**H5方向**

**《移动端设计》学习手册V1.0**

|  |  |
| --- | --- |
| 编写人 | 杨来志 |
| 版本 | V1.0 |
| 时间 | 2017-11-01 |

目录

[1. 学习手册的目的 2](#_Toc1274)

[2. 总监寄语 3](#_Toc17554)

[3. 学习内容 3](#_Toc19884)

[3.1. 面向对象1 3](#_Toc32177)

[3.2. 面向对象2 6](#_Toc20119)

[3.3. 面向对象3 9](#_Toc12345)

[3.4. 面向对象4 12](#_Toc26823)

[3.5. jquery1 15](#_Toc22383)

[3.6. jquery2 18](#_Toc28782)

[3.7. jquery3 19](#_Toc11451)

[3.8. jquery4 22](#_Toc869)

[3.9. jquery5 24](#_Toc28232)

[3.10 ajax1 26](#_Toc29080)

[3.11.ajax2 28](#_Toc10930)

[3.12.html5 30](#_Toc18803)

[3.13.css3 32](#_Toc5493)

[3.14 css3动画效果 34](#_Toc5107)

[3.15.响应式式布局和移动端页面布局 37](#_Toc16267)

[3.16.bootstrap1 39](#_Toc8153)

[3.17.bootstrap2 41](#_Toc31070)

[3.18. bootstrap项目 42](#_Toc21102)

3.19移动端项目2

3.20移动端项目3

# 学习手册的目的

为了更好的帮助学生做到日清日结，日试日毕，帮助学生巩固当日学习内容，帮助老师掌握学生学习情况，针对当日学习内容编写学习手册，将重点学习内容以知识点简答题、技能题、面试题、周练习案例等内容，望各位老师和学生务必重视，并且按时完成每日任务，从而保证教学结果，稳扎稳打，最终实现高薪就业的目标。

# 总监寄语

# 学习内容

## 面向对象1

### 技能练习题

考试范围：购物车管理系统

考试时间：90分钟

分值：每题10分，满分100分

|  |
| --- |
| 题目：  1. 用五种方式创建对象  张三：生火  2. this  普通函数中的this事件中的this类中的this函数call以后替代执行的this  2017-08-19_1321143. set get方法  使用set get方法做一个运动的div并且 点击时改变文字内容。  4. set get方法  set get方法中分析this和设置属性 |

### 面试题：

#### 1. 对象是怎么样定义的,属性是怎么定义的？

|  |
| --- |
| 可以用function定义一个对象,然后根据形参在里面this.x=x可以定义属性,或者可以通过动态的添加一个属性 |

#### 2. 对象是怎么样创建的？

|  |
| --- |
| 1用var obj={}就可以创建一个对象  2.用new Object()也可以创建一个对象 |

#### 3. 讲几个常见的js内置对象,他们有什么好处？

|  |
| --- |
| Object,Array,String,Number,Boolean,Date,Function 可以方便的使用一些方法和常量,譬如String里面就有很多字符串函数 Date可以处理时间 |

#### 4. prototype(原型)是什么,它是怎么使用的？

|  |
| --- |
| 每个函数都有一个prototype属性, 它是一个引用变量, 默认指向一个空Object对象 ,有备用的意思,当调用一个对象的函数或者属性的时候,如果在当前对象里面找不到,那么就到原型里面去找 |

#### 5. 继承有几种方式？分别是什么？

|  |
| --- |
| 原型链继承  优点：可以继承父类构造函数中的属性和方法 也可以继承其原型对象中的属性方法；  缺点：在调用子类构造函数的同时无法向父类函数传参；  冒充继承  优点：可以传参  缺点：不可以继承原型对象中的属性和方法  组合继承  互补互足，趋于完美 |

#### 6. js是如何实现继承的？

|  |
| --- |
| 使用prototype属性就可以实现继承 一般是child.prototype=new parent()就可以实现 |

#### 7.什么是闭包？

|  |
| --- |
| 闭包是有权访问另一个函数作用域的变量的函数。  简单的说，Javascript允许使用内部函数---即函数定义和函数表达式位于另一个函数的函数体内。而且，这些内部函数可以访问它们所在的外部函数中声明的所有局部变量、参数和声明的其他内部函数。当其中一个这样的内部函数在包含它们的外部函数之外被调用时，就会形成闭包。 |

#### 8.谈谈你对this的理解？

|  |
| --- |
| 1、谁来调用函数 this就指向谁  2、call&apply 冒充的对象是谁，this就指向谁；  3、闭包使用时；外层函数的this指针和内层函数的this指针不是同一个东西； |

### 

### 3.1面试题：

1.对象是什么？

一切皆对象

对象：是一系列信息打包的产物

面向对象：以对象的思维去看待事物

面向对象编程：以面向对象的方法编程

2.面向对象是什么？

面向对象是什么？

面向对象是一种对现实世界理解和抽象的方法，是计算机编程技术发展到一定阶段后的产物，是一种高级的编程思想。

面向对象是一种思想，很多种语言之中都有面向对象的思想，这种思想已经不同于面向过程的逻辑

提起对象，先做一个简单的回忆：

回顾对象：javascript中万事万物皆对象。

不用了解其中内部机制， 但是只要会用就可以了。 比如之前学习过的DOM,BOM,时间Date对象是最典型的一个实例：

Date.get  ||  Date.set系列问题。

DOM.appendChild( )....

面向对象：只关注对象提供的功能， 不关注内部的细节。

面向对象是什么？

上课的时候， 我留作业，我统一规定作业的目标，你们完成 ，我不会去管你怎么完成的， 只要你完成就可以。

面向对象的特点：

抽象：抽出和问题相关的东西；

对学生而言：学生管理系统：这时候关注的点在于：学生成绩，学生分数；

世纪佳缘：需要知道，性别年龄，三维;

封装：制造模具的过程

1.用对象：

2.写对象：

继承：

多重继承：继承好多份遗产；

对象的创建方式：（目前先学2种）

**第一种：Object构造函数创建**

|  |  |
| --- | --- |
|  | var Person = new Object();  Person.name = 'Nike';  Person.age = 29; |

这行代码创建了Object引用类型的一个新实例，然后把实例保存在变量Person中。

**第二种：使用对象字面量表示法**

|  |  |
| --- | --- |
|  | var Person = {};//相当于var Person = new Object();  var Person = {   name:'Nike'; age:29;   } |

对象字面量是对象定义的一种简写形式，目的在于简化创建包含大量属性的对象的过程。也就是说，第一种和第二种方式创建对象的方法其实都是一样的，只是写法上的区别不同

对象有方法和属性

**方法        属性**

什么是方法和属性那？

var timer=null

DOM.timer=null

测试证明：

属性其实就是变量！

婚前婚后的差别；

婚前：谁都不属于谁。

婚后：已经有了归宿了。

变量是自由的：属性是属于对象的；

方法也是同理；

OOA----面向对象分析方法

OOD---面向对象设计

OOP---面向对象编程

3.面向过程编程和面向对象编程的比较

    假如把程序当成讲故事，面向过程是编年史，面向对象是纪传史。对于复杂的程序/宏大的故事，事实都证明了对象/纪传是更合理的表述方式。

4.值传递和引用传递

js的数据类型回顾：undefined、Null、Boolean、Number、String、Object,function

这些数据类型在使用的时候是有区别的。他们分为：

基本类型: undefined、Null、Boolean、Number、String五种 (简单的数据段);

引用类型: object (由多个值构成)。

// 基本类型

var num1 = 5;

var num2 = num1;

num2 = 10;

console.log(num1 + ' | ' + num2);

// 引用类型

var obj1 = new Object();

obj1.num = 5;

var obj2 = obj1;

obj2.num = 10;

console.log(obj1.num + ' | ' + obj2.num);

总结：引用类型复制的是指针的指向。

了解：

                 堆                                               栈

面试题：

1.送分题

var a=10;

var b=sum(a);

console.log(a+","+b);//10、20   100

function sum(num){

    return num+=10;

}

2.一般题

var arr=[1,2,3,4];

var arr1=arrr;

arr1[0]=111;

console.log(arr+","+arr1);

3.百里挑一

var arr=[11,12,13];

var arr1=sum(arr);

console.log(arr1+","+arr);

function sum(num){

    return num[0]+=10;

}

变量声明提升

全局变量的定义和函数的定义都会被提升，提升到这个js文件正式执行之前。

如果在赋值语句执行之前试图使用全局变量，值是undefined

    var b;

    var a=10;

    function play(){

        console.log(a);//这个函数里面没有a,就会去上一级找，直到找到或者找完

    }

    function play2(){

        //var a;

        console.log(a);//undefined

        var a=11;//a=11

    }

    play2()

    var b=111;//b=111

3.2面向对象2

技能题库

1. 函数声明

普通函数声明 匿名函数声明 构造函数 函数的独立执行 函数的call 参数 return

用构造函数写随机数 随机颜色

2. 函数中的call apply argument 的使用

使用call和apply替换对象

3. 函数对象 argument length

使用call和apply替换对象 结合argument和length设置对象属性

4. 函数

用函数创建对象 用函数实现div动画 用函数查找数组中的某项 set get方法设置年龄

5. 面向对象

用面向对象在构造函数中用正则表达式判断字符串是否符合要求。

3.2知识点集

### 变量、作用域及内存

#### 基本类型值与引用类型值

#### 传递参数

#### 检测类型

#### 执行环境与作用域

#### 块级作用域

#### var关键字

#### 变量查询

#### 内存生命周期

#### 变量内存及什么是内存泄露

### 基本包装类型

#### 基本包装类型概述

#### Boolean 类型

#### Number 类型

#### String 类型

3.3面向对象3

技能题库

现在要介绍的是，对象之间的"继承"的五种方法。

比如，现在有一个"动物"对象的构造函数。

　　function Animal(){

　　　　this.species = "动物";

　　}

还有一个"猫"对象的构造函数。

　　function Cat(name,color){

　　　　this.name = name;

　　　　this.color = color;

　　}

怎样才能使"猫"继承"动物"呢？

**一、 构造函数绑定**

第一种方法也是最简单的方法，使用call或apply方法，将父对象的构造函数绑定在子对象上，即在子对象构造函数中加一行：

　　function Cat(name,color){

**Animal.apply(this, arguments);**

　　　　this.name = name;

　　　　this.color = color;

　　}

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

apply

arguments是参数的意思，指的是在调用函数的时候传入的所有参数

这也叫冒充继承。可以理解为冒充别的构造函数生成的实例

function Person(na,ag){

     this.name=na,

     this.age=ag;

}

//子类

function Man(){

     Person.apply(this,arguments);//冒充继承

     this.sex="男";

}

var liu=new Man("刘耀夏",12);

console.log(liu.name);//刘耀夏

1. call

call方法和apply是孪生兄弟。基本是一致的，call方式把所有参数逐个传入，apply是传入一个数组

//apply方式

//子类

function man(hair,eye,skin){

    person.apply(this,[hair,eye,skin]);       //第二个参数直接写个arguments就可以了

    this.feature = ['beard','strong'];

}

//call方式

//子类

function man(hair,eye,skin){

    person.call(this,hair,eye,skin);

}

总结：三种继承方式的区别

prototype方式不能往里传参，灵活性较差，一般用于绑定公共的方法,可以节省内存

apply有两个参数，第一个是this,第二个是传入的数据，是个数组

call 参数不确定呢，第一个是this,后面是传入的一个个参数

**二、 prototype模式**

第二种方法更常见，使用prototype属性。

如果"猫"的prototype对象，指向一个Animal的实例，那么所有"猫"的实例，就能继承Animal了。

　　Cat.prototype = new Animal();

　　Cat.prototype.constructor = Cat;

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

代码的第一行，我们将Cat的prototype对象指向一个Animal的实例。

　　Cat.prototype = new Animal();

它相当于完全删除了prototype 对象原先的值，然后赋予一个新值。但是，第二行又是什么意思呢？

　　Cat.prototype.constructor = Cat;

原来，任何一个prototype对象都有一个constructor属性，指向它的构造函数。如果没有"Cat.prototype = new Animal();"这一行，Cat.prototype.constructor是指向Cat的；加了这一行以后，Cat.prototype.constructor指向Animal。

　　alert(Cat.prototype.constructor == Animal); //true

更重要的是，每一个实例也有一个constructor属性，默认调用prototype对象的constructor属性。

　　alert(cat1.constructor == Cat.prototype.constructor); // true

因此，在运行"Cat.prototype = new Animal();"这一行之后，cat1.constructor也指向Animal！

　　alert(cat1.constructor == Animal); // true

这显然会导致继承链的紊乱（cat1明明是用构造函数Cat生成的），因此我们必须手动纠正，将Cat.prototype对象的constructor值改为Cat。这就是第二行的意思。

这是很重要的一点，编程时务必要遵守。下文都遵循这一点，即如果替换了prototype对象，

　　o.prototype = {};

那么，下一步必然是为新的prototype对象加上constructor属性，并将这个属性指回原来的构造函数。

　　o.prototype.constructor = o;

**三、 直接继承prototype**

第三种方法是对第二种方法的改进。由于Animal对象中，不变的属性都可以直接写入Animal.prototype。所以，我们也可以让Cat()跳过 Animal()，直接继承Animal.prototype。

现在，我们先将Animal对象改写：

　　function Animal(){ }

　　Animal.prototype.species = "动物";

然后，将Cat的prototype对象，然后指向Animal的prototype对象，这样就完成了继承。

　　Cat.prototype = Animal.prototype;

　　Cat.prototype.constructor = Cat;

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

与前一种方法相比，这样做的优点是效率比较高（不用执行和建立Animal的实例了），比较省内存。缺点是 Cat.prototype和Animal.prototype现在指向了同一个对象，那么任何对Cat.prototype的修改，都会反映到Animal.prototype。

所以，上面这一段代码其实是有问题的。请看第二行

　　Cat.prototype.constructor = Cat;

这一句实际上把Animal.prototype对象的constructor属性也改掉了！

　　alert(Animal.prototype.constructor); // Cat

**四、 利用空对象作为中介**

由于"直接继承prototype"存在上述的缺点，所以就有第四种方法，利用一个空对象作为中介。

　　var F = function(){};

　　F.prototype = Animal.prototype;

　　Cat.prototype = new F();

　　Cat.prototype.constructor = Cat;

F是空对象，所以几乎不占内存。这时，修改Cat的prototype对象，就不会影响到Animal的prototype对象。

　　alert(Animal.prototype.constructor); // Animal

我们将上面的方法，封装成一个函数，便于使用。

　　function extend(Child, Parent) {

　　　　var F = function(){};

　　　　F.prototype = Parent.prototype;

　　　　Child.prototype = new F();

　　　　Child.prototype.constructor = Child;

　　　　Child.uber = Parent.prototype;

　　}

使用的时候，方法如下

　　extend(Cat,Animal);

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

这个extend函数，就是YUI库如何实现继承的方法。

另外，说明一点，函数体最后一行

　　Child.uber = Parent.prototype;

意思是为子对象设一个uber属性，这个属性直接指向父对象的prototype属性。（uber是一个德语词，意思是"向上"、"上一层"。）这等于在子对象上打开一条通道，可以直接调用父对象的方法。这一行放在这里，只是为了实现继承的完备性，纯属备用性质。

**五、 拷贝继承**

上面是采用prototype对象，实现继承。我们也可以换一种思路，纯粹采用"拷贝"方法实现继承。简单说，如果把父对象的所有属性和方法，拷贝进子对象，不也能够实现继承吗？这样我们就有了第五种方法。

首先，还是把Animal的所有不变属性，都放到它的prototype对象上。

　　function Animal(){}

　　Animal.prototype.species = "动物";

然后，再写一个函数，实现属性拷贝的目的。

　　function extend2(Child, Parent) {

　　　　var p = Parent.prototype;

　　　　var c = Child.prototype;

　　　　for (var i in p) {

　　　　　　c[i] = p[i];

　　　　　　}

　　　　c.uber = p;

　　}

这个函数的作用，就是将父对象的prototype对象中的属性，一一拷贝给Child对象的prototype对象。

使用的时候，这样写：

　　extend2(Cat, Animal);

　　var cat1 = new Cat("大毛","黄色");

alert(cat1.species); // 动物

3.3 知识点集

### 创建对象

#### 工厂模式

#### 构造函数

#### 构造函数的规范

##### 构造函数与工厂模式的不同

##### 构造函数执行流程

##### 构造函数this的使用

##### 构造函数与普通函数的区别

### 原型

#### 原型概述

#### \_\_proto\_\_属性

#### 原型模式的执行流程

#### hasOwnProperty()函数、in 操作符

#### constructor属性

#### 原型链

#### 原型声明顺序与重写原型

#### 组合构造函数+原型模式：

#### 动态原型模式

#### 寄生构造函数

#### 稳妥构造函数

3.4 面向对象4

3.4 面试题

现在要介绍的是，对象之间的"继承"的方法。

比如，现在有一个"动物"对象的构造函数。

　　function Animal(){

　　　　this.species = "动物";

　　}

还有一个"猫"对象的构造函数。

　　function Cat(name,color){

　　　　this.name = name;

　　　　this.color = color;

　　}

怎样才能使"猫"继承"动物"呢？

1.继承

1.1原型链继承

原型链继承是使用prototype属性实现了继承

如果"猫"的prototype对象，指向一个Animal的实例，那么所有"猫"的实例，就能继承Animal了。

    Cat.prototype = new Animal();

　　Cat.prototype.constructor = Cat;

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

代码的第一行，我们将Cat的prototype对象指向一个Animal的实例。

　　Cat.prototype = new Animal();

它相当于完全删除了prototype 对象原先的值，然后赋予一个新值(重写)。

但是，第二行又是什么意思呢？

　　Cat.prototype.constructor = Cat;

原来，任何一个prototype对象都有一个constructor属性，指向它的构造函数。如果没有"Cat.prototype = new Animal();"这一行，Cat.prototype.constructor是指向Cat的；加了这一行以后，Cat.prototype.constructor指向Animal。

　　alert(Cat.prototype.constructor == Animal); //true

更重要的是，每一个实例也有一个constructor属性，默认调用prototype对象的constructor属性。

　　alert(cat1.constructor == Cat.prototype.constructor); // true

因此，在运行"Cat.prototype = new Animal();"这一行之后，cat1.constructor也指向Animal！

　　alert(cat1.constructor == Animal); // true

这显然会导致继承链的紊乱（cat1明明是用构造函数Cat生成的），因此我们必须手动纠正，将Cat.prototype对象的constructor值改为Cat。这就是第二行的意思。

这是很重要的一点，编程时务必要遵守。下文都遵循这一点，即如果替换了prototype对象，

　　o.prototype = {};

那么，下一步必然是为新的prototype对象加上constructor属性，并将这个属性指回原来的构造函数。

　　o.prototype.constructor = o;

**还可以直接使用prototype实现继承**

由于Animal对象中，不变的属性都可以直接写入Animal.prototype。所以，我们也可以让Cat()跳过 Animal()，直接继承Animal.prototype。

现在，我们先将Animal对象改写：

　　function Animal(){ }

　　Animal.prototype.species = "动物";

然后，将Cat的prototype对象，然后指向Animal的prototype对象，这样就完成了继承。

　　Cat.prototype = Animal.prototype;

　　Cat.prototype.constructor = Cat;

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

与前一种方法相比，这样做的优点是效率比较高（不用执行和建立Animal的实例了），比较省内存。缺点是 Cat.prototype和Animal.prototype现在指向了同一个对象，那么任何对Cat.prototype的修改，都会反映到Animal.prototype。

所以，上面这一段代码其实是有问题的。请看第二行

　　Cat.prototype.constructor = Cat;

这一句实际上把Animal.prototype对象的constructor属性也改掉了！

　　alert(Animal.prototype.constructor); // Cat

直接用构造的函数的原型对象=另一个构造函数的原型对象的方法，这样是2个构造函数共用一个原型对象，危险极大，慎用。

prototype=另一个构造函数的实例的话，没有太大的危险，但是不能往里传参，灵活性较差，一般用于绑定公共的方法,可以节省内存

只要用了Prototype，一定记得把原型对象上的构造器属性改回来

1.2冒充继承

冒充是通过父类生产的实例，从而继承父类的东西。

使用call或apply方法，将父对象的构造函数绑定在子对象上，即在子对象构造函数中加一行：

　　function Cat(name,color){

**Animal.apply(this, arguments);**

　　　　this.name = name;

　　　　this.color = color;

　　}

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

apply

arguments是参数的意思，指的是在调用函数的时候传入的所有参数

这也叫冒充继承。可以理解为冒充别的构造函数生成的实例

function Person(na,ag){

     this.name=na,

     this.age=ag;

//子类

function Man(){

     Person.apply(this,arguments);//冒充继承

     this.sex="男";

}

var liu=new Man("刘耀夏",12);

console.log(liu.name);//刘耀夏

apply：是通过把自己的孩子放在别人家里实现的继承。

            把子类构造函数构造出的实例在父类中执行一遍。

        格式：父类.apply(this,需要传给父类的参数)

        tip:为了解决父类需要传参，子类也需要传参的问题，建议把需要传给父类的参数写在前面，传给子类的参数写在后面，这样在调用apply方法的时候，可以直接使用arguments.

例如：

    //父类

    function Person(na,ag){

        this.name=na;

        this.age=ag;

    }

    1;

    //子类

    function Man(){

        Person.apply(this,**arguments**);

        this.sex=**arguments[2]**;

    }

    var xiaoming=new Man("刘",1,"**不详**");

    console.log(xiaoming.name);

    console.log(xiaoming.sex)

1. call

call方法和apply是孪生兄弟。基本是一致的，call方式把所有参数逐个传入，apply是传入一个数组

//apply方式

//子类

function man(hair,eye,skin){

    person.apply(this,[hair,eye,skin]);       //第二个参数直接写个arguments就可以了

    this.feature = ['beard','strong'];

}

//call方式

//子类

function man(hair,eye,skin){

    person.call(this,hair,eye,skin);

}

由于Animal对象中，不变的属性都可以直接写入Animal.prototype。所以，我们也可以让Cat()跳过 Animal()，直接继承Animal.prototype。

现在，我们先将Animal对象改写：

　　function Animal(){  }

　　Animal.prototype.species = "动物";

然后，将Cat的prototype对象，然后指向Animal的prototype对象，这样就完成了继承。

　　Cat.prototype = Animal.prototype;

　　Cat.prototype.constructor = Cat;

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

与前一种方法相比，这样做的优点是效率比较高（不用执行和建立Animal的实例了），比较省内存。缺点是 Cat.prototype和Animal.prototype现在指向了同一个对象，那么任何对Cat.prototype的修改，都会反映到Animal.prototype。

所以，上面这一段代码其实是有问题的。请看第二行

　　Cat.prototype.constructor = Cat;

这一句实际上把Animal.prototype对象的constructor属性也改掉了！

　　alert(Animal.prototype.constructor); // Cat

prototype继承：1.子类继承父类的一个实例:子类的原型时父类实例有的所有东西；缺点：不能传参

                        2.子类的原型对象=父类的原型对象：只能继承父类原型的所有东西(构造函数里面的东西是没有弟)，缺点：危险很大，只要修改了子类的原型，父类的原型也跟着变化(其实此时他两的原型是同一个对象)

冒充继承：1.apply：通过将子类的实例传入父类的构造函数，从而实现继承；可以传入参数，灵活性较好，第二个参数是数组  缺点：不能继承父类原型对象上的东西

                2.call：和apply是一样的，只是传入的第二个参数是父类中需要的参数要一一写入。

**利用空对象作为中介**

由于"直接继承prototype"存在上述的缺点，所以就有第四种方法，利用一个空对象作为中介。

　　var F = function(){     };

　　F.prototype = Animal.prototype;

　　Cat.prototype = new F();

　　Cat.prototype.constructor = Cat;

F是空对象，所以几乎不占内存。这时，修改Cat的prototype对象，就不会影响到Animal的prototype对象。

　　alert(Animal.prototype.constructor); // Animal

我们将上面的方法，封装成一个函数，便于使用。

　　function extend(Child, Parent) {

　　　　var F = function(){};

　　　　F.prototype = Parent.prototype;

　　　　Child.prototype = new F();

　　　　Child.prototype.constructor = Child;

　　　　Child.uber = Parent.prototype;

　　}

使用的时候，方法如下

　　extend(Cat,Animal);

　　var cat1 = new Cat("大毛","黄色");

　　alert(cat1.species); // 动物

这个extend函数，就是YUI库如何实现继承的方法。

另外，说明一点，函数体最后一行

　　Child.uber = Parent.prototype;

意思是为子对象设一个uber属性，这个属性直接指向父对象的prototype属性。（uber是一个德语词，意思是"向上"、"上一层"。）这等于在子对象上打开一条通道，可以直接调用父对象的方法。这一行放在这里，只是为了实现继承的完备性，纯属备用性质。

总结：三种继承方式的区别

prototype方式不能往里传参，灵活性较差，一般用于绑定公共的方法,可以节省内存

apply有两个参数，第一个是this,第二个是传入的数据，是个数组

call 参数不确定呢，第一个是this,后面是传入的一个个参数

1.3组合继承:

                当既需要继承构造函数的内容又需要继承原型中的内容，就使用组合继承。

            function Dog(name){

                  this.name = name;

            }

            Dog.prototype.say = function(){

                  alert("汪")

            }

            function Erha(name){

                  Dog.call(this,name)//继承构造函数的属性及方法

            }

            Erha.prototype = new Dog();//继承原型对象中的属性及方法

            Erha.prototype.constructor=Erha;

            //既可以传参   也可以继承原型对象

            var ha = new Erha("哈士奇")

            console.log(ha.constructor==Erha)

三种继承方式：

        1.原型继承:继承的是原型对象

        2.冒充继承:继承的是构造函数

        3.组合继承(混合继承):既要继承原型对象，又要构造函数里面的东西。

2.匿名函数和闭包

2.1匿名函数

    没有名字的函数就是匿名函数

        1.绑定事件的时候

document.getElementByTagName("div")[0].onclick=function(){}

        2.函数自执行的时候

            自执行函数：可以自己执行的函数:   可以用于一次性的功能，最大的好处在于用完就销毁了，可以及时释放内存，避免内存泄露

内存泄露：不再需要的内存占用没有及时的清除就是内存泄露

(function(){

})()

2.2闭包优缺点

要理解闭包，首先必须理解Javascript特殊的变量作用域。 变量的作用域无非就是两种：全局变量和局部变量。 Javascript语言的特殊之处，就在于函数内部可以直接读取全局变量

var a=6;

function play(){

    console.log(a)

}

play()

var a=6;

function play(){

   var a=2;

}

play();

console.log(a)

上边的变量a是全局变量，下边的b是私有变量 注意：如果右边的b=6前面不写var，b就被升级为全局变量了

攻城狮   ：无所不能

            :在函数外部声明全局变量；在函数内部声明的时候不加var

出于种种原因，我们有时候需要得到函数内的局部变量值。但是，前面已经说过了，正常情况下，这是办不到的，只有通过变通方法才能实现。

 右边案例中fun1是一个函数，里面套叠了一个fun2的函数，因为fun2和a从属一个作用域下，因此可以返回a的值最后我们执行的最后把fun2这个函数通过fun1返回出去。因此fun1（）执行以后就是fun2，而fun2要执行就需要给他再加一个（），因此就变成了fun1()();你也可以通过赋值的方法将它拆解开，例如var fun=fun1();var a1=fun();console.log(a1);

function play(){

    var a=1;

    function sing(){

        return a;

    }

    return sing;

}

console.log(play()())

到底什么是闭包？

闭包是指有权访问一个函数作用域中的变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,通过return 把另一个函数返回出来

闭包的特点和优点

闭包的特点

    1.函数嵌套函数2.函数内部可以引用外部的参数和变量3.参数和变量不会被垃圾回收机制回收

闭包的优点

    1.希望一个变量长期驻扎在内存中2.避免全局变量的污染3.私有成员的存在

闭包减少代码全局污染

面试题：

function foo(x){

    var a=3;

    return function(y){

        console.log(x+y+(++a));

    }

}

var bar=foo(2);

bar(3);//打印的结果是多少？ 9     2+3+4

function fun(n, o) {

        console.log(o);

        return  {

            fun:function(m){

                return fun(m, n);

            }

        }

    }

var a=fun(0);//undefined   a:function fun(m){ return fun(m,0)}

a.fun(1);//0    fun(1,0)

a.fun(2);//0

a.fun(3);//0

var b=fun(0).fun(1).fun(2).fun(3);

var c=fun(0).fun(1);

c.fun(2);

c.fun(3);

2.3关于this对象

        this指向拥有这个方法的对象。

2.4模仿块级作用域

(function(){

})(

3.4 知识点集

### 继承

#### 原型链继承

#### 冒充继承

#### 组合继承

### 匿名函数和闭包

#### 匿名函数

#### 闭包优缺点

#### 关于this对象

#### 模仿块级作用域

#### 静态私有变量

#### 模块模式

3.5 JQuery1

3.5面试题

jquery

简写为jq

jq是js的库，里面给我们封装好了很多的方法（使用这些方法的时候必须借用Jq对象，在同一个语句中，jq对象只能执行Jq方法，js对象只能执行Js方法，不能交叉使用）            函数库

它可以用最少的代码，完成更多复杂而困难的功能，所以赢得到了开发者的青睐

主指：以更少的代码、实现更多的功能。（**write less,do more**)

比较关键的版本信息

2011年11月发布了jQuery1.7，引入了.on()和.off()简介的API解决事件绑定及委托容易混淆的问题

2013年1月发布了jQuery1.9，CSS的多属性设置，增强了CSS3。

2013年4月发布了jQuery2.0，5月发布了jQuery2.0.2，一个重大更新版本，不在支持IE6/7/8，体积更小，速度更快。可以用于移动端(zepto)

关于版本的一些常识问题

大版本：第一个小数点之前的，比如1.x.x升级到2.x.x，这种升级规模是最大的，改动的地方是最多的，周期也是最长的，jQuery从1.x.x到2.x.x用了7年

小版本：第一个小数点到第二个小数点之间的，比如1.7升级到1.8，改动适中，增加或减少了一些功能，一般周期半年到一年左右。

微版本：最后一个小数点后面，比如.18.1升级到1.8.2，修复一些bug或错误之类。

jq升级带来的学习成本问题：并不是每次升级一个版本都会增加或剔除功能的，一半左右都是内部优化，升级到新版本并不需要任何学习成本。就算在新的版本增加了一些功能，只需要几分钟了解一下即可使用，无需清零之前的知识，只需后续累加。

jq的功能：jQuery作为JavaScript封装的库，他的目的就是为了简化开发者使用JavaScript的繁琐操作，功能功能和js一样，只是使用更简单。

jq的优势

jq最大的优势是：使用方便，不用考虑兼容

下载jq:在百度搜索中输入：jquery下载，可以免费下载Jquery的所有版本

文档：在百度搜索中输入:jquery+详细的版本号,就可以出现详细的版本号。

基础核心

1.$符开头，和jQuery完全一样，$是jQuery的简写

2.jQuery的代码模式是采用的**连缀方式**，可以不停的连续调用jq提供给我们的方法。

**之所以可以这么写是因为绝大部分方法的返回值都是使用这个方法的Jq对象**

例如： $('#box').css('color','red').css('font-size','50px');//连缀

3.注释方式和js一样

//   单行注释

/\*             \*/                 多行注释

4.js中的   window.onload                           ①

    jq中的:$(function(){                                ②

               })

比较：①js的入口，必须等页面的全部内容加载完才会执行里面的内容；一个页面最多能有1个window.onload

          ②是jq的入口，等页面结构加载完才会执行里面的代码；一个页面可以使用②的次数不受限制

<script>

    window.onload  =function(){

            //在这写你要写的js代码

    }

<script/>

<script>

    $(function(){

            //在这写你要写的代码

      })

<script/>

常规选择器

选择器引擎：继承了CSS的语法，可以对DOM元素的标签名、属性名、状态等进行快速准确的选择，并且不必担心浏览器的兼容性。

**简单选择器：●在使用jQuery选择器时，我们首先必须使用“$()”函数来包装我们的CSS规则。而CSS规则作为参数传递到jQuery对象内部后，再返回包含页面中对应元素的jQuery对象。随后，我们就可以对这个获取到的DOM节点进行行为操作了。**

案例：想获取id的box的元素:    $("#box")

      想获取class名为desk的元素： $(".desk")

查看选择的元素的个数：

        方法：size()

        属性；length

案例：这两个方法都是达到目的

      console.log($("li").size())

      console.log($("li").length)

小问题：不要在一个页面中的多个元素使用同一个id

            如果这么做了，jq只会选择第一个,杜绝这种用法

进阶选择器

群组选择器：一次获取多个选择器的dom对象，中间用,分隔

                    例如：$("#box,.ul1").css("background","red")

后代选择器:可以像CSS一样选择一个元素的指定后代元素，并不一定是他的直接子元素

    例如： $(".ul1 li").css("background","green")

子代选择器：找的是直接子元素

        $(".ul1>li").css("background","green)

通配选择器：获取所有元素标签的dom对象,这个，一般都不会用

    例如：$("\*")

警告：在实际使用上，通配选择器一般用的并不多。比如：$('\*')， 这种使用方法效率很低，影响性能，建议尽可能少用。

上面说的选择器，可以自由搭配

例如：

$("#box  ul,li,span").css("color","red");

可以在id和class中指明元素前缀，可以顺便判断元素标签

  例如：   $("div**.**box").css("background","yellow")

高级选择器   低版本的jq可能不支持

后代选择器；可以追溯好几层的迭代关系

            例如：$("ul li a")

next选择器：只获取指定节点的下一个同级dom对象

            例如：$(".box+div").css("background","blue")

                    首先找到.box节点，然后判断他相邻的是否为div，如果是就返回

    jq为下一个同级节点提供了一个方法:next()

        例如：$(".box").next("ul").css("background","#5F9EA0")

            会首先找到.box，然后找下一个同级相邻元素，如果是ul就返回

nextAll选择器：获取某节点后面所有符合规则的同级元素

            例如：$(".box~div").css("background","blue")

                    首先找到.box节点，找他后面所有同级的div并返回

        jq提供了一个等价的方法：nextAll()

在层次选择器中，上面三种高级选择器是不支持ie6的。jq1.x是支持ie6的

jq为后代选择器提供了一个等价的find()方法，会找指定元素内所有符合的元素

    例如：$(".box2").find("li").css("background","#000000")

子选择器：只找直接的子元素，不会选择子元素的子元素  兼容ie6

         例如：$(".box2>li").css("background","#000000")

        jq为子选择器提供了一个等价的children方法

        例如：$(".box2").children("li").css("background","#000000")

                只会找.box2内标签为Li的**直接子元素**

在 find()、next()、nextAll()和 children()这四个方法中，如果不传递参数，就相当于传递 了“\*”

推荐使用**find(),next(),nextAll(),children()**这些jq自己的方法，运行速度比高级选择器快，好记还好用

jq还提供了其他方法

prev()  :同级的上一个元素

          例如：$("#box").prev("p").css("background","green");

prevAll():同级所有上面的元素

          例如：$("#box").prevAll("p").css("background","green");

nextUntil()和prevUntil()方法是选定同级的下面或上面的所有节点，选定所有的非指定的所有元素，一旦遇到指定元素就停止选定。你指定的元素要写在prevUntil()里面

    例如：$(".ul1").prevUntil(".box").css("background","orangered")

siblings():同级的所有元素，不包括自己，如果往里传参了，就在结果中再筛选一次

      例如：$(".ul1").siblings("div").css("background","orchid")

3.5 知识点集

### jQuery入门

#### 什么是 jQuery

#### 学习 jQuery 的条件

#### jQuery 的版本

#### jQuery 的功能和优势

#### 下载及运行 jQuery

### 基础核心

#### 代码风格

#### 加载模式

#### 对象互换

#### 多个库之间的冲突

### 常规选择器

#### 简单选择器

#### 进阶选择器

#### 高级选择器

### 过滤选择器

#### 基本过滤器

#### 内容过滤器

#### 可见性过滤器

#### 子元素过滤器

3.6 JQuery2

引入jq后，可以写js，也可以写jq，在同一个语句中，千万不要把js和Jq交叉使用

js获取的dom元素和jq获取的dom元素是不一样的,不用混写

例如：$("#box").style.width="40px";//**这是一个错误案例**

1.设置文本内容

js:

     document.getElementsByTagName("div")[0].innerHTML="<h1>早上好</h1>";

     document.getElementsByTagName("div")[1].innerText="<h1>早上好</h1>";

     //innerHTML可以往节点里面写标签

     //innerText只能往里写文本内容

     //console.log(document.getElementsByTagName("input")[0].value)

     //获取表单的内容

jq:

html()：可以获取指定节点的所有内容，包括标签                     还可以设置指定节点的内容，可以往里写标签

text():只能获取指定节点的文本内容，不包括标签                    可以设置指定节点里面的文本内容，写标签会原样输出到页面上

val():可以获取表单中的value值                                  也可以设置表单中的value

注：当我们使用html(),text()设置元素里面的内容的时候，会自动先清空之前的数据，如果想在原有数据的基础上做删改怎么办？看下面

html()还可以传入一个函数function,这个函数有2个形参，第一个是当前元素在集合中的位置，第二个当前元素原本的内容

案例：

    $("div").html(function(index,html){

        //  console.log(index,html)  //打印下看看index和html分别是什么

          return html+"hello"   //在原本内容基础上添加个hello

     })

text()和html()这个一样

val()也一样

tip:        键值对:

在对象中，属性和属性值一一对应的，千万不能写一对多，但是可以用数组和对象来存储大量数据。

键和值（属性和属性值）通过":"连接，每一对键值对之间通过",连接"     。建议键名也加双引号

{

    "user":"ylz",

    age:[11,1,21,3,6],

    school:{"name":"清华","id":"110"}

}

json是一种数据的格式                 ------对象的写法

2.元素属性操作:**attr()**

可以用来获取某个节点的属性， 例如：$("#box").attr("id")

可以用来设置某个节点的属性，例如：$("#box").attr("id","box1")

还可以一次给一个节点设置多个属性，例如：$("#box").attr(**{"id":"box1","clasName":"box3"}**)

\*还可以给集合中的每一个元素设置属性,通过传入一个function来实现，例如：

    $("div").attr("id",function(index,value){  //div是个集合，可以对这个集合中的么一个元素的id属性在原有的基础上进行修改；

          //console.log(index,value)

          return value+"1";      //return后面跟的是返回的新属性值，如果不想在原属性的基础上做修改可以直接return 一个新属性值

     })

3.元素样式操作：**css()**方法

获取某个元素的样式，例如：$("div").css("background");

设置某个元素的样式，例如：$("div").css("background","red");

一次获取某个元素的多个样式.例如：$("div").css(["background","width","height"]);传入的是个数组

还可以针对集合中的元素做修改，例如：$("div").css("background",function(index,value){ return  新的属性值  })  传入的第二个参数是回调函数

设置某个元素行内多个css 样式，例如：：$("div").css(**{"background":"red","width":"100px"}**)

添加类名：一次添加一个类名，例如：$("div").addClass("box1");

一次添加多个类名,例如：$("div").addClass("box1 box2 box3");//添加的每个类名中间用空格连接

移除元素原本的某个类名，例如：$("div").removeClass("box1")

一次移除元素上的多个类名，例如：$("div").removeClass("box1 box2 box3");//中间是空格

**来回切换类名**，可以用于点击事件里面，点一次加个类名，再点就把刚加的类名去掉，注意添加的类名的顺序(放在原本类名的后面)，例如：

     $("button").click(function(){

          $("div").toggleClass("desk");

     })

一次切换多个类名，例如：

    $("button").click(function(){

          $("div").toggleClass("desk box bag");

     })

toggleClass()里面可以传回调函数，可以在里面进行设置：根据父元素来设置class属性

$('div.foo').toggleClass(function() {

     if ($(this).parent().is('.bar') {

         return 'happy';

     } else {

        return 'sad';

     }

});

4.遍历方法：$.each()

可以遍历对象数组，例如：

//在jq中如果想用this，得用$(this)

     $("img").each(function(index){

//        this.src="test"+index+".jpg"，   jq方法里面可以写js方法，不建议使用，切忌一个语句中jq,js混用

          $(this).css("width",index\*200+"px")

     })

$.each()可以遍历一个对象集合，并且可以根据索引值给集合中的每一个元素设置属性

5.css操作

width():获取一个元素的宽度，例如：$("div").width();得到的宽度是不带单位的，如果想设置宽度，直接在width()方法里面传入数值就可以了。默认单位是Px，如果需要其他单位，可以传入

还可以传入个函数，然后根据当前元素在集合中的位置来设置它的宽度,例如：

$("div").width(function(index,width){  return 你想设置的宽度  })

height()方法同width()方法

元素偏移方法

相对于可视口的偏移：offset().left

相对于父元素的偏移,例如:$(".son").positon()   结果是个对象

获取竖直方向的当前滚动条位置，例如：$(window).scrollTop()

设置当前滚动条竖直方向的位置，例如：$(window).scrollTop(300)

水平的话用scrollLeft()

3.6 知识点集

### 基础 DOM 和 CSS 操作

#### DOM 简介

#### 设置元素及内容

#### 元属性操作

#### 元素样式操作

#### CSS 方法

### DOM 节点操作

#### 创建节点

#### 插入节点

#### 包裹节点

#### 节点操作

### 表单选择器

#### 常规选择器

#### 表单选择器

#### 表单过滤器

### 基础事件

#### 绑定事件

#### 简写事件

#### 复合事件

3.7 JQuery3

3.7 面试题

DOM操作

js:增删改查

增：appendChild()向父节点的末尾追加内容        insertBefore()向指定节点的前面加内容

删:remove()不向下兼容(6,7,8)   removeChild()从父节点上移除

改：replace()

查：document.getElementById()

jq的DOM操作：

6.1.创建节点

     //创建节点

     var 变量名=$('想添加的结构');//这里建议写单引号，避免与里面类名的双引号有冲突

     //往页面元素中添加

     $("body").append(刚才写的变量名)

注：如果页面上有5个div，然后创建了一个span节点往div里面插入。如果不指明往第几个div里面插入，jq就会往每个div里面都插入一次

6.2插入节点：append()

向指定元素内部后面插入节点content，例如：$("div").eq(2).append(\_span);//\_span是刚创建的节点，把\_span插入到第三个div里面

通过匿名函数来向指定元素内部后面插入节点content,例如：

   $("div").append(function(index,html){//根据Index进行选择

        if(index==2){

            return html+"<span></span>"

        }

  })

appendTo():将指定元素移入到元素content内部后面，这其实是append写法的顺序变换，例如：

    var \_span=$('<span></span>');//新创建的节点

     //$("div").eq(2).appendTo(\_span)

     \_span.appendTo($("div").eq(2));//将\_span插入到第三个div内部后面

prepend(content)  将一个节点内容插入到指定内容的前面

  var \_span=$('<span></span>');

     $("div").eq(0).prepend(\_span)

prepend也可以传入匿名函数，和append一样

prependTo(content)和appendTo(content)也是一样的

append和appendTo都是往指定的父节点的末尾追加内容的，区别在于位置不一样；

prepend和prependTo都是往指定父节点的前面加内容的，区别也是位置不一样

外部插入节点的方法:after()

向指定元素的外部插入节点，after(),例如：

var \_span=$('<span></span>');

$("div").after(\_span)；   //如果不指明第几个div，会把这个\_span在每个div后面都插入一次

after方法还可以传入一个匿名函数，然后再匿名函数中判断是往第几个div里面插入

向指定元素前面插入节点:before()

var \_span=$('<span></span>');

$("div").before(\_span)；   //如果不指明第几个div，会把这个\_span在每个div前面都插入一次

before()方法里面也可以传入匿名函数，跟after一样

insertAfter(content)跟after(content)功能是一样的，只是insertAfter()是要找的元素在前面，要插入的在括号里面   顺序

insertBefore(content)跟before的关系和上面一行写的一样

包裹节点：使用字符串代码将指定元素的代码包含着，wrap()

例如：

$("div").eq(0).wrap("<span></span>")；//会在第一个div外层包裹一个span标签

如果不指定哪个div被包裹，会把页面上所有的div都包裹一次

如果传入的一个页面上已经存在的节点，会复制一份来包裹它，原先就存在的节点并不会改变

unwrp()移除包裹，只移除一层，如果包裹了多层并且向都移掉，就得移除多次。移除的时候只移除标签，不移除内容

wrapAll()   :把所有满足条件的元素外面包裹一层

                wrap():每个满足条件的元素外面都包一层自己的外衣

节点操作

复制节点：clone()

        例如：$("body").append($("div").clone(true));//把页面中的div克隆一下，添加到body中,clone()写true是把事件也复制过来

删除节点：remove()

        例如：$("div").remove();//如果不指明是哪个div，就在页面中所有div删掉

        remove()方法里面可以传入参数，表示只删除指定的元素,例如：$("div").remove("#box")表示只删除id为box的div

保留事件的删除节点：detach()；

        把一个节点删除之后添加到另一个地方上，那个节点原本绑定的事件还可以使用

        例如：

    $("div").click(function(){

          alert("大家好，又是新的一周开始了")

     })

//   var d=$("div").remove();

//   console.log(d);

     //$("body").append(d)；//添加到body上，但是div的click的处理函数没了

     var d=$("div").detach();

     $("body").append(d);    //添加到body上，但是div的click的处理函数还在

清空节点：empty()

$("div").empty();//删掉节点里的内容

替换节点：

$('div').replaceWith('<span>节点</span>');//将div替换成span元素

注意：节点被替换后，所包含的事件行为就全部消失了。

replaceAll()方法和replaceWith()写法相反，效果一样。

表单

常规选择器：我们可以使用 id、类(class)和元素名来获取表单字段，如果是表单元素，都必须含有 name属性，还可以结合属性选择器来精确定位。

例如：

                $('input').val();//元素名定位，默认获取第一个

$('input').eq(1).val();//同上，获取第二个   eq(index)找集合中的具体某个

$('input[type=password]').val();//选择 type 为 password 的字段

                $('input[name=user]').val();//选择 name 为 user 的字段

H5为表单新增了一个属性disabled，表示表单不可用

例如：<input type="text"  disabled />

作用是表明表单不可用 写法：disabled:"disabled",可以简写为:disabled

绑定事件

bind();

可以通过bind()绑定一个事件，例如：

    $("div").bind("click",function(){

          alert("我被点了")

     })

还可以把绑定的事件分出去写。一般不用，适用于事件处理函数比较复杂，例如：

    $("div").bind("mouseover mouseout",deal);//给div绑定两个事件，这两个事件执行同一个事件处理函数

     function deal(){

          console.log("我被点了");

     }

还可以通过bind()方法一次绑定多个事件，且每个事件对应的处理函数都不一样

$("div").bind(**{**  //绑定了mouseover,mouseout两个事件

     "mouseover":function(){

          console.log("我被摸了")

     },

     "mouseout":function(){

          console.log("不要走")

**}**

})

安装包：installer

卸载程序:uninstaller

bind()是绑定事件,unbind()是解绑事件处理函数，例如：

$("div").unbind("click");//解绑了div元素上的点击事件

on也是绑定事件的，off是解绑的

简写事件：一些比较常用的事件，jq单独拉出来封了个方法，

例如:click(),dblclick(),mousedown(),mouseup(),mouseover(),mouseout()等

复合事件

当鼠标移入触发第一个fn1,当鼠标离开触发fn2,例如：

    $("div").hover(enter,out);    //enter,out是两个函数

     function enter(){

          console.log("鼠标放进来了");

     }

     function out(){

          console.log("鼠标离开了")

     }

3.7 知识点集

### 事件对象

#### 事件对象

#### 冒泡和默认行为

### 高级事件

#### 模拟操作

#### 命名空间

#### 事件委托

#### on、off 和 one

### 案例

#### 数据操作

3.8 JQuery4

3.8 面试题

1.事件对象

帮我详细了解事件发生的具体信息的对象。

使用方法：

$("div").click(function(e){//通过在事件处理函数里面传入形参的方式来接收事件对象，不用考虑兼容性

        console.log(e)

})

常用的参数：type:事件的类型

                   target：事件触发的源头

                   clientX,pageX,screenX

2.冒泡和默认行为

冒泡是事件发生的时候从事件的触发点一级一级向父级传递的过程。

默认行为是一些浏览器自己帮我们设定好的效果和功能。（右键菜单，a链接，submit按钮）

阻止冒泡：e.stopPropagation()

阻止默认行为：e.preventDefault();

判断是否执行了上面两个方法，是在方法前面加  is

既阻止冒泡又阻止默认行为：return false

3.模拟操作

模拟出用户的行为操作

实现方式：trigger(),triggerHandler()

                triggerHandler模拟的行为不会触发冒泡和默认行为，而trigger会

                triggerHandler只会找符合的第一个，trigger会都匹配

                triggerHandler的执行结果不是jq对象，是事件处理函数的返回值；trigger的返回结果是个Jq对象

4.命名空间

给同一个节点绑定多个相同事件的时候，想有针对性的进行解绑，就需要利用命名空间来区分。

例如：

$("button").bind("click.a",function(){

    alert(1)

})

$("button").bind("click.b",function(){

    alert(2)

})

$("button").unbind("click.b");//这样写的只解绑第二次绑定的click事件

解绑的时候还可以写后缀名的简写，这样的话可以解绑所有后缀名为这个后缀的事件。

5.On,off,one

on,off是jq1.7版本推出来的，目的是取代bind，优势是可以实现委托绑定。

off是解绑事件，用法和unbind一样。  On 和off是一对，bind和unbind是一对。

one也是绑定事件的一种方法，目的是绑定一次性事件。绑定的方式和bind和on是一样的，只是执行一次之后就自己销毁了。

动画

1.显示，隐藏

show(),hide()

里面可以传入的参数：1、具体的毫秒

                                2.时间参数(fast,normal,slow)

                                第二参数可以传入回调函数，会在动画执行完执行

toggle()可以用来切换显示/隐藏

2.滑动，卷起

slideDown(),slideUp() 参数和上面完全一样

slideToggle()是用来切换滑动/卷起的

3.淡入淡出

fadeIn(),fadeOut()   参数参上

fadeTo(),第一个参数是时间相关的，第二个可以传入具体的透明度。执行完之后透明度只能再用fadeTo切回1

fadeToggle()是切换

4.自定义动画:animate()

参数：第一个参数是个对象，表示你想执行完动画的状态(css样式)

        第二个参数是时间，毫秒数

        第三个参数是回调函数，会在动画执行完之后执行

5.列队动画

排好队依次执行的动画

实现方式：1.一行一行写

                2。回调函数里面套另一个动画

                3.连缀

                4.queue()当动画中有颜色变化的时候，queue方法的返回值不是jq对象。可以通过往匿名函数里面传参然后再函数末尾执行的方式返回jq对象

6.动画的相关方法

stop()

第一个参数：是否清空后续动画

第二个参数：是否把当前动画直接执行到末尾状态

延迟方法

在列队动画中如果想等待一会再执行可以使用呢delay()方法

参数是等待的毫秒数

7.动画的全局属性

禁止动画：当浏览器特别破的事件可以把动画给禁了 $.fx.off=true

动画的帧数：$.fx.interval=值，默认是13.平时也不用去改

3.8 知识点集

### 动画效果

#### 显示、隐藏

#### 滑动、卷动

#### 淡入、淡出

#### 自定义动画

#### 列队动画方法

#### 动画相关方法

#### 动画全局属性

### 案例

#### 轮播图

3.9 JQuery5

3.9 面试题

扩展jq插件的作用是非常强大的，它可以节省大量开发时间

套路：  但凡要封插件，先把这个写上去

(function($){

})(jQuery)

扩展的方法：$.extend()   $.fn.方法名()

        $.extend():用于扩展函数方法，调用的时候使用$.函数名,例如：$.say()

        $.fn.方法()用于扩展jq方法，调用的时候必须基于一个jq对象,$("div").say()

好处：

     1.使用闭包：(function)($){                  }(jquery)

        a.避免全局依赖

        b.避免第三方破坏

3.9 知识点集

### 插件开发

#### 插件概述

#### 自定义插件

### 案例

#### 3D小汽车

#### 放大镜

3.10 AJAX1

3.10 面试题

1. 安装服务器端环境  
     
   服务器：能够提供某种服务的机器（计算机）称为服务器  
     
   服务器软件：使计算机具备提供某种服务能力的应用软件，称为服务器软件。  
   通过安装相应的服务软件，然后进行配置后才能使计算机具有了提供相应服务的能力HTTP  
     
   服务器：Apache、Nginx、IIS、Tomcat、NodeJS等,hbuilder/webstorm这种编辑器是自带服务器  
     
   端口：端口号是计算机与外界通讯交流的出口，每个端口号对应不同的服务。端口号由数字组成，其取值范围从0 到 65535
2. Ajax 概述  
   什么是ajax:  
   ajax是前后端数据交互的重要手段  
     
   Ajax 全称为：“Asynchronous JavaScript and XML”（异步 JavaScript 和 XML）， 使用 Ajax，我们可以**不用刷新页面更新页面内容**，并且实现异步提交，提升了用户体验。  
     
   Ajax 这个概念是由 JesseJamesGarrett 在 2005 年发明的。它本身不是单一技术，是一串 技术的集合，主要有：  
   1.通过这个XMLHttpRequest 对象，通过这个对象可以在不刷新的情况下向服务 器发送请求；  
   2.服务器上的文件，以 XML、HTML 或 JSON 格式保存文本数据；  
     
   3.其它 JavaScript，解释来自服务器的数据（比如 PHP 从 MySQL 获取的数据，node服务器）并将其 呈现到页面上。  
   1. ajax的优点  
      1.不需要插件支持（一般浏览器且默认开启 JavaScript 即可）；               //IE4，5S是不支持的    -公司最低IE8，特别坑咱们的IE6  
      2.用户体验极佳（不刷新页面即可获取可更新的数据）；     //注册页面，写个用户名，离开输入框就会发送请求在服务端验证这个用户名是否可用  
      3.提升 Web 程序的性能（在传递数据方面做到按需放松，不必整体提交）；  
        
      4.减轻服务器和带宽的负担（将服务器的一些操作转移到客户端）；
3. XMLHttpRequest         ajax的核心对象  
   创建xhr对象    
   **var request=new XMLHttpRequest();**  
   xhr对象创建的兼容性问题  
   **var** **ajax="";**  
        if(XMLHttpRequest){  
              ajax=new XMLHttpRequest();            
        }else if(window.ActiveXObject){//兼容ie低版本的  
              ajax=new ActiveXObject();  
        }else{  
              console.log("对不起，你的电脑不支持ajax");  
   **}**
4. 封装 Ajax

3.10 知识点集

### 安装服务器端环境

#### Deployd是什么

#### Deployd如何安装

#### 介绍接口

### Ajax 概述

#### 什么是ajax

#### ajax的优点

### XMLHttpRequest

#### 创建xhr对象

#### xhr对象创建的兼容性问题

### GET 与 POST

#### get方式请求Deployd的接口

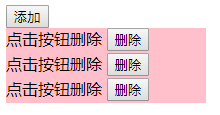
#### post方式请求Deployd的接口

#### get方式与post方式的区别

### 封装 Ajax

第二周综合案例

1. 根据题目要求实现相应效果
2. 按钮和DIV。
3. 按钮内容为隐藏。
4. 点击按钮时候div隐藏，按钮内容为显示。
5. 再次点击按钮DIV出现按钮，按钮内容为隐藏。
6. 必须使用JQuery实现上述条件。
7. 根据题目要求实现相应效果



1. 使用面向对象和JQuery完成以下操作。
2. 创建div内容为上图。
3. 点击添加按钮后，添加一项内容。
4. 点击删除时删除对应的一行。

## 3.11.第二周

3.11技能练习题

1. 用XMLHttpRequest实现与服务器的通信，将做好的页面上传至服务器，共用户查看。
2. JQAjax 中Load（）方法让Ajax异步载入一段HTML内容。
3. $get() ，$.post()， $.getJSON()，$.getScript()的应用。
4. 使用表单序列化方法.serialize()，会智能的获取指定表单内的所有元素。

3.11面试题

ajax是什么？

是现在前端与后端交互的一种手段。最大的好处在于不必刷新页面就可以更新页面内容

json是什么？

是一种专门用来存储数据的格式，里面的内容和js的对象是一样的。

在json里面可以使数组也可以使对象。json里面的所有内容整个是一个对象（可以使数组，也可以是对象）

cpu与事件处理机制

事件处理机制是js自身的特性，是用来处理事件的。js是靠事件驱动的，通过事件处理机制捕获到事件的触发，进而执行它对应的事件。

CPU是处理器，很多程序的执行都是要依赖处理器来完成的。

js中时间的绑定过程是一定要依赖处理器的，事件的处理函数是靠事件处理机制来完成的。

当已进入页面的时候，咱们的代码就交给浏览器自上而下，自左而右依次执行。比如说绑定点击事件，当浏览器执行到这一行，给相应的节点绑定点击事件，绑定的过程是不会触发事件的。只有当事件处理机制捕获到用户的点击操作，就可以执行响应的处理函数。

jq中ajax的其他用法：

$.ajax()是ajax的最底层的方法，由于参数过多，不便于使用，封了以下几个简易的方法。

load：异步载入一段 HTML 内容

post：一般用于向服务器发送数据

get：一般用于从服务器请求数据

getJSON：用于获取json文件的，

他们的参数都是一样的，第一个是请求的地址，第二个是传递的参数，第三个是成功的回调函数

$.post("[http://www.bugwhy.com/php/index/getNews.php"](http://www.bugwhy.com/php/index/getNews.php%22);, {

          "type": "html",

          "shareViewPage": 1

     }, function(data) {

          console.log(data)

     })

getScript：有时我们希望能够**特定的情况再加载 JS 文件**，而不是一开始把所有 JS 文件都加载了， 这时可使用$.getScript()方法

    $("button").click(function(){ //当点击按钮时候，加载指定的js文件

          $.getScript("alert.js")

     })

3.11知识点集

### Json

#### JSON 语法

#### 解析和序列化

### JQ中的AJAX

#### $.load()方法

#### $.get()和$.post()

#### $.getScript()和$.getJSON()

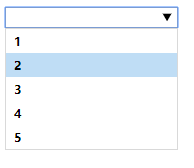
#### $.ajax()方法

#### 表单序列化

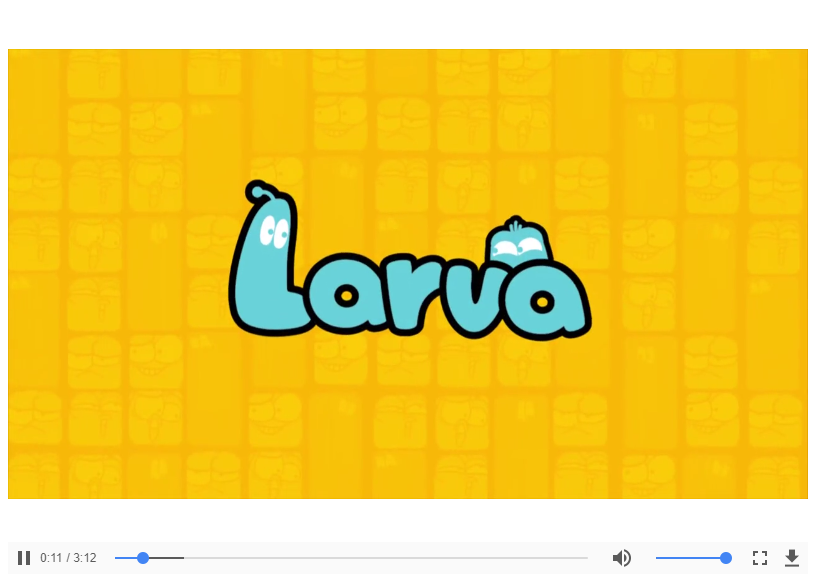
## 3.12.HTML5

### 3.12.1技能练习题

1. 创建一个表单，鼠标移动到表单内部显示下拉菜单，如下图



1. 创建一个Video 视频文件，显示在页面中如下图：



1. 使用面向对象封装一个视频播放器如下：



### 3.12.2面试题

1.概述

HTML5并不仅仅是html规范的最新版本，而是一系列现代web内容相关技术的总称

1.2html5的特点

用户至上

化繁为简：(1).以浏览器的原生能力代替复杂的 JavaScript； (2).DOCTYPE 被简化到极致； (3).字符集声明被简化； (4).简单强大的 API

引入语义：新增了一些可用性很高的标签

原生媒体支持：HTML5 的一次大改进救生衣支持在浏览器中直接播放视频和音频文件，以前都需要借助 插件才能实现此类功能。

2。新增表单属性                    这个表单必须放在form里面，当点击form里面的提交按钮的时候，会自动发起验证

2.1电子邮箱类型   email

<input type="email" name="" id="" value="" />

        这个主要验证@和.

2.2数字类型  number

<input type="number" name="" min="10" max="20" step="3" id="" value="" />

        验证表单中是否是数字。还可以通过min和max设置这个输入框中数字的范围。借助step还可以控制点击向上和向下按钮时数值变化的大小。

2.3范围   range

<input type="range" name="" min="0"  max="20" id="ran" value="30" />

    一个可以拖动的滑块，可以设定它的范围，value是初始大小，如果不写默认是中间

2.4颜色color

    这是个可以选定颜色的输入框

2.5日期   date

 <input type="date" name=""  id="" value="" />

这个输入可以通过点击选取我们需要的日期，格式是2019/9/25

2.6网址  url

 <input type="url" name=""  id="" value="" />

验证这个输入框里面的网址是否符合网址的格式

2.7搜索     search

 <input type="search" name=""  id="" value="" />

这个会把之前输入过的内容提示出来

3.新增input属性

3.1自动补全：autocomplete     这个功能是可以自己设置开启和关闭的。这个属性的值是on和off，

3.2自动获取焦点：autofocus   这个可以在一进入页面的时候就给某个输入框获取焦点。   属性值也是autofocus

3.3不验证   novalidate  规定当表单提交时不进行验证

在form标签中写：novalidate="novalidate"

3.4.新增height和width属性，只适用于input中的image类型

可以设置图像的宽度和高度

<input type="image" src="1\_4.png" width="300" height="200">

3.5.list    可以实现数据列表的下拉效果，类似select

        <input type="text" name="" list="**test**" id="" value="" />    通过输入框中的list属性的值和datalist中id联系起来

          <datalist id="**test**">

              <option value="http://wwww.xinlang.com"; label="新浪"></option>

              <option value="http://www.baidu.com"; label="百度"></option>

              <option value="3" label="谷歌"></option>

          </datalist>

3.6placeholder  提示字的显示和隐藏

一个输入框，在默认情况会显示我们写在placeholder中的提示语，当我们往输入框里面写东西的时候，提示会消失，当输入框失去焦点的时候，如果输入框为空，则提示语重新出现

<input type="text" placeholder="请输入你的名字" />

3.7.在表单中添加required="required"，在我们提交表单的时候，就会判断那个输入框是否为空，如果是空会出现提示。只有不是空，才能提交

4,video   视频

HTML5提到最多的视频处理，这也是互联网找中一个重要的部分。

新增了一个video标签来实现这个功能

video提供了播放，暂停和音量控件来控制视频，还可以通过width和height来控制视频的尺寸

video中的视频格式目前只支持三种格式：OGG,MP4,WebM,而且这三种格式的视频在不同浏览器中年的支持情况也不一样。使用video至少要提供两种视频格式的文件（OGG和MP4）

使用案例

<video width="360" height="240"   src="source/video.mp4">

  视频不能播放

  </video>

src:视频媒体的路径

autoplay:是否自动播放

width和height：设置视频的高度和宽度

利用source元素来向浏览器提供备选视频格式,source和video的src一样都是写视频路径的，soure可以写多个

  <video width="360" height="240">

  <source src="source/video.mp4"/>

  <source src="source/video.ogg"/>

  视频不能播放

  </video>

constrols:video自带的控制是否使用                 ，可以直接简介的video标签内容写上controls

例如：

<video width="320" height="180"

  src="source/video.mp4" autoplay controls>

  </video>

poster：海报功能，让视频没有播放的时候出一张图，避免看不到效果

loop:反复播放视频

muted:规定音频被静音

paused:媒体处于暂停或未播放，这个值就是true

ended：如果媒体播放结束，这个值就是true

duration:媒体的总时长，单位是秒

currentTime：获取或设置媒体播放位置

方法：

play():播放媒体文件

pause():暂停播放

5音频   audio

音频提供您了三种格式：mp3,ogg,wav，不同浏览器的支持情况也不一样

可以利用source来提供备选的格式

3.12.3知识点集

### HTML5 概述

### 基本格式

### HTML5新标签

#### 新增的结构化标签

#### 新增的非结构化标签

### 表单新属性

#### 新增的表单元素

#### 新增的表单元素的属性

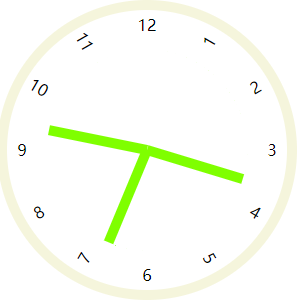
### 案例

#### 自定义视频播放器

## 3.13.css3

### 3.13.1技能练习题

1. 使用CSS制作一个时钟 ，如下：



1. 使用css3实现以下效果：
2. 使用Css3实现鼠标滑过，图片变小效果

4、使用原生JS CSS 3 面向对象封装轮播图：

### 3.13.2面试题

1.选择器

1.1相邻兄弟选择器

必须是同一个父级且紧邻在指定元素后面             -------相邻兄弟选择器  ,使用+作为结合符

                div+h2{  //div后面紧跟的那个h2

                     color: deepskyblue;

                }

1.2通用兄弟选择器

必须拥有同一个父级的某元素后面的所有兄弟元素   ----通用兄弟选择器  ，使用~做为结合符

                div~h2{//div后面所有同级的h2

                     color: deepskyblue;

                }

1.3属性选择器         跟Jq非常类似

将元素附带的属性用于选择器中，从而选择带有指定属性的元素，只判断是否具有某个属性，但是不判断属性值

                h1[class]{//选择h1中带有class属性的

                     color: red;

                }

附带id属性的元素：   [id]

附带id属性的p标签    p[id]

附带id和class的p标签   p[id][class]

附带id且id为rainy的p标签    p[id="rainy"]

属性值以"r"开头的P标签   p[class^="r"]

属性值以"s"结尾的p标签   p[class$="s"]

属性值中包含字符"s"的P标签   p[class\*="s"]

1.4目标伪类

a链接中的href指向的是页面中的某个元素，这个元素就是目标伪类。通过点击a链接来触发伪类。例如下面

            div:target{

                     background: green;

                }

           <a href="#new2">跳到内容2</a>

           <div id="new2">内容2</div>

1.5元素状态伪类

表单中的元素的状态可以用状态伪类来区分

:enabled   每个一起用的元素

:disabled  每个被禁用的元素

:checked   每个被选中的元素(单选框和复选框)

1.6结构伪类

:first-child，匹配属于其父元素的首个子元素

:last-child,匹配属于其父元素的最后一个子元素

:empty,匹配没有子元素(包括文本节点)的每个元素

:only-child:匹配属于其父元素的唯一子元素

1.7否定伪类

:not(selector) ,匹配非指定元素/选择器的所有元素

例如

                input:not([type\*="x"]){//input类型里面不包含x的所有

                     background: orange;

                }

1.8伪元素:first-letter

:first-letter用于选取指定选择器的首字母。常用于排版细节，如首字母突出显示，下沉等。

1.9伪元素::selection

::selection  用户选取的部分

例如：

                div::selection{

                     color: green;

                     background: aqua;

                }

2.内容生成

2.1 在指定节点前面插入内容 :before

                a:before { //在每个a链接内容的前面插入一个图片

                 content: url(1.jpg);//content是插入的内容，可以用""包裹具体的文字，也可以直接写图片路径

                     padding-right: 5px;

                }

2.2伪元素:after和:before

:before是插入到指定元素内容的前面，:after是插入到指定元素内容的后面

2.3伪元素：first-line  选取指定选择器内容的首行

例如：  p:first-line{          }

2.4内容content

content属性与:before和:after配合使用，来插入生成内容

常用的值：url：图像      字符串：纯文本

3.  2D转换

转换是使元素改变形状，尺寸，位置的一种效果

移动(translate)，旋转(rotate)，缩放(scale),倾斜(skew)

3.1 转换的原点

transform-orgin属性用来指定元素的转换原点位置

默认是原点是元素的中心点  ，可以用数字/百分比/关键字(right,top,left,bottom)

例如：transform-orgin(right,top);

2d转换的案例：

div:hover{

    transform:rotate(90deg) translate(200px,200px) scale(1.2);

}

4.过渡效果

过渡效果一般是通过一些简单的css动作触发平滑过渡功能。

过渡效果transition的参数:指定变化的名字(background,color);动画的总时长；动画的几种方式(匀速，加速，减速);动画执行的延迟时间

案例：

translition:background-color  1s ease 0s,color 1s ease 0s;

如果每个动画的样式都是统一的，可以直接使用all

5.给文字添加阴影  text-shadow

可使用text-shadow写艺术字

text-shadow里面的参数:阴影的横向偏移位置；纵向偏移位置；阴影的模糊半径；阴影的颜色

h1{text-shadow:2px 2px 2px red}

### 3.13.3知识点集

### CSS3选择器

#### 相邻兄弟选择器

#### 通用兄弟选择器

#### 属性选择器

#### 目标伪类

#### 元素状态伪类

#### 结构伪类

#### 否定伪类

#### 伪元素:first-letter

#### 伪元素::selection

### 内容生成

#### 伪元素:after和:before

#### 伪元素:first-line

#### 内容content

### CSS3变形效果

#### transform

#### transform-origin

#### 浏览器版本

#### 3D 变形简介 （了解）

#### transform-style

#### perspective

#### 3D 变形属性

### 过渡效果

#### 过渡简介

#### transition-property

#### transition-duration

#### transition-timing-function

#### transition-delay

#### 简写和版本

### 案例

#### CSS3时钟

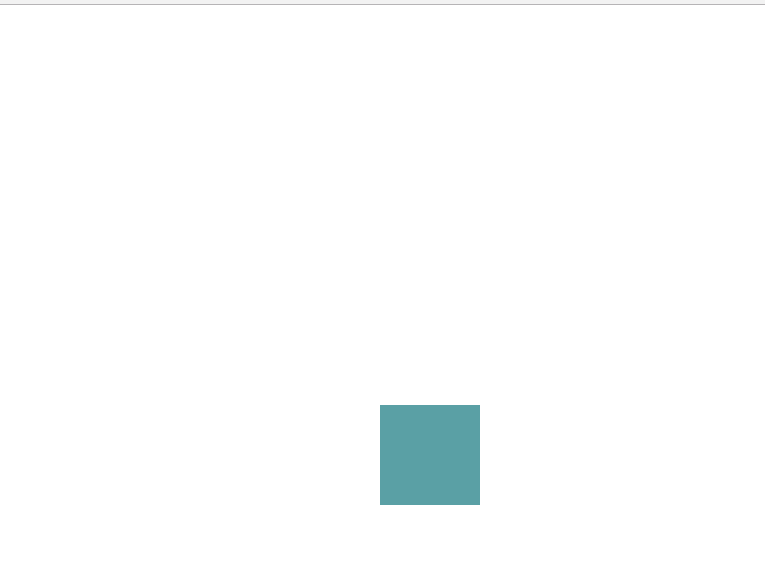
## 3.14css动画效果

### 3.14.1技能练习题

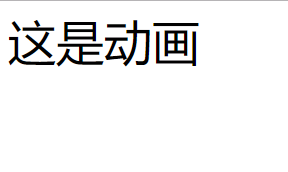
1. 用HTML5新增标签做一个进度条，如下图：

C:\Users\524316532\Desktop\新建文件夹 (2)\2017-09-01_201122.png、

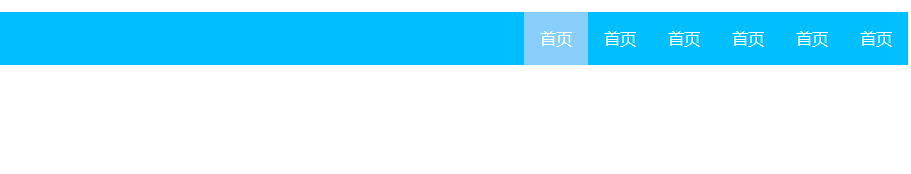
1. CSS3 媒体查询做动画：如下图

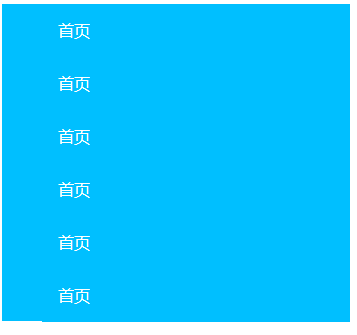


1. 用CSS3新增属性制作轮播图，如下图：
2. 引入动画文件制作一个动画效果，如下图：

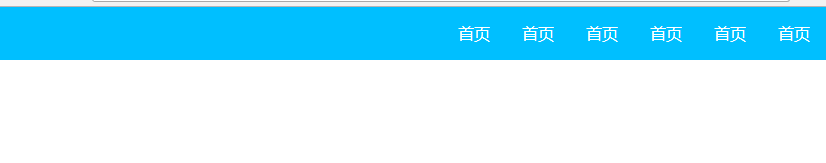


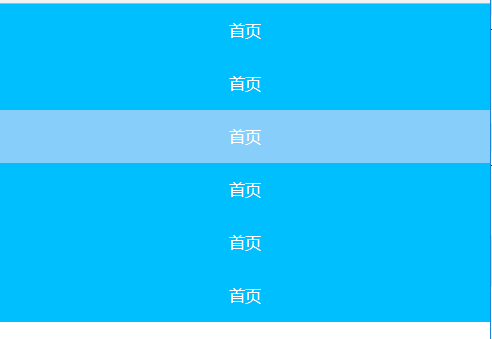
1. 使用原生JS写一个弹性盒子，如下：





1. 使用CSS3布局弹性盒子：





7、使用audio标签添加音频

### 3.14.2面试题

1.css3前缀

在css3的很多属性刚推出来的时候，还不是很稳定的时候，浏览器厂商为了实现这些属性，采用前缀的方法来实现。

各大浏览器厂商前缀的列表：

opera浏览器也支持-webkit-前缀

2.rem和em

rem和em都是相对单位，由浏览器自动转换为相应的像素值

区别：rem根据html元素中的字体大小来决定的，html中字体大小始多少，1rem就是多少

          em是使用em的那个元素的字体大小来决定的。如果不重新定义，就采用那个元素的默认值(如果没有默认值就继承父元素的字体大小)

建议使用rem

3.css动画效果

css3提供了类似flash关键帧控制的动画效果，通过animation来实现。

animate实现动画效果主要有2个部分组成：

        1.通过类似flash动画中关键帧声明一个动画

        2.在animation中年调用关键帧声明的动画

关键帧的声明：

        @keyframes 动画的名字(name){

                        0%{

                           background: green;

                           left: 0;

                           top: 0;

                     }

                     50%{

                           left: 500px;

                           top: 300px;

                           background: pink;

                     }

                     100%{

                           left: 0;

                           top: 10px;

                           background: red;

                     }

}

2.通过animation来调用动画          animation里面有好多参数

animation(动画的名字 动画的播放时间  动画的执行方式(匀速，加速) 动画的延迟执行时间   是否循环执行 方向交替 结束之后不再返回)

循环次数的参数：    默认是1 ，   infinite:无限次

方向交替：normal    默认时每次播放都是正向    alternate:方向是交替的

结束之后不再返回   ：none，按照预期进行和结束    forwards动画结束后再最后关键帧位置

案例：

animation: play 3s linear .5s  alternate forwards;

3.animate.css

这是一个设置文字动画效果的插件

使用方式：1.引入css:animate.css

                2.在需要动画的位置加类名。

                        animated:必须家的类名

                        另一个类名：我们想要的动画效果的名字。

### 3.14.3知识点集

### CSS3 前缀和 rem

#### CSS3 前缀

#### 长度单位 rem

### 动画

#### 动画简介

#### animation 属性详解

#### 简写和版本

### 第三方动画库：animate css（了解）

### 案例

#### JQ版响应式瀑布流

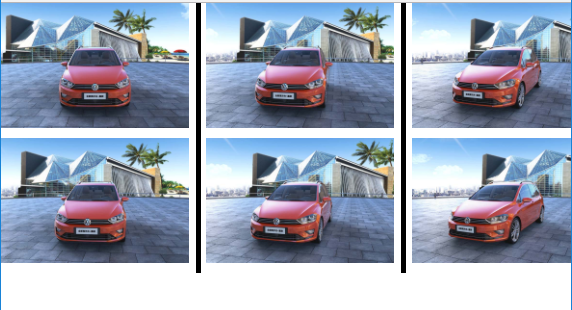
## 3.15. 响应式布局和移动端页面布局

### 3.15.1技能练习题

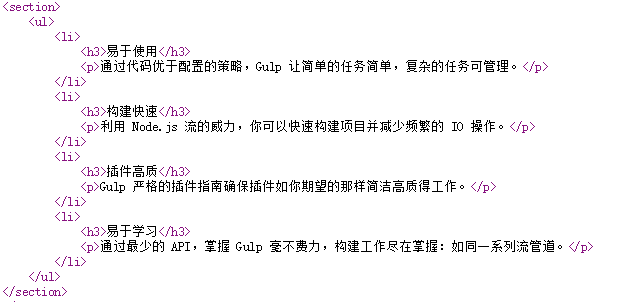
1. 用css3实现多列布局 如下



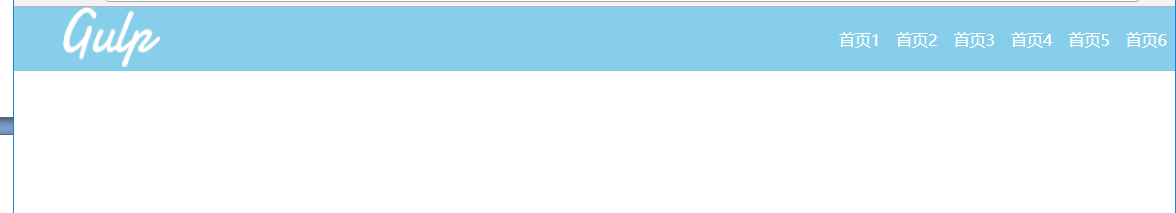
1. 使用css3实现图片多列布局加中心线 如下：



1. 在 HTML5 规范中，section 元素定义为文档中的节:如下



1. 使用CSS3实现弹性布局，使用JQ 原生JS实现点击显示弹性列表：







1. Flex项目属性的用法
2. 使用Flex项目属性实现导航栏鼠标滑过变大，缩小的效果，如下图：

### 3.15.2面试题

**Flex 布局教程：语法篇**

网页布局（layout）是 CSS 的一个重点应用。

布局的传统解决方案，基于[盒状模型](https://developer.mozilla.org/en-US/docs/Web/CSS/box_model)，依赖 [display](https://developer.mozilla.org/en-US/docs/Web/CSS/display) 属性 + [position](https://developer.mozilla.org/en-US/docs/Web/CSS/position)属性 + [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float)属性。它对于那些特殊布局非常不方便，比如，[垂直居中](https://css-tricks.com/centering-css-complete-guide/)就不容易实现。

2009年，W3C 提出了一种新的方案----Flex 布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。

Flex 布局将成为未来布局的首选方案。本文介绍它的语法，[下一篇文章](http://www.ruanyifeng.com/blog/2015/07/flex-examples.html)给出常见布局的 Flex 写法。网友 [JailBreak](http://vgee.cn/)为本文的所有示例制作了 [Demo](http://static.vgee.cn/static/index.html)，也可以参考。

以下内容主要参考了下面两篇文章：[A Complete Guide to Flexbox](https://css-tricks.com/snippets/css/a-guide-to-flexbox/) 和 [A Visual Guide to CSS3 Flexbox Properties](https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties)。

**一、Flex 布局是什么？**

Flex 是 Flexible Box 的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性。

任何一个容器都可以指定为 Flex 布局。

.box{

display: flex;}

行内元素也可以使用 Flex 布局。

.box{

display: inline-flex;}

Webkit 内核的浏览器，必须加上-webkit前缀。

.box{

display: -webkit-flex; /\* Safari \*/

display: flex;}

注意，设为 Flex 布局以后，子元素的float、clear和vertical-align属性将失效。

**二、基本概念**

采用 Flex 布局的元素，称为 Flex 容器（flex container），简称"容器"。它的所有子元素自动成为容器成员，称为 Flex 项目（flex item），简称"项目"。

容器默认存在两根轴：水平的主轴（main axis）和垂直的交叉轴（cross axis）。主轴的开始位置（与边框的交叉点）叫做main start，结束位置叫做main end；交叉轴的开始位置叫做cross start，结束位置叫做cross end。

项目默认沿主轴排列。单个项目占据的主轴空间叫做main size，占据的交叉轴空间叫做cross size。

**三、容器的属性**

以下6个属性设置在容器上。

* flex-direction
* flex-wrap
* flex-flow
* justify-content
* align-items
* align-content

**3.1 flex-direction属性    排列方向**

flex-direction属性决定主轴的方向（即项目的排列方向）。

.box {

flex-direction: row | row-reverse | column | column-reverse;}

它有4个值。

* row（默认值）：主轴为水平方向，起点在左端。
* row-reverse：主轴为水平方向，起点在右端。
* column：主轴为垂直方向，起点在上沿。
* column-reverse：主轴为垂直方向，起点在下沿。

**3.2 flex-wrap属性**

默认情况下，项目都排在一条线（又称"轴线"）上。flex-wrap属性定义，如果一条轴线排不下，如何换行。  是否换行

.box{

flex-wrap: nowrap | wrap | wrap-reverse;}

它可能取三个值。

（1）nowrap（默认）：不换行。

（2）wrap：换行，第一行在上方。

（3）wrap-reverse：换行，第一行在下方。

**3.3 flex-flow     上面两个的简写（方向，换行）**

flex-flow属性是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap。

.box {

flex-flow: <flex-direction> || <flex-wrap>;}

**3.4 justify-content属性**

justify-content属性定义了项目在主轴上的对齐方式。

.box {

justify-content: flex-start | flex-end | center | space-between | space-around;}

它可能取5个值，具体对齐方式与轴的方向有关。下面假设主轴为从左到右。

* flex-start（默认值）：左对齐
* flex-end：右对齐
* center： 居中
* space-between：两端对齐，项目之间的间隔都相等。
* space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

**3.5 align-items属性**

align-items属性定义项目在交叉轴上如何对齐。

.box {

align-items: flex-start | flex-end | center | baseline | stretch;}

它可能取5个值。具体的对齐方式与交叉轴的方向有关，下面假设交叉轴从上到下。

* flex-start：交叉轴的起点对齐。
* flex-end：交叉轴的终点对齐。
* center：交叉轴的中点对齐。
* baseline: 项目的第一行文字的基线对齐。
* stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度。

**3.6 align-content属性**

align-content属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。

.box {

align-content: flex-start | flex-end | center | space-between | space-around | stretch;}

该属性可能取6个值。

* flex-start：与交叉轴的起点对齐。
* flex-end：与交叉轴的终点对齐。
* center：与交叉轴的中点对齐。
* space-between：与交叉轴两端对齐，轴线之间的间隔平均分布。
* space-around：每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。
* stretch（默认值）：轴线占满整个交叉轴。

**四、项目的属性**

以下6个属性设置在项目上。

* order  排列顺序
* flex-grow  剩余空间的分配
* flex-shrink  缩小比例
* flex-basis   分配多余空间之前占据的空间
* flex  简写
* align-self    设置某项为特例

**4.1 order属性**

order属性定义项目的排列顺序。数值越小，排列越靠前，默认为0。

.item {

order: <integer>;}

**4.2 flex-grow属性**

flex-grow属性定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大。

.item {

flex-grow: <number>; /\* default 0 \*/}

如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。

**4.3 flex-shrink属性**

flex-shrink属性定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小。

.item {

flex-shrink: <number>; /\* default 1 \*/}

如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小。如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小。

负值对该属性无效。

**4.4 flex-basis属性**

flex-basis属性定义了在分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。

.item {

flex-basis: <length> | auto; /\* default auto \*/}

它可以设为跟width或height属性一样的值（比如350px），则项目将占据固定空间。

**4.5 flex属性**

flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。

.item {

flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ];//变大 变小 分配空间之前占据的大小

}

该属性有两个快捷值：auto (1 1 auto) 和 none (0 0 auto)。

建议优先使用这个属性，而不是单独写三个分离的属性，因为浏览器会推算相关值。

**4.6 align-self属性**

align-self属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。

.item {

align-self: auto | flex-start | flex-end | center | baseline | stretch;}

该属性可能取6个值，除了auto，其他都与align-items属性完全一致。

### 3.15.2知识点集

### 媒体查询

#### 什么是媒体查询

#### 如何使用媒体查询满足不同的分辨率

### 响应式布局

#### 什么是响应式布局

#### 如何实现响应式布局

### rem布局

#### 什么是rem布局

#### 如何实现rem布局

### 分栏布局

#### 分栏布局页面的编写

第三周综合案例

1. 使用AJIX 将一个本地文件输出到当前页面
2. 使用AJIX 将一个表格输出到当前页面

## 3.16. Bootstrap1

### 3.16.1技能练习题

1. 用XMLHttpRequest实现与服务器的通信，将做好的页面上传至服务器，共用户查看。
2. JQAjax 中Load（）方法让Ajax异步载入一段HTML内容。
3. $get() ，$.post()， $.getJSON()，$.getScript()的应用。
4. 使用表单序列化方法.serialize()，会智能的获取指定表单内的所有元素。

### 3.16.2面试题

bootstrap:

是一个前端界面框架

        必须引入bootstrap.css;如果需要插件效果必须引入jq和bootstrap.js

核心：12栅格。

          将父容器均等的分为12份，通过类名控制子元素占据父元素12份中的几份

           父容器的类名：container,container-fluid,row

           字容器类名前缀：xs,sm,md,lg  .这些前缀后面只能跟具体的数字

重点看：字体，轮播，表格，表单

swiper:专门做轮播的插件

            使用方法:必须引入swiper.css和swiper.js

                        具体见文档

作业：1.写一个好看的轮播（自己满意），要求图片路径是ajax出来的

          2.把gulp用bootstrap写

          3.把bootstrap中的界面效果和插件自己试试

### 3.16.3知识点集

### Bootstrap 介绍

#### Bootstrap 概述

#### Bootstrap 特点

#### Bootstrap 结构

### 排版样式

#### 常用的排版样式

### 表格与按钮

#### 常用的表格

#### 常用的按钮

### 表单与图片

### 栅格系统

### 辅助类和响应式工具

## 3.17.Bootstrap2

### 3.17.1技能练习题

### Gulp官网

### 3.17.2面试题

### 3.17.3知识点集

### 组件应用(导航组件)

### 插件应用（轮播插件）

### 触屏事件介绍

#### touchstart手指放到屏幕时触发

#### touchmove手指在屏幕中滑动时触发

#### touchend手指从屏幕上离开时触发

#### touchcancel事件被系统终止时触发

#### touch.js

* + 1. )

3.18

## Bootstrap项目

技能题集 apple官网

## 3.19移动端项目2

百度外卖

3.20移动端项目3

手机端京东

第四周综合案例

1. 使用css3制作轮播图

·使用纯CSS3

1. 使用css3制作一个时钟
2. 使用css3 Bootstrap 制作网页
3. 京东商城
4. 美团首页