**H5和CSS3新增内容（自己总结）**

HTML5新增标签及属性

CSS3新增样式

Web API

**选用标签的标准**

如果要支持比较老的浏览器（通常是桌面版网页）则不使用新标签

手机平板等网页开发可以使用新标签

不管什么情况下，都应该优先根据语义来选用标签

根据搜索引擎优化的要求和建议选用标签

**布局**

流式布局

定位

浮动/清除

Flex

布局框架（如Bootstrap栅格）

**CSS单位与计算**

px

%

rem/em

vw/vh

mm/cm

calc()

**CSS半透明/透明**

大部分标签元素默认是背景透明的，但也有不透明的（如iframe、button、图片等）

transparent

RGBA

HSLA

opacity

**动画**

transition：transition能过渡大部分样式（<http://dwz.cn/3NvjUN>）而且支持多样式同时过渡，注意标签元素上有多个样式类时，与过渡样式相关的最后2个样式会形成过渡效果。

animation：注意使用animation-play-state:paused可以使动画就地暂停，animation-play-state: ‘’可以使动画在暂停的状态继续进行

jQuery动画：jQuery动画与CSS3动画不同，jQuery动画是js驱动的，jQuery动画与CSS3动画是互补关系（而非谁取代谁）。jQuery内置了一些动画，如（fadeIn/slideUp等），也支持通过animate()方法自定义动画。而且提供了强大的动画控制API，如动画延迟、链式动画（上一个动画结束后开始下一下动画）、控制动画停止、关闭动画效果、动画频率调整、还可以指定回调函数参与每一个动画的每一步等。jQuery中的动画返回Deferred对象，因此支持Promise异步编程。jQuery默认不能过渡颜色，使用jQuery.Color插件可以解决问题。

触发动画：CSS一般在页面加载时会自动进行transition和animation，可以在浏览器的查看器（动画窗格）中观察到标签元素上的动画帧并对动画进行控制及设置不同的速度进行预览。使用:hover、:active伪类可以配合鼠标指针触发动画效果。通过js添加删除标签元素的样式类时也会触发相应的动画。jQuery动画的触发和控制由jQuery动画API控制。

动画结束事件/回调：CSS3动画支持DOM事件：animationstart（动画开始时）、animationend（动画结束时）、animationiteration（动画重复执行时），可以像普通click事件一样使用它们。jQuery动画通过回调函数获取执行完成通知，形如：$(..).animate({css}<, duration><, easing><, callbakc>)，因为jQuery动画返回Deferred对象，所以也可以使用.then(success,fail)或.done(function)等方法得到通知。

同时进行多个动画：在关键帧定义中可以定义多个样式，animation-name也可以指定多个关键帧动画名称。transition-property也可以指定多个样式。jQuery的animate方法中也可以指定多个样式。还可以在同一个标签元素上调多次animate方法（分开调用，而不是链式调用）

顺序进行多个动画：可以通过animation-delay、transition-delay延迟，安排出顺序执行效果。还可以通过监听动画事件在动画结束后再触发下一个动画。jQuery中animate()的链式调用本身就是顺序执行的。

requestAnimationFrame：会与浏览器呈现页面同步配合执行，因此比使用setInterval要更加流畅。requestAnimationFrame是H5新增的方法，与setInterval相比，没有提供频率的控制方法，因为它的频率与页面呈现频率是同步的，通常60次/s，相当于16.66667ms一次。

矩阵与图形变换

贝塞尔曲线与时间函数

**媒体查询与响应式页面**

媒体查询@media赋予CSS探测设备类型和设备属性（如屏幕宽高等）的能力，之前媒体查询主要用来控制页面打印效果（即针对页面上打印部分定义打印专用样式）；在移动互联网时代媒体查询对于页面适配各种大小的屏幕至关重要，可以说媒体查询是响应式页面的核心技术。Bootstrap就是以媒体查询技术为基础的响应式UI框架。

**浏览器兼容性和Web App页面**

浏览器兼容性问题涉及的面比较广，从HTML标签、CSS样式到JS都存在兼容性问题。兼容性问题给Web开发造成了额外的成本支出，在兼容性上要求越高，付出的代价越大。

浏览器的兼容性主要通过测试解决，桌面网页最为困难的是兼容IE6/7/8/等老浏览器（可以使用IE Tester或IE11的开发者工具测试）！解决浏览器兼容性没有什么难度，繁锁而已，如果要兼容老浏览器，则应放弃使用新功能或者对老浏览器显示友好的提示信息（使开发测试时间增长、成本增加）。

手机浏览器中最为困难的是Android版的UC浏览器，Android版的UC几乎不支持H5和CSS3，甚至是常用的CSS样式有些都不支持，如果项目要兼容UC手机浏览器，那写CSS时，就只能使用最基本的样式！也不要指望页面能做多漂亮多复杂了。

Android系统上除了UC这个奇葩之外，本身的碎片化、更新慢也是大问题。如很多手机现在的Android系统还是4.4，甚至更古老（尽管Adnroid7都发布了）。如果开发应用内嵌页面，可以建议Android工程师采用腾讯出的WebView控件，能够使页面对H5的支持和微信及手机QQ一样好，从而达到和iOS同级别的效果。

直接使用框架可以有效地避免兼容性问题，如各种UI框架和jQuery框架都会考虑兼容性。另外使用重置样式表也可以抹平各浏览器的差异。

在开发Web App时需要注意，移动设备，无论是iOS还是Android都使用Webkit内核，尤其是iOS，需要使用以下样式：

-webkit-appearance: none; 禁用浏览器特定外观（如按扭等表单元素）

-webkit-tap-highlight-color: rgba(0,0,0,0); 将触摸时的高亮颜色设为完全透明

-webkit-touch-callout: none; 禁用页面上的上下文菜单（上面有复制、粘贴等功能）

-webkit-text-size-adjust: none; 禁用页面自动调整文字大小

-webkit-user-select: none; 禁止用户选择页面上的文字

回弹滚动

overflow: auto; /\* auto | scroll \*/

-webkit-overflow-scrolling: touch;

另：有一些样式还需要使用-webkit前缀，如：

CSS滤镜要写成-webkit-filter

或者要兼容特别老的iOS或Android系统时CSS3样式要加-webkit前缀

另：如果需要对某些标签进行深度控制，则需要使用-webkit伪类，如：

input[type=range]::-webkit-slider-runnable-track{ }

input[type=range]::-webkit-slider-thumb{ }

可以控制滑动条的样式

progress::-webkit-progress-bar { }

progress::-webkit-progress-value { }

可以控制进度条的样式

另：使用meta标签可以要求浏览器支持或关闭某些行为

<meta name="format-detection" content="telephone=no" /> 禁用手机号探测

<meta name="format-detection" content="email=no"> 禁用email探测

固定到手机桌面（桌面书签）

<meta name="apple-mobile-web-app-title" content="常伟">

<link rel="apple-touch-icon" sizes="57x57" href="icon57.png" />

<link rel="apple-touch-startup-image" href="launch6.png">

<meta name="mobile-web-app-title" content="常伟">

<meta name="mobile-web-app-capable" content="yes">

关闭iOS键盘首字母自动大写，自动修正

<input type="text" autocapitalize="off" />

<input type="text" autocorrect="off" />

**LESS/SASS/SCSS**

Less 是 CSS 预处理语言，在CSS中引入了变量、函数、混入等编程机制，从而使CSS获得更高的灵活性和可扩展性。Less可以使用js直接在页面上转换成CSS，也可以使用命令行工具Lessc完成从less文件到css文件的转换，还有相应的自动化构建插件。注意Less中使用calc()或滤镜等函数时需要使用 ~””或e(””)进行转义，否则会被误识别为less函数，而less中并没有这些函数。

Sass （Sass3时提供了兼容CSS的语法，被称为SCSS）是对 CSS 的扩展，除支持变量、函数、混入外还支持条件判断和循环语句，Sass让 CSS 语言更强大、更优雅。

值得一提的是，Less是基于Node.js和js的，而Sass/SCSS是基于Ruby的（也需要Node.js），Sass/SCSS不能直接在页面上通过js转换成CSS。

**JS基本数据类型（typeof的返回结果）**

number（Infinity/NaN）

string

boolean

function

object（null、各种值装箱对象、内置对象、自定义对象）

undefined

判断对象是否为某个【类/构造函数】的实例（注意继承）用instanceof（对象与函数的关系）

获取对象的原型用Object.getPrototypeOf(obj)，如果原型中有constructor则可获得构造函数[获得的是函数而不是函数名字]

判断A对象是否为B对象原型链上的一环A.isPrototypeOf(B) （对象与对象之间的关系）

**原型与原型链**

如果在构造函数中直接为对象添加属性和方法，则会导致每个对象都包含一份自己的属性和方法，当大量创建对象时就会浪费很多存储空间。而将属性和方法添加到构造函数的原型中就可以使通过该构造函数创建出来对象共用这些属性和方法。即原型实现了属性和方法的**共享**。

为构造函数指定原型：

ObjFunc.prototype = object.create(baseFunc)

ObjFunc.prototype.constructor = baseFunc

ES6中支持 class 和 extends 关键字，就不需要用上面的方式了（但本质上还是一样的）。

当访问对象的属性和方法时，JS运行时会首先在该对象自身中查找要使用的属性和方法，找到就调用；如果找不到，则会到原型中查找。因此原型中的属性和方法虽然是共享的，但它们的优先级要低一点。

JS运行时在查找可调用的属性和方法时并不只查找一层原型，因为原型本身也是对象，因此JS运行时还会查找原型对象的原型，一直查找到Object的原型为止。这样就构成了一个链式结构，称为原型链。通过原型链可以形成类似**继承**的效果。

但JS中的继承与其它编程语言中的继承还是有区别的！在其它编程语言中，任何类的对象都是由一个实例构成的，属性值也只有一个值，不管继承多少层。而JS通过原型链形成的继承是由多个实例构成的链，属性值也可以有多个值（每一层原型上都可以有一个不同的值），继承的层次越多，涉及的实例就越多，值也越多！

JS的原型链在Angular中还被用于构造作用域链。

**this**

js中的this与其它编程语言中的不同，js中的this指向的对象不是固定的，而是可以改变的（通过function对象的call、apply、bind，jQuery的proxy等可以改变this指向）。以下是this的默认指向：

1. 定义在全局作用域中的普通函数中的this指向window对象（严格模式下本条无效，参见后面严格模式的说明）
2. 事件处理函数中的this指向触发事件的标签元素
3. 构造函数中的this指向当前正在创建的对象（严格模式下必须使用new）
4. 原型中的函数内的this指向当前实例
5. 其它框架中的函数内的this都有特定的指向，如jQuery.each(func)中func中的this指向当前迭代的标签元素对象。

**DOM**

文档对象模型，包括document本身和页面上的各种标签元素。DOM是访问和控制文档及标签元素的API，通过DOM可以查找、修改、添加、删除标签元素，还可以添加事件监听函数以响应页面上的各种事件。H5新增了一些DOM API（如querySelector）在一程度上增强了DOM，但实际开发中，使用jQuery进行开发还是非常普遍的。因为jQuery不仅提供了强大的DOM操作能力，而且能够解决浏览器兼容性问题。此外jQuery还提供了Ajax请求、动画等实用功能，还提供了Callbacks、Deferred等高级API。

**BOM**

浏览器对象模型，包括window、location、history、navigator等

window对象是全局作用域，window对象会随页面的加载或刷新而重置，即全局作用域生命周期相关的内容都会被销毁。window对象提供了窗口相关的API，用于控制窗口及窗口与窗口之间的关系。window对象还提供了XHR、setTimeout、alert、WebWorker等各种编程接口和功能支持。实际上从window对象出发可以访问几乎所有的Web API。

location提供了与url相关的API，通过location提供的属性可以获取url的各个组成部分。location还提供了对页面进行导航控制的基本功能，如对href赋值、assign()、reload()、replace()方法等。值得注意的是location提供了hashchange事件，可以监听hash fragment的变化，这对于单页Web应用中的路由机制提供了重要的支持。

history对象提供了浏览历史记录控制功能，如back()、forward()、go()等，H5还新增了状态相关的API：state、pushState、replaceState配合window的popstate事件可以为路由机制提供重要的支持。

navigator对象提供了浏览器和操作系统信息的描述，如userAgent、H5还在navigator中加入了很多设备访问功能，如geolocation、battery、vibrate等（浏览器目前支持不太好）。

**JS语言核心对象**

包括Object、Function、Array、与值类型对应的包装对象（Boolean、Number、String）、Math、Date、Error、Regexp、JSON，这些对象是JS编程语言自带的，无论JS在何处（如Photoshop、Mongodb等）运行都会包含这些对象。

**Map/Reduce**

js中的数组支持map/reduce（ES6）。map可以对数组中的每一个元素进行加工，然后返回一个同样长度的**数组**，reduce可以将数组中的每一个元素都拼在一起生成**一个**最终结果。

Map/Reduce是一个非常经典的编程思想，广泛应用于数据处理（如大数据技术），Map/Reduce的好处是能够将数据映射与数据归结分离开，有利于简化问题，在大数据技术中还可以分布式计算提高数据处理速度。

因为Map返回的还是数组，因此可以构成如下的链式调用：

data.map(function(i){ }) .map(function(i){ }).reduce(function(prev, current, seed){ })

**图片预加载和懒加载**

图片的可以为页面带来更好的展示效果，同时也能向用户传达更丰富的信息。在页面上展示图片时有两类典型问题：

* 1. 图片最初是隐藏的，但在需要显示时应能立即显示出来（如轮播图、相册展示、幻灯片）
  2. 网页很长，而且网页上有大量的图片（如电商网站中的商品列表），但这些图片未必用户都会看到，如果用户根本就不滚动页面来查看这些图片，那这些图片的加载就会成为一种浪费。

解决这两类问题就需要使用图片预加载和懒加载技术。

图片预加载是在图片显示之前就让浏览器加载这个图片，这样当浏览器真正需要显示这个图片时就能瞬间将它显示出来，不需要再去下载和解码。图片预加载通常使用Image对象将图片载入内存。Image对象实际上就是一个<img>标签，只不过没有append到页面上而已。

图片懒加载则是让网页下部分的图片都先显示同一个loading图片，当页面向下滚动，这些图片露出来时，才将img标签的src改为正确的图片地址，这时浏览器才去下载图片并解码显示到页面上。图片懒加载的关键是监视页面的滚动及窗口大小变量并计算图片是否处于窗口工作区内（即用户能看到）。有很多图片懒加载插件或独立脚本，如：

jQuery.lazyload.js

echo.js

**Ajax/XHR/HTTP/jQuery Ajax**

Ajax即通过XHR API使用js发起的异步网络请求，它不会导致页面刷新，因此是现代Web App的关键技术。

HTTP协议是Web开发中最重要的网络协议，HTTP协议详细规定了请求和响应报文。

请求报文由4个部分构成：

* 1. 请求行：包含请求方法和URL
  2. 请求头：包含HTTP协议中定义的请求头和自定义请求头（每个头1行）
  3. 空行：分隔请求头与请求体（由协议实现方控制，没有提供API）
  4. 请求体：随请求发送到服务端的数据（可多行）

常用的请求方法是GET和POST。GET没有请求体，数据只能放在URL中，因些可以发送的数据很少，而且用户能够通过浏览器地址栏看到发送的数据。POST请求可以通过请求体发送大量的数据，可以支持各种数据编码方案（需要用Content-Type头指明），常见的有urlencoded、json、xml、text等。其它请求方法在RESTful中常用。

请求头中有一些与协议相关的重要数据，如Content-Type、Content-Length、支持断点续传的Range、携带浏览器信息的UserAgent、还有常用来存储用户身份信息和偏好设置的Cookie等

响应报文也由4个部分构成：

1. 响应行：包括响应状态码、和状态描述、HTTP版本（最新2.x只用于https，http最新1.x，广泛使用的是1.1）
2. 响应头
3. 空行
4. 响应体

XHR能够灵活地控制HTTP的请求和响应报文：请求行在open方法中控制，请求头通过xhr.setRequestHeader('name', 'value')设置，请求体通过sned方法发送。通过xhr对象的status属性和statusText可以获取响应行信息，通过xhr.getResponseHeader('name')可获取单个响应头，通过xhr.getAllResponseHeaders()可获取所有响应头。通过xhr.response、xhr.responseText、xhr.responseXML可获取响应体。

XHR还支持很多事件，可以监控请求/响应状态的变化，如：

readystatechange最常用

loadstart

progress

abort

error

load也比较常用，尤其是针对现代浏览器（它的使用更简单）

timeout

loadend

jQuery中也包含了对Ajax的强大支持，jQuery框架能获得巨大的成功与其对Ajax的卓越支持是分不开的。在jQuery Ajax是对XHR的封装，它提供了丰富的高级API，可以很方便地发起各种请求，很方便地控制和获取请求/响应报文的各个方面，监听请求/响应状态的变化（包括单次请求和全局监视）

$.ajax({请求配置})

$.get(url, data, success, type)

$.post(url, data, success, type)

此外，jQuery的Ajax还支持JSONP（$.getJSON(url, data, success)），支持直接加载HTML到页面$('div').load(url, data, success)，支持通过代码加载js脚本$.getScript(url, success)。更值得称赞的是，jQuery的Ajax返回的是Deferred对象，支持Promise异步编程！

**REST**

在常规网络请求中，通过在URL中插入【动词】来【表示】不同的功能

获取数据： /api/card/id

保存数据： /api/card/【save】/id

添加数据： /api/card/【add】

删除数据： /api/card/【remove】/id

更新数据： /api/card/【update】/id

REST 表达（表示）状态迁移【转移】

把上面表示不同功能的动词，从URL中转移到请求行的请求方法中

原先的请求行：

POST /api/card/【save】/id

上面这种写法没有注意到HTTP协议中请求方法的作用，到处使用POST/GET这两个请求，把这2个请求方法当成是万能的，在URL中插入各种各样的动词来表示本应该由请求方法表示的功能，这种写法还对URL产生了不良的影响：

1. URL是一个地址，地址中应该使用名词，而不应该出现动词！
2. 插入动词之后，本来是同一个资源，只需一个地址，现在变成一堆地址！

按照REST风格改造之后：

获取数据： GET /api/card/id

添加数据： POST /api/card

更新数据： PUT /api/card/id

删除数据： DELETE /api/card/id

URL中的动词被删除了，一个资源只有一个URL【表示功能的动词从URL中转移到了请求方法中】，请求方法是HTTP协议规范定义的，所以大家都使用一组一致的动词，不需再自己创造各种各样只有自己能看懂的动词了。

现代的应用程序框架中普遍支持RESTful，如Backbone和Angular($http、ngResource)。

**跨域访问与同源策略**

因为在同一个浏览器窗口中能够同时打开多个网站的页面，而且它们都处于同一个会话中，如果不禁止跨域访问则会造成用户隐私数据泄露和登录身份冒用的问题，所以浏览器会使用同源策略限制跨域访问。

在浏览器中，通过JS代码访问不同域名下的URL（JS的XHR/AJAX）或者iframe（JS访问iframe内部的DOM）时，会被禁止访问。而不是通过JS代码进行的跨域访问不存在跨域问题！比如可以跨域加载图片、引用JS文件、下载各种文件、使用iframe跨域嵌入其它网站的页面都是允许的。

跨域访问被禁止有时会给应用开发带来阻碍，但在符合特定条件时也有相应的方法在保证安全的情况下能够解决跨域访问的问题。

1. 在对方的服务器中的响应头中添加 Access-Control-Allow-Origin 允许哪些域进行跨域访问它是值可以是 域名，或者 \* 。这种方案只有在对方信任或不在乎（安全）的情况下才能使用。

**在用Ionic开发Web App时，通常会遇到浏览器端和服务端分离的情况，这时可使用CORS浏览器插件自动在响应头中添加Access-Control-Allow-Origin头。这样可以轻松实现开发时的跨域。**

1. 如果域名都是同一个根域名的子域名，则可以使用document.domain = "根域名" 来统一JS执行环境中的域名。这种方案只能在同一个公司或组织的内部使用。
2. JSONP JSON Padding，原理是：浏览器不限制通过script标签引入其它网站的脚本，所以可以通过JS向页面上动态添加一个script标签并且指定其 src 为一个特殊的url，对方的服务器针对这个url的请求，会进行特殊处理，如：

向head标签中动态添加以下script标签

<script src="http://api.baidu.com/weather/zhengzhou/functionName"></script>

会导致浏览器向上述URL发起一个GET请求（JSONP只能是GET请求）

对方的服务器收到这个请求后，会返回一个特殊的JS文件:

functionName && functionName({

天气数据

})

如果此时在页面中定义了functionName函数，则functionName函数会被调用，并且能够得到天气数据！

这种将JSON数据放入指定函数参数位置的跨域访问解决方案被称为JSONP。即使用JSON填充函数后面的（）内部的空白（padding）

这种方案可以跨域任意域名，但是必是对方有意这样设计才能使用。如果对方不支持将JSON数据padding到函数名后面的()中，则JSONP无法使用。

**在jQuery中 $.getJSON() 这个方法支持 JSONP !!!**

在url后面加 callback=? 即可，jQuery会自动生成函数名并将调用转交给getJSON中的回调函数。

**$.ajax()方法也支持JSONP**，设置type:’GET’, dataType:’jsonp’即可

**Angular中的$http服务也提供了jsonp()方法支持JSONP**

1. 将要请求的URL发送给自己的服务端，让服务端发起请求（服务端没有跨域限制），服务端请求成功后，将数据再回传给浏览中的JS----服务端代理请求。

这种方式只要自己的服务端支持一下就可以了，是比较常用的方案，没有任何限制，而且这种方安还可以实现其它方案无法实现的功能：

通过服务端抓取别人的网页，将网页上的数据提取出来，变成JSON返回

在Node.js中，使用cheerio模块可以像使用jquery一样从HTML字符串中筛选并提取想要的数据。

1. 使用任何可以利用的浏览器端中间机制实现跨域交换数据（），如：

window.name在代码中使用name变量时实际上使用的是window对象的name属性，但是name属性是window对象的内部属性。它只接受字符串值，如果给它赋其它值，将会直接被转换成字符串！！！！尤其是赋一个对象给name变量的时候，会导致数据丢失！！！(对象toString()后是[object Object])。但是name有一特别性质可以被用来做跨域数据交换name值不会随全局作用域被销毁，不管窗口跳转到哪个页面，不管窗口打开了多少个页面，name的值都是通用的。其它的，诸如 location.hash 也可以用来做跨域数据交换（主要是iframe）

**路由**

在传统的由多个页面构成的网站中不存在路由的需求，那时通过链接进行跳转（导航）就能满足需求，但是随着Ajax和移动互联网的发展，单页应用越来越多，在单页（通过1个URL打开一个页面）中模拟多页面切换没有问题，但问题时怎样构造出一个能够通过URL实现“页面切换”并且这个URL还要满足**可链接、可收藏、可分享**的特性就成了一个有难度的技术问题。现在成熟的解决方案是利用url中的hash fragment来的变化来触发“页面切换”，这种将url(hash fragment)与“页面”映射起来的机制被称为路由（Router）,路由中的关键技术是**监听url(hash fragment)的变化**，支持hashchange事件的浏览器很容易做到这一点，不支持hashchange事件的浏览器通常需要使用setInterval大概以100ms/次的频率轮询检查url的变化。很多应用框架都支持路由，如Backbone和Angular（除ngRouter外，还有uiRouter支持更强大的功能），React也通过React-Router提供路由支持，另外还有很多支持路由功能的独立脚本库如：router.js、director.js等

**触控与手势**

随着移动设备的广泛应用，对触屏的支持势在必行。

H5中新增了Touch API来支持触控，包括以下几个类（对象）：

TouchEvent表示触控事件，类似于普通的事件（如click）对象，只不过比普通事件对象多了touches、targetTouches、 changedTouches属性。

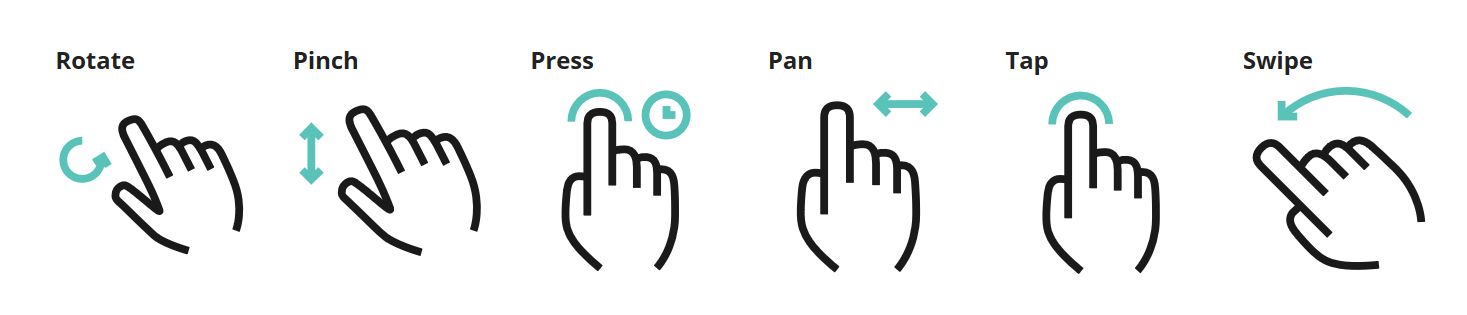
Touch表示一个触控点（一个手指与屏幕的接触）包含identifier（唯一识别号）、screenX、clientX、pageX、target等，甚至还包含radiusX（接触半径）、rotationAngle（旋转角度）、force（力度）等属性。

TouchList表示一组触控点（多点触控）其中有多个Touch对象，它自身有length（触控点个数）、item(index)（获取指定触控点）等属性。

触控事件类型有touchstart（手指开始接触）、touchmove（手指在屏幕上移动）、touchend（手指离开屏幕）、touchcancel（受来电话或电量不足提示等影响而导致触控被中断）。监听触控事件和监听普通事件一样，使用addEventListener()即可。

对于触摸屏来说，触控事件是优先的，而传统的点击事件是由触控事件转换而来的，即用户接触屏幕后迅速触发touchstart事件，300ms之后，系统才会触发click事件。因此在开发手机页面时，如果简单地使用click事件则会带来一种轻微的延迟感。很多框架都专门针对这种延迟给予了优化，还有一个叫faskclick.js的脚本专门解决这个问题。

上面的基本的触控事件比较底层，使用起来不是很方便（需要监听多个事件），尤其是应用开发中如果能够直接使用**手势**则更加高效。手势是一些常用的触控操作手法，是单点或多点触摸组合加上特定的移动过程构成的。常见的手势有：



常规的框架只能支持部分手势。如：

angular应用框架中官方提供了一个ngTouch模块，这个模块提供了ngSwipeLeft和ngSwipeRight指令可以像普通HTML中的onclick一样使用，分别表示向左轻扫屏幕和向右轻扫屏幕。另外该模块还有一个$touchProvider，可以解决ngClick的300ms延迟问题（与faskclick.js作用相同）。

ionic UI框架也提供了一些手势支持，比angular支持的多一些，不过还是以单点触控为主。如swipe轻扫可以支持上、下、左、右4个方向。还支持4个方向拖拽drag。支持长按hold，轻点tap，双击double-tap等。ionic中的列表（List和Item）已经集成的常用的左右滑动显示操作按扭的功能，ionic的侧滑菜单本身也支持手势操作。

如果要支持更复杂的多点触控手势，就需要使用功能更多的脚本库了，如：

Hammer.js <http://hammerjs.github.io/>（最强大，它还提供了jQuery插件、angular插件）

Touch.js <http://touch.code.baidu.com/> （来自百度，比Hammer弱一点）

**ES6**

ECMA于2015年6月发布的JS标准（也被称作ECMA2015，该版本于2016年6月小幅修订）、目前在Node.js和桌面浏览器上已经完成了支持（主流的现代浏览器都支持），移动端的支持较弱（支持约50%的特性，Android由于碎片化严重，老机型支持更差），目前获得最广泛支持的是ES5.1。ES7预计在2017年发布。

ES6为JS带来了大量的新特性，是JS语言升级最大的一个版本：

* 1. 支持let块级变量和const常量
  2. 支持变量解构赋值
  3. 扩展了JS的核心对象
  4. 增强了语言反射能力（通过代码获取代码自身信息的能力，如一个类包含哪些方法）
  5. 增加了Promise
  6. 支持Class定义类
  7. 支持Module模块化
  8. 支持迭代接口和生成器等

由于ES6现在还没有被所有浏览器支持，如果想在项目中使用ES6，而且还想保证浏览器兼容性，就需要使用转换工具将ES6转码成ES5。常用的转换工具有：

1. Traceur：由谷歌提供，可以在网页上就地转换，也可以在线转换、还可以使用Node.js命令行本地转换；
2. Babel：支持ES6**语法**转换，如果要支持新增加的API，则还需要使用babel-core。Babel目前由Facebook支持（因为Facebook的React给JS扩展了一套新的组件语法jsx，而这个语法需要babel来转换成普通js）。Babel推荐使用webpack之类打包工具进行转换，Babel也提供了babel-standalone支持浏览器转换，但不应在商业正式环境中使用。

上述两个工具的使用可参考：<http://t.im/16ax8>

**模块化**

JS的作用域隔离机制较弱，因此使用不当时会产生命名冲突的问题，尤其是在单页应用日益复杂、使用的第三方框架越来越多的情况下更容易产生冲突问题。而且JS本身没有直接提供使用代码加载其它脚本文件的机制。因此产生了JS模块化概念。

JS模块化通常要解决2个问题：

1. 提供定义JS模块的办法
2. 提供按需加载模块及其依赖模块的办法

常用的JS模块化方案有以下几种：

1. Node.js中的模块化方案（通常称为CommonJS规范，最简单易用，通过约定取消了配置）
2. 浏览器中使用中RequireJs（称为AMD规范）和SeaJs（称为CMD规范），这两个都需要配置
3. ES6中的模块化方案（无需配置）

目前在浏览器端使用最广泛的是RequrieJs（jQuery、Underscore、Backbone、Angular、ionic等流行的库和框架都支持），RequireJs通过AMD规范描述了定义和使用模块的方法，本身支持JS文件的按需加载，另外RequireJs还有很多官方和第三方插件可以实现各种文件（如HTML、CSS、其它文本文件等）的动态加载。

另外还需要注意的是Webpack打包工具支持混合使用Nodejs中的require()导及ES6中的import … from …，还支持直接从npm模块包中导入js，因此使用Webpack + npm也越来越流行，尤其是在React阵营中几乎成为主流。

**组件化技术**

自从HTML产生之后，就以其简单易出成果而被广泛应用，HTML的功能也越来越强大，标签数量也越来越多，到H5标签已经超过100个。但是希望自定义HTML标签的人并不会因为HTML标签越来越多而止步，他们的期望更高，那就是可以自由地增加HTML标签。于是产生了各种支持自定义标签的技术，都可以称之为Web组件化技术，常见的有：

1. 现代浏览器支持的直接自定义新标签的技术（可以理解为是XML），在HTML代码中直接写新标签，默认是inline的，通过CSS可以给予更多的控制。老浏览器不支持。
2. 通过Angular的指令或组件自定义标签。成熟，功能强大，商业应用广泛。ionic中就通过Angular定义了大量的UI控件。
3. 通过Vue.js的组件定义标签。Vue.js中可以定义componet（组件），然后在模版中可以像标签一样使用（Vue.js见下文第三方库及框架列表）。
4. 通过React组件定义标签。这种其实算不上自定义标签，因为它不是HTML，而是JS（所谓的标签其实是jsx组件，编译后变成了js，与HTML关系不大）。
5. Web Components技术，由W3C提出的技术，有望成为未来的标准。允许直接从DOM的HTMLElement类继承出新标签。这种技术是多种Web新技术的综合，包含自定义标签类、template模板、shadow封装、import导入等，目前还处于试验阶段，除了谷歌外大多数公司都还没有提供支持，离商业应用还很远。

**异步编程Promise/Deferred、多线程WebWorker**

长期以来JS都是以单线程的模式运行的，而JS又通常应用在操作用户界面和网络请求这些任务上。操作用户界面时不能进行耗时较长的操作否则会导致界面卡死，而网络请求和动画等就是耗时较长的操作。所以在JS中经常要进行异步编程。而最基本的异步编程方法是事件和回调函数。但无论是事件还是回调函数在遇到稍微复杂一点的场景时都会变得难以使用。如时机问题、等待问题等。这时就产生了Promise的概念。

Promise可以保证无论什么时候添加回调函数，都能使回调函数得到恰当的调用；还能保证异步任务的状态不会被篡改。JS中的Promise有多种实现方案，它们的API各有不同，但核心概念都是相似的（jQuery3.x的Deferred已经向ES6靠拢）。

ES6支持Promise、提供了resolve、reject、then、catch、race、all等最基本的API。

jQuery则通过Deferred额外提供了进度通知及在外部改变状态的API，支持resolve、reject、then、done、fail、always、progress、notify、state等，还有通过$.when()支持类似all的功能，支持通过promise()转换成Promise对象（不是ES6中的Promise，而是表示一种不可从外部更改状态的Deferred）

Angular则支持$q，它即兼有ES6中Promise和jQuery Defferred的特点。

使用Promise可以将异步任务本身与后续业务完全分离，因此可以简化异步编程。通过 .then() 的链式调用还可以减少回调函数嵌套的层级，使代码更加易于阅读和易于理解。

JS在H5时增加了多线程API，即WebWorker。WebWorker是一个真正的分线程，与其它系统线程一样。但与其它编程技术中的多线程不同，它是通过消息机制与主线程交互的。因此可以理解为是放入沙盒中的线程。因为没有开放其它API，避免了产生线程死锁的可能，但功能上要弱一些。

**数据结构**

在程序中表示简单的数据很简单，但要表示一系列有特定关系的简单数据就不那么简单了。要存储和使用一系列有特定关系的数据就需要构建和使用数据结构。常见的数据结构有：

线性链表：用来表示一串数据的结构，如我们经常使用的数组、队列、栈都属于这种类型。数组可以通过索引随机访问，队列和栈都有严格的限制；队列是先进先出的，栈是后进先出的。

树：用来表示有子分支数据的结构（有分枝但无环），树在用户界面（页面上的标签元素就是以DOM树的形式存在的）和文件系统中有广泛的使用，树在组织机构等业务中也有广泛的使用。树由节点构成，节点通过parent连接上级节点就能构成树，有时为了能够更方便地使用树，还会提供children、level、path等API。B树（平衡多叉树）可用于二分查找（数据库索引常用方法）。遍历所有树节点有2种方法，即深度优先法和广度优先法。

表：用来存储有名称（或哈希值）的数据的结构。表有广泛的应用，如我们使用的Collection、Hashtable等就是表，JS中的对象也可以认为是表。

图：用来存储有环形分支数据的结构，在地图、路径规划类场景中有广泛应用，在图中求最优路径是最典型的需求。

在某一些应用框架中，为了简化开发避免复杂应用带来混乱，会强制采用单向数据流的方式解决应用内部数据传递和方法相互调用问题。典型的如React + Redux（或者Flux）。在这种应用框架中，表示数据的state是不可变的，如果要改变，只能生成一个全新的数据对象替换原对象。这时就需要不可变的数据结构。专门有一个叫immutable.js的脚本库（也是Facebook的人开发的）为JS提供了不可变的数据结构，其中提供了Collection、List、Map、Set、Record、Seq等数据结构。

**算法**

解决问题的操作步骤。如排序算法、查找算法、最短路径算法（在地图上规划路线）、图形识别算法（识别车牌号、花朵、人脸）、围棋算法（阿尔法狗）等。计算机算法要求必须在有限步内可解（否则相当于死循环）。算法通常追求使用时间最短和占用内存最少，但2者往往不能兼有。用时短的算法往往占用空间较多，而占用空间少的往往需要运算的时间较长。算法除了不断追求性能外，还逐渐向着智能化的方向发展（人工智能AI、深度学习）。

**加密**

数字摘要：从一个较大的数据（如一个文件或一个长字符串）中提取出一部分信息用来作为该数据的指纹（也叫哈希值）。通常用来防篡改或验证数据是否有损坏。数字摘要算法可以达到在几万字的文件中哪怕只修改一个标点就会产生截然不同的摘要的效果。常见的MD5、SHA都是数字摘要算法。数字摘要值通常是几十个字符的长度，通过数字摘要无法反向算出原数据，即在摘要时，信息会丢失。**但是，对于密码这样的短字符串，可以通过巨大的密码字典反向映射查出原字符串，因此短密码通过数字摘要的方式存储保密效果不佳！6个字符生成的数字摘要可以在1秒内查出原字符串，6位纯数字则可以缩短到毫秒内破解！**

对称加密：加密和解密都使用的是相同的密钥。通过密钥可以从密文还原出原文。对称加密最大的问题是如何保存密钥。战争时期，各种间谍努力窃取的就是密钥。DES、3DES、AES、RC5等都是对称加密算法。我们使用无线网络上网时输入的密码就需要以加密的方式发送，因为Wifi信号充满了空间，任何人都可以监听到。但用来加密的密钥本身也需要通过Wifi从无线路由传到我们的电脑上，这时就会造成密钥泄露，因此通过较长时间监听Wifi数据就能从数据中找到密钥，进而破解出Wifi密码。

非对称加密：加密和解密使用的是不同的密钥。使用非对称加密时，首先会计算出一对密钥，分别称为公钥和私钥。公钥可以分发给其他人，甚至完全公开；私钥则由自己保存。通过私钥加密的数据只能使用公钥解密，而使用公钥加密的数据只能使用私钥解密。非对称加密的这个特性非常有趣，也有很大的应用价值。RSA即是一种非对称加密。

数字签名：非对称加密技术的一种应用。可以用来保证数据发送者的身份。因为私钥只有创建密钥对的人才有，因此可以认为只要是使用某个私钥加密过的数据就一定是这个私钥的拥有者发出的数据。因为非对称加密比较慢，不适合加密大量数据，所以通常是先使用数字摘要计算出一个很短的摘要值，再对摘要值进行私钥加密。

数字证书：非对称加密技术的另一种应用。虽然公钥是可以公开的，但怎样确保我们得到的公钥是某人或某个机构的公钥而不是别人冒充的公钥呢？这时就需要有一个可以信任的机构为公钥提供证明。这种包含了公钥拥有者名称和可靠证明的公钥（通常是证明机构的数字签名）就是证书。所以在实际的通信和数据交换时，都不会直接使用裸公钥，而是要求对方提供证书。如HTTPS请求就要求网站的服务器提供证书证明公钥是可靠的（以防冒充钓鱼），我们在手机上安装的App也需要证书才能提交到App Store。值得一提的是，证书是有有效期的，证书过期将使HTTPS网站受到影响，向CA证书颁发机申请证书是需要交费的（按年计费）。

**Angular中$rootScope、$scope、$watch()、$digest()及$apply()**

$rootScope是Angular中的一项内置服务，通过它可以获得根作用域。根作用域是Angular应用启动时创建的第一个作用域（$id等于1）其它作用域都是它的子孙作用域。作用域中有$parent属性指向父作用域从而将作用域连成一颗树。大部分作用域还会将原型（prototype）指向父作用域从而形成作用域链，以继承上级作用域的属性和方法。（但自定义指令和组件中的**隔离作用域**只有$parent指向父作用域而不会将自己的原型指向父作用域，因些隔离作用域不会受上级作用域干扰，这样自定义指令和组件就可以放在页面上任何地方使用了。隔离作用域的原型指向作用域类的prototype）

在控制器构造函数中通过$scope可以获取当前作用域。某些指令或组件中还存在隔离作用域，隔离作用域也在作用域树中，只是没有与上级作用域通过原型链连接起来，从而不受上级作用域的影响。使用$new()方法可以创建新的作用域，需指定是否是隔离的，及指定上级作用域。

作用域有自己单独的事件传播机制，通过作用域$emit()的事件会向作用域树的上级传播（含自己），通过作用域$broadcast()的事件会向作用域树的下级传播（也包含自己）。

作用域最重要的是提供了数据的修改监视和通知功能，通过$watch()可以监视作用域中数据的变化（当发现变化时可以更新界面或作用域中的其它数据）。$watch()通常在模版编译时被Angular调用为模版插值添加监视器，也可以由程序员调用。$watch()可以监视复杂的表达式。

$watch()的监视功能是由$digest()推动执行的，$digest()会遍历整个作用域树循环处理每一个watcher、计算watcher中被监视表达式的值，如果表达式的值与原值（缓存的）相比发生了变化则调用watcher中的函数处理变化，因为watcher变化处理函数有可能修改作用域中的其它值，因此$digest会遍历多次（至少2次）直到作用域中的数据稳定（检测不到任何变化）为止。

Angular控制器及自带的指令（如ngClick等）、服务（如$timeout等）都会自动调用$digest()以触发监视过程，如果在非Angular执行环境中修改作用域中属性的值则不会自动激活$digest()过程，这时需要使用$apply()而不要自己调用$digest()，$apply(func)会执行func，并且在执行之后会调用$digest()，而且会恰当处理func执行过程中的错误。

具体可参考：http://dwz.cn/3Npuio

**使用batarang浏览器插件可以观察Angular的性能，每次$digest()的时候都会在图形上显示一条竖线。**

影响Angular性能的一个重要因素就是数据绑定及很多Angular指令都会向作用域中添加watcher，一个页面上可能有成百上千个watcher。每次$digest()时都需要执行所有的watcher，这样watcher越多，性能就越受影响。假设页面上显示一个表格显示商品信息，表格有10列，分别显示商品名称、进价、销售价、库存等10个数据（通过{{}}绑定）。则显示1000条商品信息至少会产生10000个watcher，如果1个watcher的执行需要1ms，则页面上的任何操作触发$digest()都需要至少10秒才能反应过来，页面就像卡死了一样。所以减少watcher是关键。如有一个第三方提供了**bindonce**指令(bo-开头)，使用bindonce只会在数据第一次显示时产生watcher，数据显示之后watcher就移除了，所以可以大大提升性能。

除了使用bindonce之外，还可以考虑优化ngRepeat，比如使用ionic的列表（ionList/ionItem）时，如果加载的数据很多，如使用无限加载，应该使用collection-repeat替换ng-repeat。因为collection-repeat会把滚动过去的列表项对应的标签元素（DOM）及watcher从页面上删除，这样无论列表多长都不会卡了。

如果应用比较复杂，控制器很多，还可以使用异步方式加载控制器，而不是应用一打开就加载所有的控制器（angular的ng-controller及路由默认都是立刻加载控制器的）。这方面的解决方案有：angular-async-loader模块。

**各种代码库及框架**

|  |  |  |
| --- | --- | --- |
| Bootstrap | UI框架 | 响应式页面、基本排版、表单、常用UI组件、插件 |
| jQuery | 脚本库 | 增强了DOM/Ajax/动画等操作，提供了Callbacks、Deferred等异步编程工具 |
| Underscore.js | 脚本库 | 增强了JS中的数组/对象/函数等，还包含一个HTML模板引擎，使用它可以简化数组数据操作，因此Backbone框架的集合大大依赖于Underscore。Undercorce还提供了函数调用节流（可以降低函数调用频率，如窗口resize事件处理函数、鼠标mousemove事件处理函数的调用频率都是很高的）等实用功能。 |
| jQuery UI | 桌面UI库 | jQuery官方推出的，用于桌面网页开发的UI库，有丰富的控件、主题较多且可以自由定制，它可以向页面上添加丰富的功能，而不会对原页面产生影响 |
| jQuery Mobile | 移动UI框架 | jQuery官方推出的，用于移动Web应用开发的UI，控件丰富，动画效果很好。但UI设计风格显老。 |
| SUI Mobile | 移动UI框架 | 淘宝基于framework7推出，功能丰富，使用简单，多页开发模式（但可以模仿单页加载）。 |
| Frozen UI | 移动UI框架 | Frozen UI是一个开源的、简单易用的（与SUI Mobile类似），轻量快捷的移动端UI框架。基于手Q样式规范，选取最常用的组件构成的UI框架，目前多应用在腾讯手Q增值业务中。 |
| We UI | 移动UI框架 | 腾讯推出的，用于微信页面开发的UI框架，界面风格与微信相似度最高，功能和组件偏少，适合于简单的微信风格的移动页面开发 |
| APP Framework | 移动UI框架 | 原来叫JqMobi后被Intel收购，简单、支持多种系统主题，有很好的动画效果（侧滑、转场），适合开发小产品，资料较少，应用不多。 |
| ionic | 移动UI框架  应用框架 | ionic是一个非常漂亮的移动UI框架、并且它集成了Cordova（参看后面的说明）和Angular，可以快速开发出与原生应用相媲美的基于Web技术的App |
| WeX5 | 移动UI框架  应用框架 | 国内（起步科技）开发的类似于ionic的应用开发框架，在底层同样采用了Cordova。还基于Eclipse提供了快速开发工具（支持拖拽式所见即所得开发）。WeX5虽然是国内公司开发的，但即使在国内应用的也不多。与ionic不同，WeX5的应用框架采用的是Knockout.js（下面有介绍），也许这是它未能广泛应用的原因之一。 |
| MUI | 移动UI框架 | 国内（DCloud，做Hbuilder的那家公司）开发的类似于SUI的UI框架。这家公司（参加了国内的HTML5产业联盟，几家公司拼凑出来的，并没有什么影响力）还提供了一个所谓HTML5+（几乎就是抄Cordova），通过它可以让页面上的js调用手机API（需要打包成APP或运行在它开发的Webview浏览器中），不过也没有在国内获得广泛的应用（或许是缺少强大的应用框架的原因）。 |
| React | UI组件化库 | React提供了组件化UI的支持，可以很方便地开发出UI组件，并且支持嵌套（比Angular简单）。为了简化组件的使用，React还扩展了js，提供了jsx。React不算是UI框架或应用框架，因为它即没有提供现成的UI控件可以使用，也没有提供管理数据的模型和控制器机制。要想使用React开发一个项目，需要用到大量相关技术支持，如Redux、Reselect、React-Router、Webpack、Babel等，React是一个集成了很多技术的体系，单一使用React是很难做出复杂项目的。因为React过于激进，开发体系还不是很成熟，目前使用React的公司还比较少。React的远期目标是打造一个学习一次就能通吃Web和iOS、Android的应用开发技术，但要走的路还很长。 |
| Vue.js | UI组件化库 | 与React类似，只提供UI层的组件化机制，但它参考了Angular，支持数据绑定。Vue.js是尤雨溪（中国在美留学生）发布的一个开源项目，这兼有React和Angular的特点（甚至还有Backbone的影子），但从本质上来说，更接近于React。因为Angular是一个完整的应用框架，而Vue.js只是View层的。  Vue.js与React不同的是React是非常激进的，彻底革命性的，React的目标更宏大，它为原生App开发提供了一个新的技术，Vue.js则只使用了普通的js技术（使用模版而不是JSX，虽然它也支持JSX）。 |
| Require.js | JS模块化 | JS模块定义、使用代码异步加载JS、依赖解决、其它各种文件的代码加载。Require.js有很多插件，用来实现各种各样的异步加载功能 |
| Sea.js | JS模块化 | 与Require.js大同小异，来自阿里（作者王保平、阿里昵称玉伯，现供职于支付宝前端技术部）、主要在国内应用 |
| Backbone | 应用框架  类似MVC架构 | 非常小（压缩后7K多），但功能完善的前端应用开发框架，支持模型（集合）、视图、路由等，支持RESTful风格的服务端交互，是一种思路比较经典的MVC框架 |
| Angular | 应用框架  MVVM架构  MVC架构  MVW（Angular自创词） | 非常好地体现了快速应用开发（RAD）思想的框架，通过对HTML的扩展，借助声明式语言的优势实现快速应用开发。内置大量指令，支持双向绑定，支持表单输入状态管理，表单验证，表单辅助提交，支持路由、动画、RESTful数据模型。广泛使用了依赖注入实现松耦合。Angular有大量插件，逐步形成了一个生态系统 |
| Knockout.js | 应用框架  MVVM架构 | 面向对象风格的，支持数据绑定、支持自定义组件（类似于Angular的指令）的应用框架。国内应用很少（可能做服务端的比较喜欢）。 |
| Ext.js/Sencha | UI框架  应用框架  MVC架构 | 整合了树、列表、面板、布局控制、图表等全部的Windows应用控件。在所有JS的UI框架中，仿Windows风格质量最高，功能最强大的。  移动互联网时代到来之后，Ext也推出了移动开发框架，并且在应用架构上强化了MVC，还逐渐推出了自己的应用构建工具，并改名为Sencha。  通常用来开发各种内部使用的管理系统/业务系统。是很多类似UI框架的鼻祖。商业使用是收费的，因此应用较少。 |
| jquery EasyUI | UI库 | 仿Extjs的一个UI库，提供了和Extjs高度相似（界面和API，甚至文档界面都高度相似，几乎可以算是抄袭了）的应用控件，与jquery结合更好，使用上也比ext后来的版本要简单，相当于ext3.x/4.x的程度。在ext收费和越来越复杂的情况下，EasyUI真正是给大家提供了一个Easy的UI框架，因此国内有一定的应用（常用来做后台管理系统之类的，做服务端的人比较喜欢）。值得大家课余抽时间研究一下。 |
| Prototype.js | 脚本库 | DOM/BOM/Ajax/数组/对象/函数工具，一整套面向对象的实现机制。历史很早的一个脚本库，侵入式的，jQuery产生后逐渐淡出。 |
| Highchart.js | 统计图库 | 基于SVG技术，支持折线图、曲线图、柱形图、条形图、饼图、环形图、雷达图、极地图、蛛网图、仪表图等，有丰富的动画和交互效果，提供了简单易用、控制能力完善的API。产生时间较早，很成熟。商业使用收费。 |
| eChart | 统计图库 | 百度推出，与Highchart.js相似。免费，开源、功能很丰富，对移动端支持也很好。产生稍晚，百度自家产品中大量使用，借助百度影响力应用会越来越多。 |
| Chart.js | 统计图库 | 基于Canvas技术，支持折线、曲线、柱、饼等，因为使用Canvas技术，交互性较弱。 |
| artTemplate | HTML模板化 | 性能很高，语法简洁，支持编译，支持浏览器端使用，支持express。来自腾讯 |
| EJS | HTML模板化 | 默认使用<%%>的HTML模板语言，支持浏览器端使用，支持express |
| Jade | HTML模板化 | 非常独特的HTML模板语言，完全取消了标签中的<>和标签配对，过于独特限制了它的应用，支持express |

**设计原则/设计模式**

面向对象编程（OOP）中最重要的是面向对象设计（OOD）。单独设计一个类并不难，难的是设计很多类，并使用这些类及它们的实例构成复杂的应用程序来满足业务需求！当使用面向对象的思想对需求进行分析（OOA，面向对象分析）逐步构成设计成果时，如果确定类的范围、如何解决类与类、包括它们的实例之间的各种关系必须需要一些设计原则作为指导才能较好地完成设计。

基本的面向对象设计原则有：

1. 单一职责原则（封装的范围和粒度）

一个类、方法应该只负责一项功能

1. 替换原则（正确使用继承和多态）

将父类看成接口，子类完全可以替换父类，

1. 依赖倒置原则（解除依赖）

面向抽象<接口/依赖注入>编程，而不是面向实现编程

1. 接口隔离原则（抽象的粒度）

接口的范围和粒度

1. 迪米特法则（尽可能减少耦合）

一个对象应该尽可能少地了解其它对象

1. 开闭原则（保持可扩展性）

对扩展开放，对修改关闭

具体可参考http://dwz.cn/3NfHS9

设计模式是一些常用的设计结构，是从大量的面向对象设计中识别并总结出来的。设计模式可以看成是特定设计场景下的经典设计方案。是一些被命名、分类、说明了使用场景的固定设计公式。由GOF<4人邦，即提出设计模式的4个人>最先总结出23种模式并出书论述而流行起来。学习设计模式可以丰富我们的设计思路，提升我们对设计的理解，可以使我们在设计上快速成长和成熟起来，也能在一定程度上提升设计的质量和速度。但不同的编程语言有不同的特点，也有不同的框架支持，因此对设计模式不可僵化理解和应用，而应该根据情况灵活变通取舍；也不可过度应用，为了应用设计模式而设计。

GOF提出的设计模式可分为3大类：

创建型模式：用于创建实例，如单例模式、工厂模式等

结构型模式：用于解决实例间的关系，如适配器模式、外观模式、代理模式等

行为型模式：用于实现某些功能：迭代器模式、观察者模式、模版方法模式等

**JS严格模式**

ES5引入的一种新的JS运行模式，在严格模式下可以改变JS历史上遗留的一些不合理的语法或运行行为（这些语法或行为不能在普通模式下直接改变，因为会造成之前的老代码出问题），JS严格模式是为了在JS中引入新特性又不致于破坏老代码而产生的解决方案。

使用字符串字面量 ‘use strict’使脚本、<script>标签或函数进入严格模式

在严格模式下：

1. 全局变量必须先声明才能使用（包括赋值）
2. 禁止使用with关键字
3. eval执行在eval作用域中（这个作用域是新增的），这样eval就不会影响全局作用域或函数作用域了（同时也无法修改全局作用域中的变量）
4. this不会也不能指向全局作用域（window对象），这样构造函数不加new调用就会报错，从而不至于影响全局作用域
5. 禁止访问arguments.caller（表示调用当前函数的函数）、arguments.callee（表示当前函数自已，通常在js早期版本用于匿名函数递归调用），也不能通过函数名访问arguments，但对arguments的其它使用（如获取实参）是正常的！
6. arguments不可被赋值，也不再追踪参数值的变化（arguments中的值不会变）
7. 禁止使用delete删除变量（但删除对象的属性是正常的）
8. 显式报错（程序会停止执行），而不是沉默忽略错误（如对只读属性赋值、对禁止扩展的对象添加属性、删除不能删除的属性、属性名重复、参数名重复）
9. 不支持八进制，整数第1位是0将报错
10. 函数不能在if{}、for{}等{}内部声明（以后版本会引入块级作用域）
11. 新增implements, interface, let, package, private, protected, public, static, yield关键字，不能使用这些名称作为变量名，函数名、参数名等

具体可参考http://dwz.cn/A6tGP

**正则表达式**

使用特定字符构成的表达式，可以用来描述字符串模式，用来在字符串中搜索（匹配）符合条件的子字符串（当然也以替换）。使用正则表达式可以方便地构造出各种各样的字符串搜索条件，而且还可以将正则表达式编译成可执行代码，从而可以以极高的性能处理输大的字符串和文本文件。正则表达式常用来检验字符串格式（数据验证）和对大量文本进行数据分析，加工（数据处理）。

**Git/SVN**

源代码管理（SCM）工具，Git是分布式，任何人都可以拥有完整的代码库，几乎所有的源代码管理操作都可以不用连网，而在自己的电脑上完成，所以是开源项目的不二之选，在公司中使用也有很多优势。SVN是集中式的，代码库及历史版本都集中管理在中央服务器上，开发人员只有当前使用的版本，SVN简单、能够实现代码的严格控制，因此在要求较强控制和管理的场景下应用较多。

不管将来入职的公司采用的是Git还是SVN，入职时或入职后都会分配一个帐号，通过这个帐号可以获得公司的源代码并加入到开发团队中。如果是Git，一般会给一个网址，打开之后与“码云”或github类似，找到项目克隆到自己的电脑上就行了（提示权限时输入用户名和密码）。如果是SVN，则需要使用相应的SVN工具（如TortoiseSVN、苹果系统中还可用Versions等；有些开发工具自身就支持SVN，如WebStorm，需要配置一下），工具装好后，还需要项目网址（通常与帐号一块提供），然后将项目checkout（迁出）到自己的电脑上就可以了，SVN使用比Git简单得多，从代码库获取代码就叫checkout（迁出），代码修改后更新到代码库叫checkin（迁入），其它的事都是领导或管理员操作。

**PhoneGap/Cordova**

H5具有跨平台的能力，而且包含了与App相关的大量新特性，随着移动浏览器不断增强，使用H5开发（write once）然后使用原生外壳打包成各个平台的App（run anywhere）就成为一种可能。PhoneGap就这样产生了，PhoneGap的目标是弥补手机与Web之间的间隙。

PhoneGap后被Adobe收购，但PhoneGap并没有获得广泛的应用（那时H5还没发布），Adobe想通过PhoneGap赚钱就必须吸引更多的人加入到PhoneGap的阵营，而又必须保留营利的机会。于是Adobe将PhoneGap捐给Apache Software Foundation成为一个开源项目（吸引更多的人开发它<降低成本>和使用它<扩大影响力>），而又保留了PhoneGap的商标。

ASF接收PhoneGap的源代码后无法使用PhoneGap的名字，只能改名，于是产生了Cordova。Adobe则在Cordova的基础上继续使用PhoneGap的名字发展相关的开发工具，如桌面版工具（PhoneGap Desktop）、移动版运行和测试用的App（PhoneGap App）、和能够营利的构建云服务（PhoneGap Build）。最终的结果是，Adobe把吃力的事（写底层代码Cordova）抛给了开源组织，自己则把精力放在赚钱上（研发基于Cordova的开发工具PhoneGap三件套）。

现在核心代码在Cordova中，PhoneGap则是Adobe研发的一系列Cordova开发工具和云服务。

Cordova提供了很多插件，可以使js获得访问原生API的能力，如控制系统状态栏、获得电池信息、访问手机存储空间等；Cordova还提供了App构建工具，将Web App打包成各种平台上的原生App，以发布到应用市场中。值得一提的是Cordova中不存在Ajax跨域限制，因此可以像原生应用一样访问任何域名下的接口。

**各种Web服务器及服务端技术**

Java系：以Java EE（最早叫J2EE）中的Servlet（Web核心）、JSP（呈现HTML）、EJB（增强并规范Java语言面向对象特性）等为核心Web服务端开发技术，随后产生了很多相关技术和框架如：Structs（MVC）、Spring（解除耦合）、Hibernate（对象和关系型数据库的映射）等（Structs、Spring、Hibernate也常简称为SSH）。

PHP系：以PHP为核心的Web服务端开发技术，因为PHP语言及开发方式比较杂乱，因此产生了很多框架，如：Laravel、Symfony2、Nette、CodeIgniter、Yii、PHPixie、Zend Framework等，还有国产框架Think PHP。

.NET系：微软公司的Web服务端开发技术。使用C#、VB.NET等编程语言和.NET Framework，分为两大系列，早期的ASP.NET Web Form（组件化、事件驱动、与Windows应用程序的开发方式相似）、 和后来的ASP.NET MVC。另外ASP是微软在.NET之前的Web开发技术，与JSP一样古老。

其它：Node.js、Python、Ruby等

**高并发网页**

1. 减少HTTP请求次数，如将小图片整合成大图片(CSS雪碧图技术)，将js、css打包成一个文件（Grunt/Gulp/Webpack等）
2. 减少文件字节数（选择适当的编码方式/代码压缩<去除空白，缩短变量名等，使用Grunt/Gulp>）
3. 文件压缩<二进制压缩>后传输（现代服务器如IIS等都支持的功能, 属于服务端技术）
4. 使用浏览器端及网络缓存（可减少请求次数）技术
5. 将CSS/JS/图片等资源与HTML分离到不同的服务器上（分散请求）
6. 使用CDN（静态网页及各种资源都适用，分散请求，还有利于浏览器缓存）
7. 服务端缓存（Memcache等Redis各种内存数据库/分布式内存数据库，提升处理速度，属于服务端技术）
8. 集群（负载均衡网络硬件解决方案/反向代理软件解决方案，增强服务端处理能力，属于服务端技术）
9. 不规律高并发（云计算弹性资源池，按需分配资源以解决平时资源利用不充分的问题，属于服务端技术）
10. 大规模数据存储和计算（需用大数据技术、属于服务端技术）

大规模实时数据处理（流式大数据技术、属于服务端技术）

**TypeScript是什么？**

TypeScript是JavaScript的超集，它让JavaScript支持**强类型**。在强类型的编程语言中，变量的类型从声明时就确定了，并且不能再发生变化，如果赋值为其它非兼容类型则直接导致编译出错（例如age变量类型声明为Number则赋值为String时就会报错，而这种情况在js中是不会报错的）。强类型可以使开发工具（IDE）更容易通过代码提供的类型信息发现代码中的错误，也能在编程时给程序员更好代码提示，还有利使用工具对代码进行准确地重构。强类型语言的开发工具通常都做得特别好，智能化程度较高，大型项目的开发效率也更高。TypeScript包含了ES6和ES7的语法，它是JavaScript“未来版”。它不能直接运行，需要先编译成普通JavaScript才能在浏览器中运行。它是开源的。它来自于微软。

**如何使用TypeScript？**

TypeScript依赖于NodeJS，运行前需要安装和编译

安装：**npm install -g typescript**

编译：**tsc helloworld.ts 【**把TypeScript文件（\*.ts)编译成普通的JavaScript】

**使用TypeScript的好处?**

1. 强类型，有更高的可预测性，更易纠错（即开发工具更加智能）。
2. 编译为JavaScript的过程中，可以检查出很多普通JS难以发现的错误。
3. 增加了模块，命名空间和强大的面向对象编程支持，构建大型复杂应用程序更加容易。
4. Angular 2 框架就是用 TypeScript 编写的，并且官方推荐开发人员也使用它。
5. React中也可以使用TypeScript编写代码

**TypeScript几个重要特性？**

**1. 强类型**

var name: string = 'zhangsha', // String

age: number = 18, // Numeric

sex: boolean = true; // Boolean

// 直接赋值时也可以简写为var name = 'zhangsha'; 因为此时编译器可自动推断为String类型

**TypeScript的基本数据类型**

number（数值类型）：所有数字都是数值类型的，无论是整数、浮点型

string（字符串类型）：可以使用单引号或双引号

boolean（布尔类型）：true 或者 false，用 0 和 1 会造成编译错误。

any（任意类型）：该类型的变量可以设定为任何类型，但应尽可能少地使用它，优先考虑使用范型。

array（数组类型）：有两种语法：my\_arr: number[];或者my\_arr: Array

void （空类型）：用在不返回任何值的函数中。

所有变量类型列表请参考http://www.typescriptlang.org/docs/handbook/basic-types.html

**2. Interfaces 接口**

注意此接口是编程语言级的，不是开发应用时常说调用服务端接口的那个接口。

编程语言级的接口通常是一种面向对象程序设计上的考虑，为的是强制某些类实现一组方法或属性等成员。TypeScript不支持多重继承，但一个类可以实现多个接口。

实现接口的对象拥有的**属性顺序并不重要**。只需要满足相应接口的**属性个数**和**类型**。属性的**个数不够**，**类型不一致**，**名称拼写出错**等情况发生时编译器都会报警告信息。但对象的属性个数多于接口的属性个数时，编译器不会出现警告信息。

1. **类**

在搭建大型规模的应用程序时，许多开发者会优先选择面向对象编程语言(Java、C#)。TypeScript 提供一个**类类型**，可以用类实现**继承**，**抽象**，**封装**，和 Java、C# 中的非常相似。TypeScript 给开发者另外一种选择，也可以用来构建大型应用程序。

TypeScript 创建类时使用class关键字

**4. 泛型**

泛型（Generics）是允许同一个**函数**或**类**接受**不同类型参数**的一种**模板**。相比于使用 any 类型，使用泛型来创建可复用的组件要更好，因为泛型会保留参数类型。

function identity<T>(arg: T): T {

return arg;

}

// 会输入字符串类型的参数

var output = identity<string>('myString');

// 会输入数字型的参数

var output2 = identity<number>(100);

**5. 模块**

**模块在其自身的作用域里执行，而不是在全局作用域里。**这意味着定义在一个模块里的变量，函数，类等等在模块外部是不可见的，除非你明确地使用**export**形式之一导出它们。 相反，如果想使用其它模块导出的变量，函数，类，接口等的时候，你必须要导入它们，可以使用 import形式之一。

两个模块之间的关系是通过在文件级别上使用imports和exports建立的。

模块使用**模块加载器**去导入其它的模块。在运行时，模块加载器的作用是在执行此模块代码前去查找并执行这个模块的所有依赖。大家最熟知的JavaScript模块加载器是服务于Node.js的 CommonJS和服务于Web应用的Require.js。

可以通过import 和 require 关键字在另一个模块中导入其他模块。

**6. 装饰器**

装饰器（Decorators）为我们在**类的声明**及**成员**上通过**元编程语法**添加**标注**提供了一种方式。可以在**切面上给类增加一些特性**。装饰器目前处在实验性阶段。

装饰器是一种**特殊类型的声明**，它能够被附加到**类声明**，**方法**， **访问符**，**属性**或**参数**上。 装饰器使用 **@expression**这种形式，expression求值后**必须为一个函数**，它会在运行时被调用，被装饰的声明信息做为参数传入。

创建装饰器：

function sealed(constructor: Function) {

Object.seal(constructor);

Object.seal(constructor.prototype);

}

使用装饰器：

@sealed

class Greeter {

}

**Angular2**

Angular2是对Angular1一次颠覆，可以开发**跨平台**的**渐近式**Web应用，引入H5的新特性Web Worker和服务端渲染来**改善了性能**，它**配合TypeScript语法**来编写严格的代码，号称**一套框架多种平台**，同时适用手机与桌面，正在飞速发展中，下一个版本Angular4（已于2017年1月6日推出beta版）更让人期待。

Angular2 更具有服务端开发的思想，是一个**前端MVC或MVVM框架**， 去除了Angular1中的控制器概念，引入了**组件开发思想**，**专注视图层业务逻辑开发**。来自Google的Angular2和来自微软的TypeScript结合，发展迅速。Angular2支持**双向绑定数据**，**脏值检查**，**模块化开发**，新的TypeScript语法，**模板**、**样式**和**业务逻辑完全分离**，以及支持生命周期事件等将是前端开发的一个方向。

Angular2 应用是由**组件**组成的。 **组件**由 **HTML 模板**和**组件类**组成，**组件类控制视图**。组件负责控制屏幕上的一小块区域，我们称之为**视图**。

Angular2 使用扩展语法编写**HTML模板**，用组件类管理这些模板，用服务添加应用逻辑， 用模块打包**发布组件**与**服务**。然后通过**引导根模块**来启动该应用。Angular在浏览器中接管、展现应用的内容，并根据我们提供的操作指令响应用户的交互。

Angular2 应用是**模块化**的，并且 Angular2 有自己的模块系统，它被称为 **Angular 模块**或 **NgModules**。

Angular2 开发的每个应用至少有一个模块（**根模块**）。根模块在一些小型应用中可能是唯一的模块，大多数应用会有很多特性模块，每个模块都是一个内聚的代码块专注于某个应用领域、工作流或紧密相关的功能。Angular2 模块（无论是根模块还是特性模块）都是一个带有@NgModule装饰器的类。

Angular2 中有很多装饰器，它们负责把**元数据**附加到类上，以了解那些类的设计意图以及它们应如何工作。

Angular2 中要显示组件的属性，最简单的方式就是通过**插值表达式**来绑定属性名。 要使用插值表达式，就把属性名包裹在**双花括号**里放进视图模板

Angular2 中使用**事件绑定**机制来响应任何 DOM 事件。 许多 DOM 事件是由用户输入触发的。绑定这些事件可以获取用户输入。

Angular2 中使用**ngModel**可以进行**双向数据绑定**

Angular2把后端开发中用来解决复杂问题、实现高弹性设计的**依赖注入**(DI)技术引入了前端。

Angular2使用了ES6的注解器，让前端也拥有切片编程的威力。

Angular2还可以与Ionic2一起配合使用。

**Vue:**

Vue是是一款MVVM模式的前端js框架，其最大的特点是基于.vue单文件的组件化开发，使一个复杂的页面可以拆分为多个独立的，可复用的组件。另外还拥有双向数据绑定，事件系统，计算属性(computed)，状态过度，自定义指令，过滤器等功能。

组件：组件（Component）是 Vue.js 最强大的功能。组件可以扩展 HTML 元素，封装可重用的代码。在较高层面上，组件是自定义元素， Vue.js 的编译器为它添加特殊功能。在有些情况下，组件也可以是原生 HTML 元素的形式，以 is 特性扩展。组件中几个重要的属性：

data:组件自身的数据，除根组件之外，必须是一个函数。

template:组件模板,可以是字符串，也可以是模板所在标签的#id，如果使用.vue文件设计组件则可以没有temlpate属性；

props:从父组件传值接受的数据，只能读取，不应修改。

computed:使用data或props计算出来的属性，其计算的结果会被缓存，只有函数中使用的变量变化时才会再调用。

methods:组件的方法(函数)

mounted:组件被渲染之后的钩子函数（被添加到俯视图上时）。

created:被创建时的钩子函数。

activated:组件变为活跃状态时的钩子函数。

mixin:混入选项。

条件渲染,v-if ,v-show,可以实现切换某个组件或元素的显示与隐藏。

v-for,用于渲染一个列表，常用于数组的渲染。

单向数据流：vue中数据的传递只能是自上而下的，父组件传递给子组件，子组件通过事件发射$emit来实现与父组件的通信，非父子组件之间的消息传递可以通过bus总线，更为复杂的情况可以通过vuex。

Slot插槽：父组件给子组件传值还可以通过标签内容传递，而slot插槽则是标记html内容分别应该放在子组件模板中的什么位置。

Vue双向数据绑定原理：vue对象在实例化时，会将data对象中的所有属性都通过Object.defineProperty方法以set,get方法的形式添加给vue实例，然后在属性的set方法中添加了监视函数，每当修改data属性时，set方法就会调用，从而调用监视函数，监视函数会修改页面中绑定这个数据的部分。

单文件组件：使用vue-cli创建项目时，可以使用.vue文件创建单文件组件，可以将一个组件写在一个vue文件中，然后在其他文件中导入就可以使用。这样大大提高的代码的结构性。

Webpack,一个功能强大的打包工具，vue使用webpack进行打包，通过vue-loader将vue文件打包成浏览器可以识别的js文件。

Vux,一个功能齐全的vue组件库，其中实现了轮播图，列表，选项卡，tabBar等移动页面常用的功能。类似的组件库还有很多，例如mintUI,vonic。