**第一部分：ECMA**

**闭包：能够读取其他函数内部变量的函数**

var closeure=function（）{

return function；

}（）；

return function 给一个全局变量，则这个函数所涉及的变量始终在内存中。

**回调函数：**

一个函数A以参数形式作为另一个函数B的参数，在B执行之后再执行A函数。

**模块：**

本来需要声明一个全局变量对象，全局变量不好，就发明了模块，用一个函数（返回一个对象）立即执行来实现，这样对象也不能随便修改。

**原型链：**

1、每一个新创建的普通对象（比如var A={}）都有一个\_\_proto\_\_指针，指向原型对象(默认Object.prototype，也可以显性指定如 var A={\_\_proto\_\_:B),那么此时A原型链上层是B对象}。

2、每一个新创建的函数对象（比如这样创建var func={}，那么func就是这个函数对象），func也有一个\_\_proto\_\_指针指向Function.prototype这个原型对象(这个对象的\_\_proto\_\_又指向Object.prototype(最顶层))。而func.prototype则是这个func对象的原型对象(不是func对象所指向的原型对象)。

3、现在var a=new func(); 这就构造了对象a,a有\_\_proto\_\_指针，指向func.prototype这个原型对象(这个原型对象是普通对象(func才是函数对象)，则根据1中所讲，没有显性指定，则指向Object.prototype)。

4、先有Object.prototype,再有Function.prototype,再有Function,再有Object，Number,String，Date,Array等ECMAscript基本类型(其实他们都是函数，java中是类)都继承Function.prototype(本质是通过new Function()实现的)，然后Object.prototype和Function.prototype分别挂载到Object和Function下当作其原型对象。接着所有的普通对象通过new Object()构造，所有的函数对象通过new Function()构造，其余比如数组对象就是new Array(),字符串对象就是new String()，构造出来的对象所拥有的属性就是prototype所拥有的属性。（constructor可以看到构造器）。

5、通常来说，A.\_\_proto\_\_===A.constructor[‘prototype’];

**基本设计模式：**

1. 给普通对象、函数对象增加方法：这样以后新建的对象就可用method才能加自己想要的方法了。

函数对象：（普通对象换成Object即可）

Function.prototype.method=**function**(name,func){  
 **if** (!**this**[name]) {  
 **this**[name] = func;  
 }  
 **return this**;  
};

1. 用new构造一个对象（通过构造器函数，这个函数区别于其他函数首字母大写）:

**var** Func=**function**(string){  
 **this**.name=string;  
};  
Func.prototype.get\_name=**function**(){  
 //随意添加方法   
};  
**var** A=**new** Func('xyh');

通过已存在对象构造一个对象（内部还是封装new 函数）：

Object.beget=**function**(o){  
 **var** F=**function**(){  
  
 };  
 F.prototype=o;  
 **return new** F();  
};  
  
**var** A={x:2};  
**var** B=Object.beget(A);

1. 模块化设计：
   1. 建立一个对象，一个js文件中方法和变量作为其属性
   2. 大模块包含小模块
   3. 不同模块之间的变量依赖靠传入参数解决

**正则表达式：**

String： .match() 全局匹配则返回一个匹配数组，数组为所有匹配的值。非全局匹配则只检查第一个匹配的，也返回一个数组，第一个是匹配值，接下来是()中值。

RegExp: .exec() 全局匹配则一次执行可以匹配一个，接着执行可以匹配下一个（while结合），非全局则是.match一样。RegExp.test() 则返回true or false。

**变量作用域：**

Js中没有块作用域，只有函数作用域，一个函数内部总是可访问的。且写在函数里的变量声明会在函数最开始声明（声明提前），但赋值需要具体语句执行到。本质是js引擎会预编译，然后ECMA解释。

**eval(‘(’+string+’)’):** 把一个字符串string解析成js执行代码并执行

**第二部分：BOM DOM**

**浏览器构建DOM、加载文件和渲染DOM过程：**

按顺序加载文件，原则是css，影响dom结点渲染的js，在内敛js中被引用js文件放在<html>,其余放入<body>尾部。

**Html构建规则、Css基本语法：**

区分块状元素和内联元素(提倡)

1 块元素可以包含内联元素或某些块元素，但内联元素却不能包含块元素，它只能包含其他的内联元素  
2 块级元素与块级元素并列，内联元素与内联元素并列

3 css样式：

嵌入方式3种和js一样的，

伪类选择器就是给html标签的某种状态a：hover，

css是靠权值（id class 为10 标签为 1，！important最高）来判断到底选取那个样式，权值相同则会后者覆盖前者,一般就近原则赋值（权值相同情况下，内联>嵌入>外部）。

div盒子模型，直接定义的宽度长度是指内容，padding是填充，border是边框，margin是与其他元素的距离，px是像素。

流动模型Flow：网页默认的模型，块级元素宽度为100%，内联元素从左到右，高度为内容的高度。

浮动模型Float: 可以使得两个块级元素在一行显示，float：left

层模型：绝对定位（相对于上一级父节点）—position：absolute left right top bottom

相对定位（相对于本身）---position : relative left right top bottom方面理解

固定定位（相对浏览器窗口，不会因为滚动条改变）--position：fixed 如上定位

**事件处理机制：ie和其他浏览器标准（DOM）不同，区分很简单if判断有无再操作**

HTML事件添加法: onclick=”js语句”

DOM0级事件添加法: 在js中获取元素然后 .onclick=函数，取消令其为null

DOM2级事件添加法： 在js中用addEventListener添加3个参数（click（不能加on），function，false（代表可以冒泡））。去除事件可以用removeEventListener。此方法可以在事件上添加多个函数触发。

ie采用attachEvent,detachEvent和DOM2级并列。

事件触发时都会产生一个event对象，直接调用即可，其下有一系列属性作用（target、type、preventDefault等、ie则另一套标准）。event可以实现阻止默认事件，阻止事件冒泡

**第三部分：jQuery**

**区分jQuery对象和DOM对象：**

两者有各自的方法，不能混用，但是可以互相转换对象, jquery获取对象不可能为空，判断获取成功用.length>0, 而DOM可以用null判断。

var dom=$jquery.get(0);

var $jquery=$(dom); //jQuery制作工厂🡺 $()

一个是原生的js操作，一个是封装后功能更加强大的操作。**两者都是DOM元素结点对象，需要和平时的数据区分。**

**选择器概念：**

基础选择器用来找到DOM元素，规则和CSS找DOM元素一样

还有很多筛选器更加精确找到和批量操作DOM元素，非常方便代替原生的复杂操作。

注意： 两个内容之间有空格的则代表后代，无空格则是叠加选择

关于转义字符\若是在”“中，则还需要转义一次

**DOM操作:**

创建，插入，删除，替换，复制，替换，包裹，属性，样式（本质还是属性class）🡺 结点 **ps： 声明一个变量指向一个结点DOM，把其append（）之后，此DOM结点即加入HTML中，若是再次用到这个变量，那么这个对象又会append（）到其他地方，而不是复制一份，这个思想和一般编程不一样，需要注意。**

jquery中很多方法既含getter、setter：attr() html()//把参数中html元素加入this对象里面 text() val() css()等

is()函数常用判断：is(里面可以是DOM对象、筛选器(:visible)、选择器（.class）)、

遍历节点:children,next,prev,siblings,cloest(逐级向上匹配),find,filter,nextAll等这些函数参数都可以跟一个筛选器用来匹配。

**事件和动画：**

$(document).ready(function(){}))====$(function(){})

在事件中获取event只需要相应的function添加event参数即可

事件用bind（‘click mouseover‘，function（event）{//动作}）或者click（function（）{//动作}）

合成事件 hover( func , func )鼠标over 和 out

toggle(func , func)点击交换事件

自定义动画animate（//style，//speed，//callback）；动画会以队列形式存储，stop(//是否消去队列中剩余动画，//是否直接到最后状态)可用来终止当前动画，is(‘:animated’)可以判断是否有动画。

**表格表单制作验证：**

基本还是采用之前的方法，先筛选出结点，然后添加事件。

**封装的Ajax：**

传输的数据可以写成3种形式 :

一种是url形式的文本字符串:

？uname=’alice’&mobileIpt=’110’&birthday=’1983-05-12’

另一种是json数据对象: {uanme:'vic',mobileIpt:'110',birthday:'2013-11-11'}

还有一种是json数组：

[

{"name":"uname","value":"alice"},

{"name":"mobileIpt","value":"110"},

{"name":"birthday","value":"2012-11-11"}

]

//load主要从服务器端获取静态数据

(Jquery对象).load（url selector//请求获取html元素的url , 数据//给服务器端份送的数据， callback函数） 。

这个函数会覆盖（Jquery对象原来的内容，变成只有url传过来的html元素）

$.get() $.post() //get通常用于获取数据，使用url传递参数给服务器（用户名， 密码发送到服务器）；post修改服务器上资源信息，发送大小无限制（提交表单等） 。两者获取数据大小无限制。两者在服务端解析方式不变。但两者有个重大区别，post默认采用utf-8编码，然后在服务器端默认utf-8解析，get则发送时采用系统浏览器上编码，所以后台需要用对应的字符集解码成字符串。这里普及一下**编码知识**：字符串在java程序中都是以unicode编码存在，即在内存中计算时都是2个字节。对一个字符串编码则意味着用特定的字符集把该字符串(内存中)编码成字节码(二进制)，想从字节码解析成对应字符串(放入内存)必须用其编码的字符集。

$.getScript() 加载js文件 $.getJSON() 加载JSON文件

$.ajax() //里面放置一个key/value对象，{ [url:”“，type:””,data](url:):””.etc}

$.each() 会根据参数对象不同，实现不同功能，可以放入[]，也可以是{}

.serlialize() 可以把DOM元素内容序列化（就是变成url文本字符串形式name=‘’&age=‘’这种形式， 服务器端解析方式不变）

.serlializeArray() 把DOM元素结点返回JSON数组

$.param() 把json对象或者json数组解析成url字符串形式

**Jquery中异步解决方案：**

异步重要依赖：call/apply：run.apply(cat,[…]) == cat.run([…])，两者只是参数不同

函数变成异步函数很简单，执行代码中setTimeout(func)

1. 利用callback回调函数，不多解释
2. 利用 on,off,trigger这些自带函数，局限是只能操作dom元素，决解方案是$(document).on(‘done’,func)，$(document).trigger(‘done’),或者声明一个jquery对象
3. 利用pub/sub模式,封装on,off,trigger自带函数，让其变成$.属性，使其能够 $.publish(‘done’，param),$.subscribe(‘done’,func(e,param)) 如此操作.
4. deferred/promise是jquery中异步函数的解决函数，promise是deferred一个特殊对象，jquery异步函数返回就是一个promise对象。都支持如下方法：done()==resolve()触发,fail()==reject()触发,then(succ,fail),always(),process()里面的函数相当于回调函数，所以若是在回调函数取值等操作完全可以在其中中完成，且所有依赖这个数据的操作都应该定义在回调函数中，不要期待利用其返回值操作。要实现pub/sub模式，则用deferred，操作函数放在done fail process中，调用时机通过resolve reject notify控制，如下 : $.when(deferred对象).done().fail() ,这里等此deferred对象调用.resolve().reject()即可执行

$.when(可以放多个ajax，一起满足条件才调用回调)，若要回调中在回调则直接接着done/then即可，这里需要注意使用$.when()时，回调函数中是一个数组，想要得到自己的值需要data[0]才行。

**插件使用：查api文档**

1、Jquery.validate.js：表单验证 class 还有其他

Jquery.metadata.js: 结合上者更加方便属性管理 全部放在class中

还可以通过jquery中的.validate()动态添加属性要求（通过name来定位DOM）

2、jquery.form.js : 把表单提交方式升级为Ajax方式，与validate结合使用更好

ajaxForm( options ) //设置提交之前，提交之后等回调函数，

ajaxSubmit( options )

1. jquery.livequery.js : （参数只能是func）用来动态绑定事件，本来新加入的结点需要重新绑定事件，但现在和css样式一样，只要加入html中就会绑定。
2. jquery.cookie.js : $.cookie() 可以操作浏览器cookie数据
3. jquery自己制作插件时，有一套规范，还有两个制作函数，基本上靠封装对象函数实现。

**第四部分：Ajax**

1. 分为3个步骤： html+css表现层🡺XMLHttpRequest和服务器交换数据🡺js操作DOM局部更新信息。重点是操作new XMLHttpRequest（）后的对象，有一系列属性和方法。这里注意get方式，发送数据到服务器端是以url?username=’’& id=’’这样的形式传递参数，post方式则直接在send（//data）中写数据。
2. 客户端和服务器端数据传递形式：

JSON/XML/HTML/字符串（url文本字符串）: 在http中header中有一个content-type,只要客户端和服务器端协商好，就可以发送任何格式的数据，url地址用来确定发送到哪个服务器下哪个项目下，然后对应后台会根据url做出处理 ，在接收到数据的时候（通常都用request对象作为中介），按照对应数据需要的解析方式。但json解析方式对于服务器和客户端都比较方便。

1. JSON语法主要是两块，一种创建对象，另一种对象数组。

创建对象：早期js中创建对象需要var p=new Person(name,age),后来就用json格式创建var p={“name”:”xxx”, ”age”: 18}

创建数组：早期数组var a=new Array(‘x’,‘y’);后来使用json格式创建var a=[‘x’,‘y’];

1. Ajax请求会在浏览器中存在缓存，所以改变url时需要清除缓存

**第五部分：requireJS**

解决的问题：全局变量污染问题+js文件异步加载依赖问题，主要针对自己写的不同模块，外部模块(jquery,bootstrap等)依旧以标签形式引入。

使用方法：首先模块符合AMD规范，然后在require.config中注册，require回调函数中使用。

注意事项：

1. AMD规范中return{ }，若是想修改得到一个变量，此变量会经过加工，则需要用函数返回get/setVariable()，否则只会返回undefined,也就是一开始定义的值
2. 定义模块时，尽量减少互相之间的依赖关系，要是有依赖则利用参数传递