5px来展示，然而。0.5px并不是所有的设备或浏览器都支持，就考虑用媒体查询或viewport将其缩放比例。其实1像素问题的产生基本发生在设置边框或分割线的时候，场景并不覆盖全局样式，因此，直接缩放需要设置的元素，才是我们真正需要的。tranform就能实现这个需求。

div {

height:1px;

background:#000;

-webkit-transform: scaleY(0.5);

-webkit-transform-origin:0 0;

overflow: hidden;}

2.用::after和::befor,设置border-bottom：1px solid #000,然后在缩放-webkit-transform: scaleY(0.5);可以实现两根边线的需求

div::after{

content:'';width:100%;

border-bottom:1px solid #000;

transform: scaleY(0.5);}

3.用::after设置border：1px solid #000; width:200%; height:200%,然后再缩放scaleY(0.5); 优点可以实现圆角，京东就是这么实现的，缺点是按钮添加active比较麻烦。

.div::after {

content: '';

width: 200%;

height: 200%;

position: absolute;

top: 0;

left: 0;

border: 1px solid #bfbfbf;

border-radius: 4px;

-webkit-transform: scale(0.5,0.5);

transform: scale(0.5,0.5);

-webkit-transform-origin: top left;}

##### # 6.5 媒体查询 + transfrom 对方案1的优化

/\* 2倍屏 \*/@media only screen and (-webkit-min-device-pixel-ratio: 2.0) {

.border-bottom::after {

-webkit-transform: scaleY(0.5);

transform: scaleY(0.5);

}}

/\* 3倍屏 \*/@media only screen and (-webkit-min-device-pixel-ratio: 3.0) {

.border-bottom::after {

-webkit-transform: scaleY(0.33);

transform: scaleY(0.33);

}}

1. Js每次调用一个函数自身+1

首先考虑使用闭包

Function getId(){

Var num = 1;

getId = function(){

Return num++;

};

Return num++;

}

c.log(getId());//1

c.log(getId());//2

使用 JavaScript 世界里的一等公民函数来创建闭包来解决试试，

通过即时函数创建闭包，返回一个自增的值.

Var getId=(function(){

Var num = 1；

return function(){

return num++；

}

}）();