抽时间用 markdown 写在 GitHub 上。

GitHub 托管地址：https://github.com/king-hcj/Love-study

爱用前端实习生资料索引（必看）：

https://aiyongbao.yuque.com/allhands/mb50kc/np1w61

爱·学习笔记

目录

一、Sublime、Atom 还是 Vscode?

有一个传言:世界上有三种程序猿,一种是用 Emacs 的,一种是用 Vim 的,一种是用其他编辑器的。Emacs，著名的集成开发环境和文本编辑器。Emacs 被公认为是最受**专业程序员喜爱**的代码编辑器之一，另外一个 vim。

因为 **Vim 和 Emacs 学习曲线比较陡**,作为第三种程序员,还是来看看当前一些简单好用的主流编辑器吧!

1.Sublime

功能概览:

Sublime Text(中文官网) 是一个代码编辑器(Sublime Text 2(最新版 sublime text 3)是收费软件,但可以无限期试用),也是 HTML 和散文先进的文本编辑器。Sublime Text 是由程序员 Jon Skinner 于 2008 年 1 月份所开发出来,它最初被设计为一个具有丰富扩展功能的 Vim.

Sublime Text 具有漂亮的用户界面和强大的功能,例如代码缩略图,Python 的插件,代码段等.还可自定义键绑定,菜单和工具栏.Sublime Text 的主要功能包括：拼写检查,书签,完整的 Python API , Goto 功能,即时项目切换,多选择,多窗口等等.Sublime Text 是一个跨平台的编辑器,同时支持 Windows、Linux、Mac OS X 等操作系统.

插件安装（知乎插件推荐）

（1）安装 Package Control

按 Ctrl+` 调出 console,粘贴安装代码到底部命令行并回车。

重启 Sublime Text。

如果在 Perferences->package settings 中看到 package control 这一项,则安装成功。

（2）用 Package Control 安装其他插件

按下 Ctrl+Shift+P 调出命令面板

输入 install 调出 Install Package 选项并回车,然后在列表中选中要安装的插件。

主要优点

（1）主流前端开发编辑器

（2）体积较小,运行速度快

（3）文本功能强大

（4）支持编译功能且可在控制台看到输出

（5）内嵌 python 解释器支持插件开发以达到可扩展目的

（6）Package Control：ST 支持的大量插件可通过其进行管理

主要槽点

（1）收费闭源,收费的问题虽然不影响使用,但是毕源带来的问题就是一个 bug 千年都不见修复.很影响使用。

（2）虽然有 Package Control 管理插件,但是安装搜索插件时,只能看到简单的一句话描述,而且之后的插件配置不方便。

（3）当插件多了后,sublime 会时不时崩溃,用户也不能清晰的知道是哪个插件引起的.只能一个个去卸载、判断。

2.Vscode

功能概览:

Vscode 同样是一个跨平台的文本编辑器,内置了对 JavaScript, TypeScript and Node.js,C++, C#, Python, PHP 等语言的强大支持。据 vscode 的作者介绍，这款产品可能是微软第一款支持 Linux 的产品。

插件安装

Vscode 更新到目前位置,内置了插件安装功能,在编辑器右侧有一个插件中心的按钮,可以方便的安装,升级,卸载插件,此外可以在 vscode 的插件商店查找适合自己的插件 vscode 插件,相比 sublime 来说,vscode 的插件管理还是很方便的,而且插件数量也相当可观,其**生态圈**日趋成熟,毕竟背后是微软帝国。

主要优点

（1）好看,分分钟逼死 notepad++

（2）加载大文件几乎秒开,试过打开 100M 的工程,无压力

（3）C#支持高亮,已经编译过的还支持引用

（4）JS、HTML 等支持高亮和补全

（5）全平台

（6）免费

（7）占用内存低

（8）基于 atom,但是性能秒 atom

主要槽点

（1）还不够稳定,经常会崩溃

（2）插件还不是很完善,但是发展速度相当快

（3）debug 的灵活性还不够

（4）对 C#支持还不够好

（5）不支持工程加载,只支持文件夹加载,引用比较复杂的大工程支持力度低

3.Atom

功能概览:

Atom 代码编辑器支持 Windows、Mac、Linux 三大桌面平台,完全免费,并且已经在 GitHub 上开放了全部的源代码.在经过一段长时间的迭代开发和不断改进后,Atom 正式版在性能和稳定性方面都有着显著的改善。

开发团队将 Atom 称为一个“为 21 世纪创造的可配置的编辑器”,它拥有非常精致细腻的界面,并且可配置项丰富,加上它提供了与 SublimeText 上类似的 Package Control (包管理)功能,更重要的是 atom 的包管理工具可视化了插件的配置,以及插件的使用帮助以及对应的 github 的地址,这很大层度上方便了开发人员将 Atom 打造成真正适合自己的开发工具。

作为一个现代的代码编辑器，Atom 有着各种流行编辑器都有的特性，功能上非常丰富，支持各种编程语言的代码高亮(HTML / CSS / Javascript / PHP / Python / C / C++ / Objective C / Java / JSON / Perl / CoffeeScript / Go / Sass / YAML / Markdown 等等)，与大多数其他编辑器相比，Atom 的语言支持已经算是覆盖非常全面了.另外，它的代码补全功能（也叫 Snippets） 也非常好用，你只需输入几个字符即可展开成各种常用代码，可以极大提高编程效率。

插件安装

Atom 具有交互性很好的 GUI 插件管理中心，在这里用户可以很方便的搜索，安装，升级，卸载，配置插件。

主要优点

（1）开发维护团队强大，且是开源项目，因此修复 bug 速度快，生态圈成长速度快。

（2）快捷键支持特别好，熟悉了各种快捷键后可以成吨提高生成效率.

（3）比较稳定，很少出现崩溃.

（4）插件管理很到位，能准确定位出问题的插件.

（5）插件的生态圈发展速度特别快，一大堆好用的插件等着用户去探索

主要槽点

（1）性能问题，启动速度很慢

（2）打开大文件是会出现 cpu 占用过高的问题

（3）目前相比于 sublime 技术层面还不够成熟,有不少 bug

4.情况分析

个人情况：

Sublime、Notepad++、VC++6.0、Codeblock、Hbuilder（云打包）。

项目使用：

《前端规范》（链接）： *IDE 工具 atom 或者 vscode。对于其他 IDE 或者编译器 code 出来的代码如有出现格式混乱问题，不予 merge，直至修改为止。*

5.总结

这三个编辑器,**各有所长**,也各有自己的不足,针对不同的开发人员,可能需要根据自己的喜好来选择,不过幸运的是,这三个编辑器的**使用方式大同小异**,基本上可以**无痛的迁移**到任何一个编辑器上。

从长远发展角度来讲,Atom 和 vscode 是由**两大巨头维护主推**的**开源**项目,开源的力量是巨大的,因此其发展速度注定会远超 sublime。

受目前个人笔记本配置限制，暂用 Sublime，后期根据项目实际逐渐向 Atom 迁移。

HTML5+CSS3 整体回顾：https://www.jianshu.com/p/c3e1e39890fb

二、HTML5

1. H5 与 HTML5

H5：一个解决方案，包含一定特效、看起来酷炫的移动端 onepage 网站的解决方案。（「HeiHei Hai Hui Hua（嘿嘿还会滑）」）。

H5 是一个技术合集，却被意会成了一项技术，变成可以在质上而不是量上描述的概念。

HTML5：互联网领域缩写词，全称 HyperText Markup Language 5（超文本标记语言 5），HTML5 并不是**一项技术**，而是**一个标准**。

2. 背景与应用

HTML5 的**设计目的**是为了在移动设备上支持多媒体。

HTML5 是 W3C 与 WHATWG 合作的结果，WHATWG 指 Web Hypertext Application Technology Working Group。

WHATWG 致力于 web 表单和应用程序，而 W3C 专注于 XHTML 2.0。在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。

应用：多媒体、动画、本地数据存储、web APP 缓存引用、拖放、表单验证、XHTMLHttpRequest 等。

个人认为，HTML5 更多的是对 HTML 的增强，结合 CSS3，将一些原本需要使用 JavaScript 等来实现的东西，通过 HTML 来完成，追求学习的简易、使用的高效、功能的强大。

3. HTML5 API：

为了更好地处理今天的互联网应用，HTML5 添加了很多新元素及功能，比如：图形的绘制，多媒体内容，更好的页面结构，更好的形式处理，和几个 api 拖放元素，定位，包括网页、应用程序缓存，存储，网络工作者等。

HTML5API 盘点：Canvas、Audio 和 Video、Geolocation（地理定位功能）、WebSockets（实现了浏览器与服务器**全双工通信**）、Web storage（localStorage、sessionStorage 和离线存储 manifest，localStorage 可用于不同标签页数据传递；离线存储详见下图）、Communication（跨文档消息通信，postMessage API）、Web Workers（后台处理、多线程支持、监听由后台服务器**广播的新闻消息**）、requestAnimationFrame（以前不够了解，参见这里）。

首先，需要在页面头加入manifest属性：

```
<!DOCTYPE HTML>
<html manifest = "cache.manifest">
...
</html>
```

然后cache.manifest文件的书写方式为：

```
CACHE MANIFEST
#v0.11

CACHE:

js/app.js
css/style.css

NETWORK:
resourse/logo.png

FALLBACK:
/ /offline.html
```

离线存储的manifest一般由三个部分组成:

1.CACHE:表示需要离线存储的资源列表，由于包含manifest文件的页面将被自动离线存储，所以不需要把页面自身也列出来。

2.NETWORK:表示在它下面列出来的资源只有在在线的情况下才能访问，他们不会被离线存储，所以在离线情况下无法使用这些资源。不过，如果在CACHE和NETWORK中有一个相同的资源，那么这个资源还是会被离线存储，也就是说CACHE的优先级更高。

3.FALLBACK:表示如果访问第一个资源失败，那么就使用第二个资源来替换他，比如上面这个文件表示的就是如果访问根目录下任何一个资源失败了，那么就去访问offline.html。

除以上列出的 API，HTML5 新标签、新特性还包括：HTML5 <!DOCTYPE>；新的特殊内容元素，比如 article、footer、header、nav、section；新的表单控件、新表单元素、 新属性、新输入类型，自动验证等。

4. HTML5 识途

参考：HTML5 完整版手册 http://www.php.cn/course/34.html

（1）浏览器支持

IE8 及以下不支持 HTML5、CSS3 Media，可使用 html5shiv.min.js

及 Response.min.js 静态包。

IE9 以下版本浏览器兼容HTML5的方法，使用百度静态资源的html5shiv包：

```
<!--[if lt IE9]>
<script src="http://apps.bdimg.com/libs/html5shiv/3.7/html5shiv.min.js"></script>
<![endif]-->
```

载入后，初始化新标签的CSS：

```
/*html5*/
article,aside,dialog,footer,header,section,footer,nav,figure,menu{display:block}
```

所有浏览器（包括旧的和最新的），对无法识别的元素会作为内联元素自动处理，为了能让旧版本的浏览器正确显示这些元素，可以设置 CSS 的 display 属性值为 block。同理，也可以为 HTML 添加新的元素，并使用 CSS 设置样式。

（2）新元素

<canvas>：定义图形，基于 JavaScript 的绘图 API（标签只是图形容器，必须使用脚本来绘制图形。）；

新多媒体元素：<audio>、<video>、<source>（<video>/<audio>元素支持多个 <source> 元素. <source> 元素可以链接不同的文件。浏览器将使用第一个可识别的格式）、<embed>（嵌入的内容，如插件）、<track>（规定外部文本轨道）；

新表单元素：<datalist>（选项列表，定义 input 可能的值）、<keygen>（密钥对生成器字段）、<output>（不同类型的输出）；

新的语义和结构元素：<article>、<aside>、<command>、<details>、<dialog>、<summary>、<figure>、<figcaption>、

<footer>、<header>、<meter>、<nav>、<progress>、<section>、<time>、<wbr>（如果单词太长，或者担心浏览器会在错误的位置换行，可以使用 <wbr> 元素来添加 Word Break Opportunity（单词换行时机））；

（3）移除的元素

<acronym>：HTML5 中使用 <abbr> 标签；<basefont>：标签定义文档中所有文本的默认颜色、大小和字体，改用 CSS 设置；<big>、 、<font>、<frame>、<frameset>、<noframes>、<strike>（添加删除线，用 <del> 标签代替）、<tt>（打字机文本，CSS 代替）；

（4）HTML5 Canvas

①创建一个画布（Canvas）（默认情况下 <canvas> 元素没有边框和内容。）；

②使用 JavaScript 来绘制图像

找到 <canvas> 元素；创建对象：getContext("2d") 对象是内建的 HTML5 对象，拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法；绘制(beginPath、moveTo、lineTo、fill、stroke 绘制当前路径、arc(x,y,r,start,stop)、Gradient 渐变、drawImage 等）；

SVG 与 Canvas

SVG：基于 XML 的可伸缩矢量图形 (Scalable Vector Graphics)；GitHub 官网的 logo 用的 SVG 生成。

SVG 的优势：SVG 图像可通过文本编辑器来创建和修改；可被搜索、索引、脚本化或压缩；是可伸缩；可在任何的分辨率下被高质量地打印；可在图像质量不下降的情况下被放大；

> SVG 是一种使用 XML 描述 2D 图形的语言。
>
> Canvas 通过 JavaScript 来绘制 2D 图形。
>
> SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。
>
> 在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。
>
> Canvas 是逐像素进行渲染的。在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

SVG 与 Canvas 的区别

Canvas	SVG
依赖分辨率 不支持事件处理器 弱的文本渲染能力 能够以 .png 或 .jpg 格式保存结果图像 最适合图像密集型的游戏，其中的许多对象会被频繁重绘	不依赖分辨率 支持事件处理器 最适合带有大型渲染区域的应用程序（比如谷歌地图） 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快） 不适合游戏应用

（5）HTML5 MathML

HTML5 可以在文档中使用 MathML 元素，对应的标签是$...$；

MathML 是数学标记语言，是一种基于 XML（标准通用标记语言的子集）的标准，用来在互联网上书写数学符号和公式的置标语言。

（6）HTML5 拖放（Drag 和 drop）

设置元素为可拖放（draggable="true"）；

拖动什么 - ondragstart 和 setData();

放到何处 - ondragover;

进行放置 - ondrop（把被拖元素通过 appendChild 追加到放置元素中）；

需使用 preventDefault()阻止默认事件;

（7）HTML5 Geolocation（地理定位）

使用 getCurrentPosition() 方法来获得用户的位置（需要用户授权）；检测浏览器支持情况（navigator.geolocation）、position.coords.latitude 、 position.coords.longitude 、coords.accuracy、coords.altitude、处理错误和拒绝（错误代码：Permission denied - 用户不允许地理定位、Position unavailable - 无法获取当前位置、Timeout - 操作超时）、在地图中显示结果;

Geolocation 对象其他方法：

watchPosition() - 返回用户的当前位置，并继续返回用户移动时的更新位置（就像汽车上的 GPS）；

clearWatch() - 停止 watchPosition() 方法;

（8） Video(视频)与 Audio(音频)

产生：大多数视频/音频是通过插件（比如 Flash）来显示的，然而，并非所有浏览器都拥有同样的插件，HTML5 规定了一种通过 video/audio 元素来包含视频/音频的标准方法。元素提供播放、暂停和音量控件。

（9）新的 Input 类型

HTML5 拥有多个新的表单输入类型。这些新特性提供了更好的输入控制和验证：color、date、datetime、datetime-local、email、month、number（max、min 、Step、value ）、range（max、min 、Step、value ）、search、tel（手机端实现点击跳转到电话拨号页）、time、url、week 等。

（10）HTML5 表单属性

HTML5 的 <form> 和 <input>标签添加了几个新属性。

<form>新属性：autocomplete、novalidate;

<input>新属性：autocomplete、autofocus、form、formaction、formenctype、formmethod、formnovalidate、formtarget、height and width、list、min and max、multiple、pattern (regexp)、placeholder、required、step 等。

（11）HTML5 语义元素

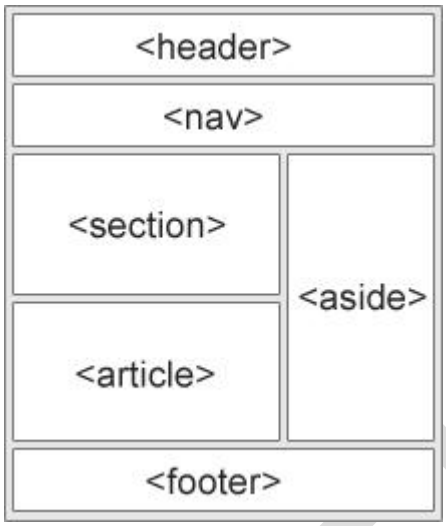
语义= 意义.

语义元素 = 元素的意义.

一个语义元素能够清楚的描述其意义给浏览器和开发者。

无语义元素实例: 和 - 无需考虑内容.

语义元素实例: <form>, <table>, and <img> - 清楚的定义了它的内容。



<figure>标签规定独立的流内容（图像、图表、照片、代码等等），元素的内容应该与主内容相关，但如果被删除，则不应对文档流产生影响。

<figcaption>（**非块元素**） 标签定义 <figure> 元素的标题，<figcaption>元素应该被置于 "figure" 元素的第一个或最后一个子元素的位置。

（12）HTML5 Web 存储

前端 HTML5 几种存储方式的总结：

https://www.cnblogs.com/LuckyWinty/p/5699117.html

关于 Cookie、session 和 Web Storage：

https://juejin.im/post/5ad5b9116fb9a028e014fb19

浅谈 session,cookie,sessionStorage,localStorage 的区别及应用场景

Web Storage--HTML5 本地存储：

https://segmentfault.com/a/1190000007926921

聊一聊前端存储那些事儿：

https://segmentfault.com/a/1190000005927232

session 和 cookie：

- 1、 **由于HTTP协议是无状态的协议，所以服务端需要记录用户的状态时，就需要用某种机制来识具体的用户，这个机制就是Session**.典型的场景比如购物车，当你点击下单按钮时，由于HTTP协议无状态，所以并不知道是哪个用户操作的，所以服务端要为特定的用户创建了特定的Session，用用于标识这个用户，并且跟踪用户，这样才知道购物车里面有几本书。这个Session是保存在服务端的，有一个唯一标识。**在服务端保存Session的方法很多，内存、数据库、文件都有。**集群的时候也要考虑Session的转移，在大型的网站，一般会有专门的Session服务器集群，用来保存用户会话，这个时候 Session 信息都是放在内存的，使用一些缓存服务比如Memcached之类的来放 Session。
- 2、 **思考一下服务端如何识别特定的客户**?: 这个时候Cookie就登场了。每次HTTP请求的时候，客户端都会发送相应的Cookie信息到服务端。**实际上大多数的应用都是用 Cookie来实现Session跟踪的**，第一次创建Session的时候，服务端会在HTTP协议中告诉客户端，**需要在 Cookie 里面记录一个Session ID**，以后每次请求把这个会话ID发送到服务器，我就知道你是谁了。

> **session 的运行依赖 session id，而 session id 是存在 cookie**中的

Localstorage 用法：

背景：HTML5 之前，存储主要是用 cookies。cookies 缺点有在请求头上带着数据，大小是 4k 之内。主 Domain 污染。主要应用如购物车、客户登录等。对于 IE 浏览器有 UserData，大小是 64k,只有 IE 浏览器支持。

HTML 存储方式目标：解决 4k 的大小问题；解决请求头常带存储信息的问题；解决关系型存储的问题；跨浏览器。

存储方式：以键值对(Key-Value)的方式存储，永久存储，永不失效，除非手动删除；大小每个域名 5M。

检测方法：window.localStorage；

常用的 API：

localStorage.getItem(key)	取记录
localStorage.setItem(key, value)	设置记录
localStorage.removeItem(key)	移除记录
localStorage.key(index)	取 key 所对应的值
localStorage.clear( )	清除记录

存储的内容：只要是能序列化成字符串的内容都可以存储,如：数组，图片，json，样式，脚本等；

HTML5 的本地存储 API 中的 localStorage 与 sessionStorage 在使用方法上是相同的，区别在于 sessionStorage 在关闭页面后即被清空，而 localStorage 则会一直保存。

（13）HTML5 Web SQL

关系数据库，通过 SQL 语句访问。

三个核心方法：

openDatabase：这个方法使用现有的数据库或者新建的数据库创建一个数据库对象。

打开数据库：

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
```

//openDatabase() 方法对应的五个参数分别为：数据库名称、版本号、描述文本、数据库大小、创建回调

transaction：这个方法让我们能够控制一个事务，以及基于这种情况执行提交或者回滚。

executeSql：这个方法用于执行实际的 SQL 语句（基本上都是用

SQL 语句进行数据库的相关操作，**前期已对 SQL 语句有过一些接触，**能够看懂和写出简单的 SQL 语句）。

实例：

```
var db = openDatabase('mydb', '1.0', 'Test DB', 2 * 1024 * 1024);
db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS LOGS (id unique, log)');
     tx.executeSql('INSERT INTO LOGS (id, log) VALUES (1, "菜鸟教程")');
     tx.executeSql('INSERT INTO LOGS (id, log) VALUES (2, "www.runoob.com")');
});
```

索引数据库（IndexedDB）API（作为 HTML5 的一部分）对创建具有丰富本地存储数据的数据密集型的离线 HTML5 Web 应用程序很有用。同时它还有助于本地缓存数据，使传统在线 Web 应用程序（比如移动 Web 应用程序）能够更快地运行和响应。

（14）HTML5 应用程序缓存

使用 HTML5，通过创建 cache manifest 文件，可以轻松地创建 web 应用的离线版本。

缓存带来的优势：

离线浏览 - 用户可在应用离线时使用

速度 - 已缓存资源加载得更快

减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。

一旦应用被缓存，它就会保持缓存直到发生下列情况：

用户清空浏览器缓存

manifest 文件被修改

由程序来更新应用缓存

注意：以 "#" 开头的是注释行，但也可满足其他用途。应用的缓存会在其 manifest 文件更改时被更新。如果您编辑了一幅图片，或者修改了一个 JavaScript 函数，这些改变都不会被重新缓存。更新注释行中的日期和版本号是一种使浏览器重新缓存文件的办法。

关于应用程序缓存的说明：

请留心缓存的内容；一旦文件被缓存，则浏览器会继续展示已缓存的版本，即使您修改了服务器上的文件。为了确保浏览器更新缓存，您需要更新 manifest 文件。

离线缓存与传统浏览器缓存区别：

①离线缓存是针对整个应用，浏览器缓存是单个文件

②离线缓存断网了还是可以打开页面，浏览器缓存不行

③离线缓存可以主动通知浏览器更新资源

（15）HTML5 Web Workers

当在 HTML 页面中执行脚本时，页面的状态是不可响应的，直到脚本已完成；web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行，一般用于更耗费 CPU 资源的任务。

（16）HTML5 SSE

HTML5 服务器发送事件（server-sent event）允许网页获得来自服务器的更新。

Server-Sent 事件指的是网页自动获取来自服务器的更新，以前也可能做到这一点，前提是网页不得不询问是否有可用的更新。通过服务器发送事件，更新能够自动到达。

EventSource 对象用于接收服务器发送事件通知。

SSE 还需要能够发送数据更新的服务器（比如 PHP 和 ASP）。服务器端事件流的语法是非常简单的。把 "Content-Type" 报头设置为 "text/event-stream"，就可以开始发送事件流了。

可以用 onmessage（当接收到消息）、onopen（当通往服务器的连接被打开）、onerror（当发生错误） 事件来获取消息。

（17）HTML5 WebSocket

> WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。
>
> 在 WebSocket API 中，浏览器和服务器只需要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。
>
> 浏览器通过 JavaScript 向服务器发出建立 WebSocket 连接的请求，连接建立以后，客户端和服务器端就可以通过 TCP 连接直接交换数据。
>
> 当你获取 Web Socket 连接后，你可以通过 **send()** 方法来向服务器发送数据，并通过 **onmessage** 事件来接收服务器返回的数据。

为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息"Upgrade: WebSocket"表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生

应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。

WebSocket 属性：

Socket.readyState：

只读属性 **readyState** 表示连接状态，可以是以下值：

0	表示连接尚未建立；
1	表示连接已建立，可以进行通信；
2	表示连接正在进行关闭；
3	表示连接已经关闭或者连接不能打开。

Socket.bufferedAmount：

只读属性 bufferedAmount 已被 send() 放入正在队列中等待传输，但是还没有发出的 UTF-8 文本字节数。

WebSocket 事件

以下是 WebSocket 对象的相关事件。

事件	事件处理程序	描述
open	Socket.onopen	连接建立时触发
message	Socket.onmessage	客户端接收服务端数据时触发
error	Socket.onerror	通信发生错误时触发
close	Socket.onclose	连接关闭时触发

WebSocket 方法

Socket.send() 使用连接发送数据

Socket.close() 关闭连接

在标准 HTML5 中，〈html〉和〈body〉标签是可以省略的；省略〈html〉或〈body〉在 DOM 和 XML 软件中会崩溃（缺少根元素？）。

在标准 HTML5 中，〈head〉标签是可以省略的；默认情况下，浏览器会将〈body〉之前的内容添加到一个默认的〈head〉元素上。

三、CSS

1. css calc（）

参考：《css 揭秘》、CSS3 的 calc()使用、calc() 如何工作。

（1）calc（）的产生与 box-sizing

平时在制作页面的时候，总会碰到有的元素是 100%的宽度，如果元素宽度为 100%时，其自身如果有别的盒模型属性设置，那将导致盒子撑破。比如说，有一个边框，或者说有 margin 和 padding，这些都会让你的盒子撑破（标准模式下，除 IE 怪异模式）。平时可能只能通过改变结构来实现。就算你通过繁琐的方法实现了，但有于浏览器的兼容性而导致最终效果不一致。CSS3 属性中的 box-sizing（content-box|border-box|inherit）在一定程度上解决了这样的问题，其实 calc()函数功能实现上面的效果更简单。

（2）什么是 calc()?

calc 是英文单词 calculate(计算)的缩写，是 css3 的一个新增的功能，用来指定元素的长度。比如说，你可以使用 calc()给元素的 border、margin、pading、font-size 和 width 等属性设置动态值。

为何说是动态值呢?因为我们使用的**表达式来得到的值**。不过 calc()最大的好处就是用在**流体布局**上，可以通过 calc()计算得到元素的宽度。

（3）calc()能做什么，怎么用?

calc()能让你给元素的做计算，你可以给一个 div 元素，使用百分比、em、px 和 rem 单位值计算出其宽度或者高度，比如说“width:calc(50% + 2em)”，这样一来你就不用考虑元素 DIV 的宽度值到底是多少，而把这个烦人的任务交由浏览器去计算。

运算规则：

calc()使用通用的数学运算规则，但是也提供更智能的功能：

- 使用“+”、“-”、“*”和“/”四则运算；
- 可以使用百分比、px、em、rem等单位；
- 可以混合使用各种单位进行计算；
- 表达式中有“+”和“-”时，其前后必须要有空格，如"widht: calc(12%+5em)"这种没有空格的写法是错误的；
- 表达式中有“*”和“/”时，其前后可以没有空格，但建议留有空格。

calc() 函数可以用来对数值属性执行**四则运算**。比如，<length>，<frequency>，<angle>，<time>，<number> 或者 <integer> 数据类型，例：

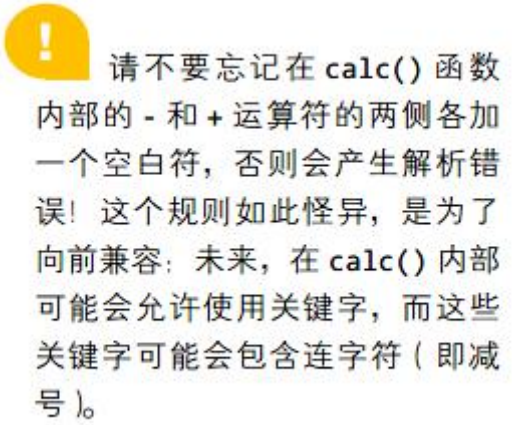
```
.foo {
    width: calc(50vmax + 3rem);
    padding: calc(1vw + 1em);
    transform: rotate( calc(1turn + 28deg) );
    background: hsl(100, calc(3 * 20%), 40%);
    font-size: calc(50vw / 3);
}
```

（4）calc()与 SASS

SASS 常用用法包括：变量（SASS 允许使用变量，所有变量以$开头）、计算功能、嵌套[选择器嵌套、&的嵌套（引用父元素）、属性嵌套]、继承（ @extend）、Mixin（@mixin 定义、@include 复用，mixin 的强大之处，在于可以指定参数和缺省值，这样我们就可以复用一些样式，只需要传递一个参数，就像调用一个函数一样！）、导入文件（@import）、注释（/*会保留到编译后的文件，单行注释//不会）。详见中文文档。

calc（）：首先，calc() 能够组合不同的单元。特别是，我们可以混合计算绝对单位（比如百分比与视口单元）与相对单位（比如像素）。例如，我们可以创造一个表达式，用一个百分比减掉一个像素值。第二，使用 calc()，计算值是表达式它自己，而非表达式的结果。当使用 CSS 预处理器做数学运算时，给定值为表达式的结果。这意味着浏览器中的值可以更加灵活，能够响应视口的改变。我们能够给元素设定一个高度为视口的高度减去一个绝对值，它将随视口的改变进行调节。

注意：calc() 函数可以嵌套；对不支持 calc()的，使用一个固定值作为降级方案；实际使用时，同样需要添加浏览器的前缀。



（5）使用场景

属性	含义
animation（动画）	用于设置动画属性，他是一个简写的属性，包含6个属性
transition（过渡）	用于设置元素的样式过度，和animation有着类似的效果，但细节上有很大的不同
transform（变形）	用于元素进行旋转、缩放、移动或倾斜，和设置样式的动画并没有什么关系，就相当于color一样用来设置元素的“外表”
translate（移动）	translate只是transform的一个属性值，即移动。

知道元素尺寸的垂直居中（top、left：50%减去半宽或半高）；使用 rem，calc() 函数能够用来创建一个基于视口的栅格；calc()使计算更加清晰。

2. css transition 与 animate

（1）animation、transition、transform、translate 傻傻分不清：

（2）Transition（过渡）

什么叫过渡？字面意思上来讲，就是元素从这个属性(如 color)的某个值如(red)过渡到这个属性(color)的另外一个值(green)，这是一个状态的转变，需要一种条件来触发这种转变，比如我们平时用到的:hoever、:focus、:checked、媒体查询或者 JavaScript。常见触发方式及用法详见这里。

在执行的过程中声明关键帧的动画；transition 产生动画的条件是 transition 设置的 property 发生变化，这种动画的特点是需要“一个驱动力去触发”，有着以下几个不足：

①这类动画必须要事件触发，所以没法在网页加载时自动发生

②是一次性的，不能重复发生，除非一再触发

③只能定义开始状态和结束状态，不能定义中间状态，也就是说只有两个状态

④一条 transition 规则，只能定义一个属性的变化，不能涉及多个属性。

语法：**transition: property duration timing-function delay;**

值	描述
transition-property	规定设置过渡效果的 CSS 属性的名称
transition-duration	规定完成过渡效果需要多少秒或毫秒
transition-timing-function	规定速度效果的速度曲线
transition-delay	定义过渡效果何时开始

transition-property 指定需要过渡的 CSS 属性。并不是所有属性都能过渡的，只有能数字量化的 CSS 属性才能过渡[颜色系、数字系、01 系（如 visibility，0 表示隐藏，1 表示显示）、渐变、阴影、all（表示所有属性都将获得过渡效果）]。

transition-duration 过渡需要的时间，单位可指定 s 秒，也可指定 ms 毫秒。默认值是 0，表示立刻变化。如果设置了多个过渡属

性，但每个属性的过渡时间都一样，设一个值即可，该值会应用到所有过渡属性上。

transition-timing-function 过渡函数，有 linear，ease，ease-in，ease-out，ease-in-out，cubic-bezier(n,n,n,n)，steps。其实它们都是贝赛尔曲线。如果预定义好的这些函数满足不了你的需求，可以通过 cubic-bezier(n,n,n,n)自定义平滑曲线。

steps 可以把过渡阶段分割成等价的几步。

transition-delay 延迟开始过渡的时间，默认值是 0，表示不延迟，立即开始过渡动作。单位是 s 秒或 ms 毫秒。该属性还能设负时间，意思是让过渡动作从该时间点开始启动，之前的过渡动作不显示。

隐式过渡（比如设成 em 的属性）、开关过渡和永久过渡（最终状态）、auto 过渡（必须靠 JS，通过 getComputedStyle 将 auto 转换为固定值，通过 style 设置该值，然后再触发 CSS 的过渡效果。）

（3）Animation（动画）

参考：CSS3 动画、CSS Animation for Beginners

在官方的介绍上介绍这个属性是 transition 属性的扩展，弥补了 transition 的很多不足（animation 是由多个 transition 的效果叠加，并且可操作性更强，能够做出复杂酷炫的效果）。

语法：**animation: name duration timing-function delay iteration-count direction play-state fill-mode;**

值	描述
name	用来调用@keyframes定义好的动画，与@keyframes定义的动画名称一致
duration	指定元素播放动画所持续的时间
timing-function	规定速度效果的速度曲线，是针对每一个小动画所在时间范围的变换速率
delay	定义在浏览器开始执行动画之前等待的时间，值整个animation执行之前等待的时间
iteration-count	定义动画的播放次数，可选具体次数或者无限(infinite)
direction	设置动画播放方向：normal(按时间轴顺序),reverse(时间轴反方向运行),alternate(轮流，即来回往复进行),alternate-reverse(动画先反运行再正方向运行，并持续交替运行)
play-state	控制元素动画的播放状态，通过此来控制动画的暂停和继续，两个值：running(继续)，paused(暂停)
fill-mode	控制动画结束后，元素的样式，有四个值：none(回到动画没开始时的状态)，forwards(动画结束后动画停留在结束状态)，backwords(动画回到第一帧的状态)，both(根据animation-direction轮流应用forwards和backwards规则)，注意与iteration-count不要冲突(动画执行无限次)

animation 与 transition 不同的是，keyframes 提供更多的控制，尤其是时间轴的控制，这点让 css animation 更加强大，使得 flash 的部分动画效果可以由 css 直接控制完成，而这一切，仅仅只需要几行代码，也因此诞生了大量基于 css 的动画库，用来取代 flash 的动画部分。

@keyframes 规则：@keyframes 定义关键帧，即动画每一帧执行的是什么，animation 是来定义动画的每一帧如何执行的。创建动画的**原理**是，将一套 CSS 样式逐渐变化为另一套样式。在动画过程中，可以多次改变这套 CSS 样式。

语法：

```
@keyframes animationname {
keyframes-selector {
css-styles;
                    }
}
```

animationname 是动画名字；keyframes-selector 帧选择器，用百分比定义，0%-100%，与 from-to 等价，作用是定义不同时间段的不同样式。

拓展：[译]详解 React Native 动画

简单一次性的动画可以使用 transition，比较逻辑清晰，可维护性较好。比较复杂的动画，这个时候便可以拿出 animation 开始你的表演，其实不仅仅用 css 能实现动画，用 js 同样可以操控元素的样式实现动画(setTimeout,setInterval,其中 requestAnimationFrame 能够更高性能地执行动画）。

3. css flex 布局

参考：一劳永逸的搞定 flex 布局、30 分钟学会 Flex 布局（知乎）、Flex 布局教程：语法篇、Flex 布局教程：实例篇

text-align: center、verticle-align: center 只能用于行内元素，对于块级元素的布局是无效的。在网页布局没有进入 CSS 的时

代，排版几乎是通过 table 元素实现的，在 table 的单元格里可以方便的使用 align、valign 来实现水平和垂直方向的对齐。

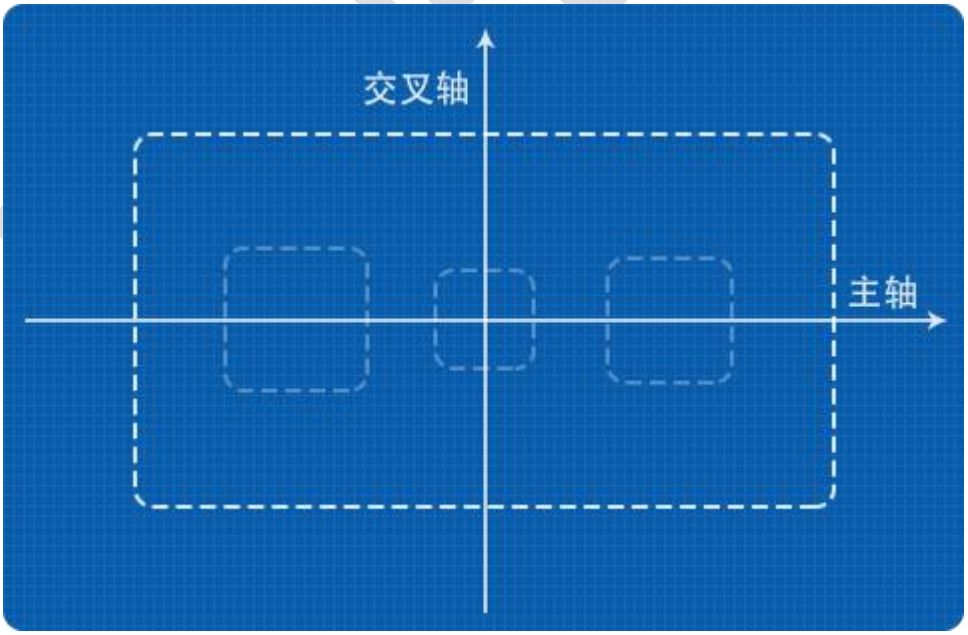
Flex 是 Flexible Box 的缩写，意为"弹性布局"。

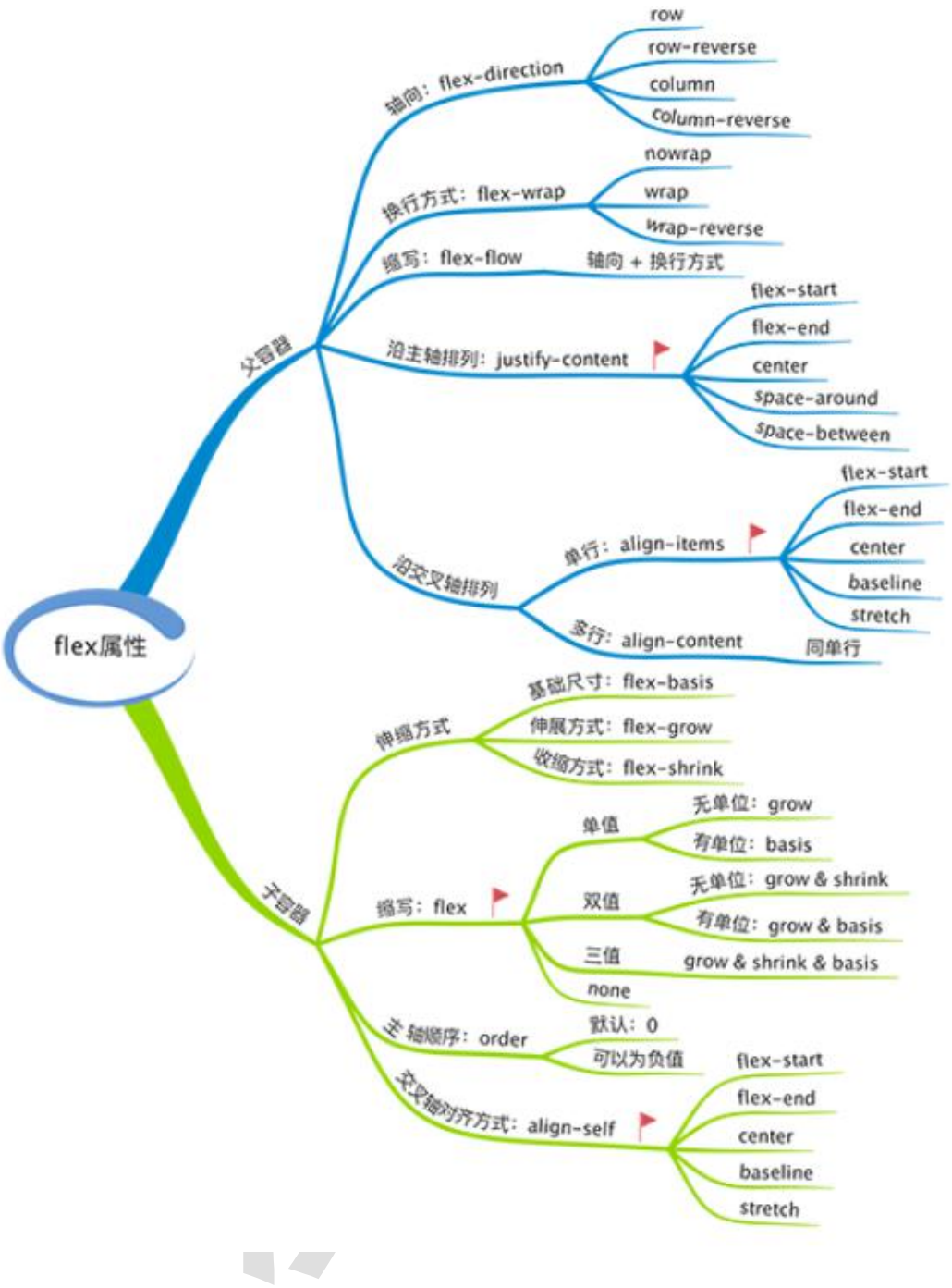
flex 水平垂直居中：

```
#dad { display: flex; justify-content: center; align-items: center; }
```

```
display: flex | inline-flex;        //可以有两种取值
```

（1）基本概念：




flex属性
父容器
轴向：flex-direction
row
row-reverse
column
column-reverse
换行方式：flex-wrap
nowrap
wrap
wrap-reverse
缩写：flex-flow
轴向 + 换行方式
沿主轴排列：justify-content
flex-start
flex-end
center
space-around
space-between
沿交叉轴排列
单行：align-items
flex-start
flex-end
center
baseline
stretch
多行：align-content
同单行
子容器
伸缩方式
基础尺寸：flex-basis
伸展方式：flex-grow
收缩方式：flex-shrink
缩写：flex
单值
无单位：grow
有单位：basis
双值
无单位：grow & shrink
有单位：grow & basis
三值
grow & shrink & basis
none
主轴顺序：order
默认：0
可以为负值
交叉轴对齐方式：align-self
flex-start
flex-end
center
baseline
stretch

（2）常用属性：

> 容器具有这样的特点：父容器可以统一设置子容器的排列方式，子容器也可以单独设置自身的排列方式，如果两者同时设置，以子容器的设置为准。

（3）父容器

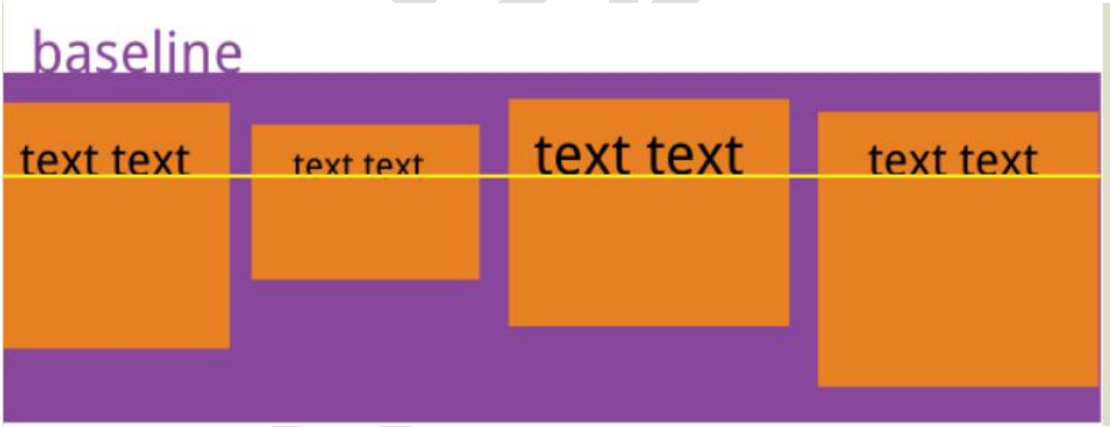
设置子容器沿主轴(水平)排列：justify-content(如 flex-start：起始端对齐；space-between：子容器沿主轴均匀分布，位于首尾两端的子容器与父容器相切；around：位于首尾两端的子容器到父容器的距离是子容器间距的一半。）

`justify-content` 属性用于定义如何沿着主轴方向排列子容器。

设置子容器如何沿交叉轴（垂直）排列：align-items

`align-items` 属性用于定义如何沿着交叉轴方向分配子容器的间距。

baseline：项目的第一行文字的基线对齐，以文字的底部为主，所有子容器向基线对齐，交叉轴起点到元素基线距离最大的子容器将会与交叉轴起始端相切以确定基线。



stretch：子容器沿交叉轴方向的尺寸拉伸至与父容器一致。

（4）子容器

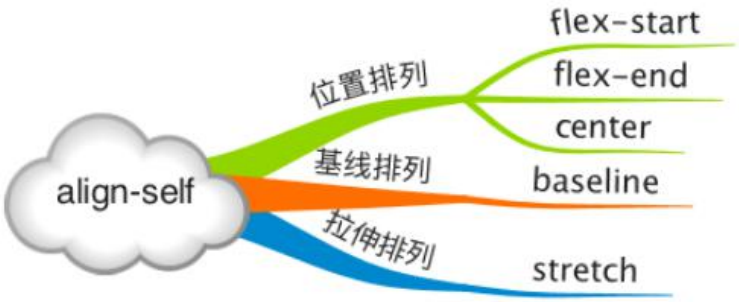
在主轴上如何伸缩：flex

子容器是有弹性的（flex 即弹性），它们会自动填充剩余空间，子容器的伸缩比例由 flex 属性确定。

单独设置子容器如何沿交叉轴排列：align-self（允许单个项目有与其他项目不一样的对齐方式）

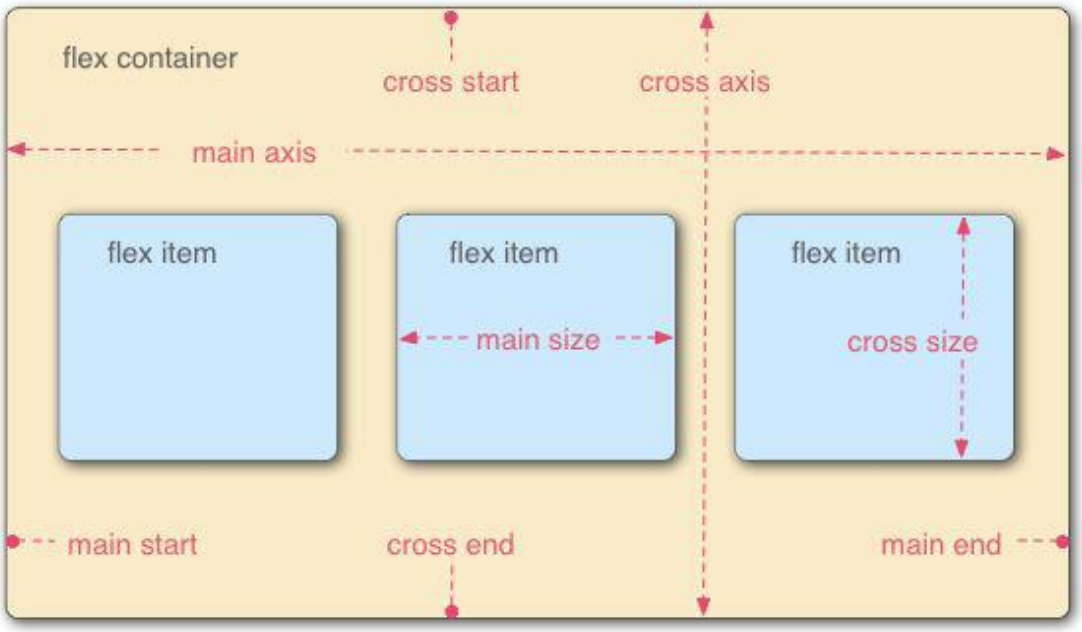
（5）轴

轴 包括 **主轴** 和 **交叉轴**，我们知道 justify-content 属性决定子容器沿主轴的排列方式，align-items 属性决定子容器沿着交叉轴的排列方式。那么轴本身又是怎样确定的呢？在 flex 布局中，flex-direction 属性（row（默认）、column、row-reverse、column-reverse）决定主轴的方向，交叉轴的方向由主轴确定（主轴



每个子容器也可以单独定义沿交叉轴排列的方式，此属性的可选值与父容器 `align-items` 属性完全一致，如果两者同时设置则以子容器的 `align-self` 属性为准。

沿逆时针方向旋转 90°）。



（6）Flex 进阶

①父容器：

设置换行方式：flex-wrap

决定子容器是否换行排列，不但可以顺序换行而且支持逆序换行。

容器的 flex-wrap 与子项的 flex-shrink、flex-grow 之间的关系：

1. 当 flex-wrap 为 wrap | wrap-reverse，且子项宽度和不及父容器宽度时，flex-grow 会起作用，子项会根据 flex-grow 设定的值放大（为0的项不放大）
2. 当 flex-wrap 为 wrap | wrap-reverse，且子项宽度和超过父容器宽度时，首先一定会换行，换行后，每一行的右端都可能会有剩余空间（最后一行包含的子项可能比前几行少，所以剩余空间可能会更大），这时 flex-grow 会起作用，若当前行所有子项的 flex-grow 都为0，则剩余空间保留，若当前行存在一个子项的 flex-grow 不为0，则剩余空间会被 flex-grow 不为0的子项占据
3. 当 flex-wrap 为 nowrap，且子项宽度和不及父容器宽度时，flex-grow 会起作用，子项会根据 flex-grow 设定的值放大（为0的项不放大）
4. 当 flex-wrap 为 nowrap，且子项宽度和超过父容器宽度时，flex-shrink 会起作用，子项会根据 flex-shrink 设定的值进行缩小（为0的项不缩小）。但这里有一个较为特殊情况，就是当这一行所有子项 flex-shrink 都为0时，也就是说所有的子项都不能缩小，就会出现讨厌的横向滚动条
5. 总结上面四点，可以看出不管在什么情况下，在同一时间，flex-shrink 和 flex-grow 只有一个能起作用，这其中的道理细想起来也很浅显：空间足够时，flex-grow 就有发挥的余地，而空间不足时，flex-shrink 就能起作用。当然，flex-wrap 的值为 wrap | wrap-reverse 时，表明可以换行，既然可以换行，一般情况下空间就总是足够的，flex-shrink 当然就不会起作用

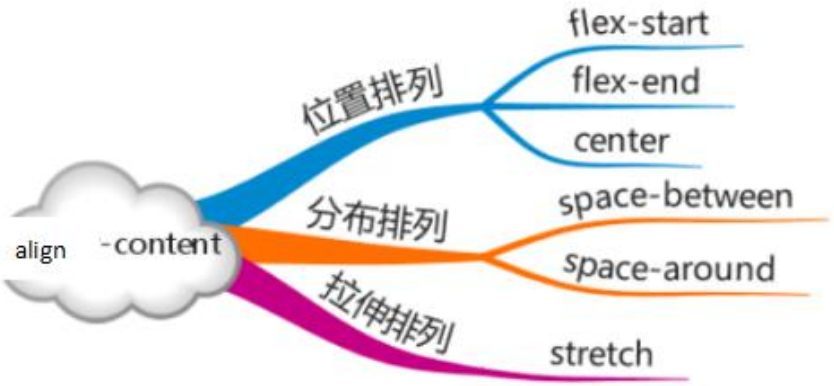
轴向与换行组合设置：flex-flow

flow 即流向，也就是子容器沿着哪个方向流动，流动到终点是否允许换行，比如 `flex-flow: row wrap` ，`flex-flow` 是一个复合属性，相当于 flex-direction 与 flex-wrap 的组合，可选的取值如下：

- `row`、`column` 等，可单独设置主轴方向
- `wrap`、`nowrap` 等，可单独设置换行方式
- `row nowrap`、`column wrap` 等，也可两者同时设置

多行沿交叉轴对齐：align-content

当子容器多行排列时，设置行与行之间的对齐方式。



当你 flex-wrap 设置为 nowrap 的时候，容器仅存在一根轴线，因为项目不会换行，就不会产生多条轴线。

当你 flex-wrap 设置为 wrap 的时候，容器可能会出现多条轴线，这时候你就需要去设置多条轴线之间的对齐方式了。

②子容器：

设置基准大小：flex-basis（在不伸缩的情况下子容器的原始尺寸。主轴为横向时代表宽度，主轴为纵向时代表高度。）

设置扩展比例：flex-grow

设置收缩比例：flex-shrink

设置排列顺序：order（替代 float，改变子容器的排列顺序，覆盖 HTML 代码中的顺序，默认值为 0，可以为负值，数值越小排列越靠前。）

当时设置 flex 布局之后，子元素的 float、clear、vertical-align 的属性将会失效。

（7）常见布局方式

骰子布局、网格布局、圣杯布局、输入框的布局、悬挂式布局、固定的底栏、流式布局等。

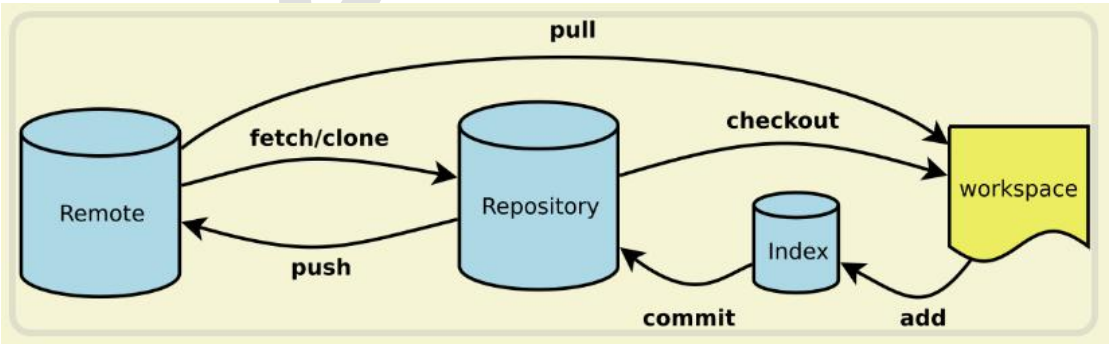
四、git 与 GitHub

参考：借助 GitHub 托管你的项目代码、一个小时学会 Git、图解 Git、这些 GIT 经验够你用一年了（知乎）、全面理解 Git、《Git 教程》

1. 术语

术语	定义
仓库（Repository）	一个仓库包括了所有的版本信息、所有的分支和标记信息。在Git中仓库的每份拷贝都是完整的。仓库让你可以从中取得你的工作副本。
分支（Branches）	一个分支意味着一个独立的、拥有自己历史信息的代码线（code line）。你可以从已有的代码中生成一个新的分支，这个分支与剩余的分支完全独立。默认的分支往往是叫master。用户可以选择一个分支，选择一个分支执行命令 `git checkout branch`.
标记（Tags）	一个标记指的是某个分支某个特定时间点的状态。通过标记，可以很方便的切换到标记时的状态，例如2009年1月25号在testing分支上的代码状态
提交（Commit）	提交代码后，仓库会创建一个新的版本。这个版本可以在后续被重新获得。每次提交都包括作者和提交者，作者和提交者可以是不同的人
修订（Revision）	用来表示代码的一个版本状态。Git通过用SHA1 hash算法表示的id来标识不同的版本。每一个 SHA1 id都是160位长，16进制标识的字符串。最新的版本可以通过HEAD来获取。之前的版本可以通过"HEAD~1"来获取，以此类推。

2. 常用 Git 命令清单



检出命令 git checkout 是 git 最常用的命令之一，同时也是一个很危险的命令，因为这条命令会**重写工作区**。

git reset：**重置暂存区**文件，与上一次 commit 保持一致，但**工作区不变**。

git配置代理命令

```
// 查看当前代理设置
git config --global http.proxy
// 设置当前代理为 http://127.0.0.1:1080 或 socket5://127.0.0.1:1080
git config --global http.proxy 'http://127.0.0.1:1080'
git config --global https.proxy 'http://127.0.0.1:1080'
git config --global http.proxy 'socks5://127.0.0.1:1080'
git config --global https.proxy 'socks5://127.0.0.1:1080' /
/ 删除 proxy git config --global --unset http.proxy
git config --global --unset https.proxy
```

五、http 抓包工具

参考：HTTP 最强资料大全

1. 浏览器自带抓包工具

与内核有关，如：360 浏览器（极速 webkit、兼容 Trident（IE））。

HTTP 协议 Header 详解、Chrome DevTools 之 Network，网络加载分析利器

Network主要有5个视窗，分别有不同的功能：

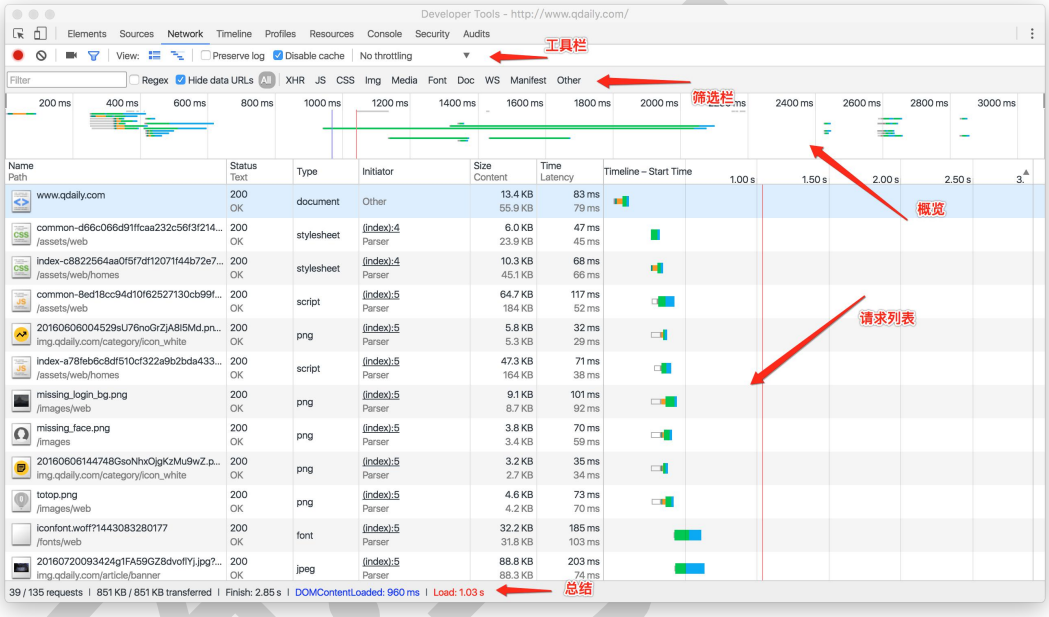
`Controls 工具栏` ：用来控制Network的功能及外观。

`Filters 筛选栏` ：根据筛选条件筛选请求列表，按住 `command/ctrl` 键可多选。

`Overviews 概览` ：资源被加载过来的时间线，如果多条时间线垂直堆叠，表示多个资源被并行加载。

`Request Table 请求列表` ：该视窗列出了所有的资源请求，默认按时间顺序排序，点击某个资源，可以查看更详细的信息。

`Summary 总览` ：汇总了请求数量，传输数据大小，加载时间等信息。



默认情况下，`Request Table 请求列表` 展示如下信息，当然，在请求列表的表头右键可以配置请求列表显示的内容。

`Name` ：资源的名称。

`Status` ：HTTP的状态码。

`Type` ：资源的MIME类型。

`Initiator` ：表示请求的上游，即发起者。 `Parser` 表示是HTML解析器发起的请求；`Redirect` 表示是HTTP跳转发起的请求； `Script` 表示是由脚本发起的请求； `Other` 表示是由其他动作发起的请求，比如用户跳转或者在导航栏输入地址等。

`Size` ：请求的大小，包括响应头和响应体的内容。

`Time` ：请求的时间，从请求开始到请求完全结束。

`Timeline` ：请求的时间线。

点击请求列表某个请求的名称，可以查看该请求的详细信息。详细信息主要有4个方面：

`Headers` ：资源的HTTP头

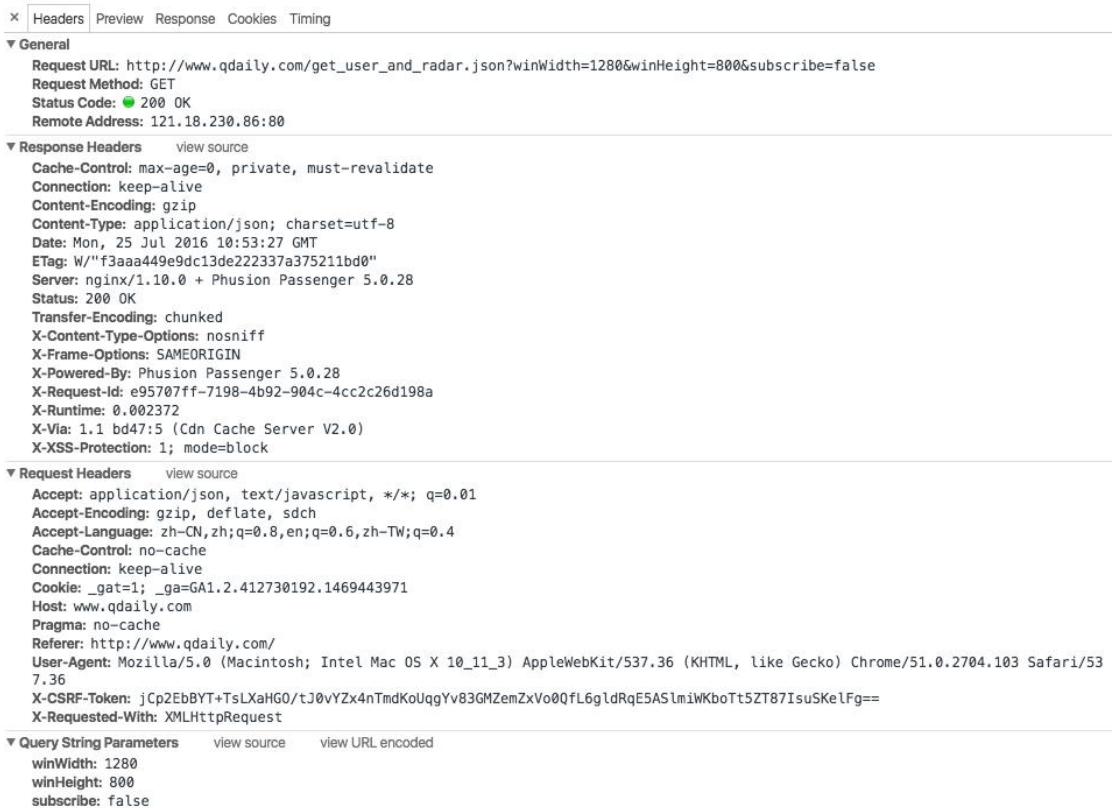
`Preview` ：预览JSON/image/text资源

`Response` ：资源的HTTP响应头

`Timing` ：资源的请求生命周期

`Cookies` ：查看HTTP请求头和响应头附带的cookie信息

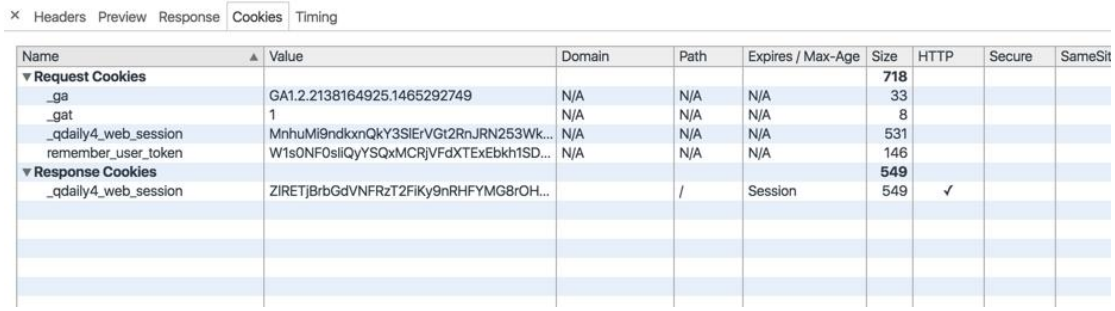
HTTP 头：包含了资源的请求 URL，HTTP 方法，响应的状态码。此外，还列出了 HTTP 请求头和响应头及其值，以及请求参数。



HTTP 响应：

HTTP响应

Cookies：



`Name` ：cookie的名称。

`Value` ：cookie的值。

`Domain` ：cookie所属域名。

`Path` ：cookie所属URL。

`Expire/Max-Age` ：cookie的存活时间。

`Size` ：cookie的字节大小。

`HTTP` ：表示cookie只能被浏览器设置，而且JS不能修改。

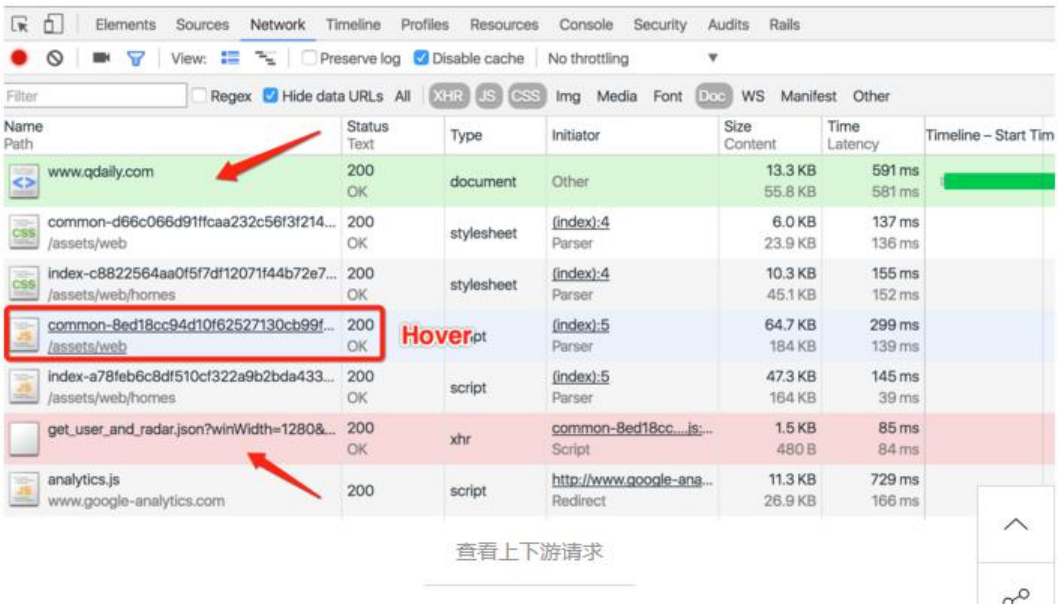
`Secure` ：表示cookie只能在安全连接上传输。

Timing： 查看资源请求的生命周期，包含 Queing/Stalled/Request/Response/Request sent/Waiting/Content Download 各个阶段。

Timeing

如何查看资源请求的上游和下游？

按时shift键，鼠标hover在请求上，可以查看请求的上游和下游，如下图所示，hover在 `common.js` 上，可以看到有一个绿色请求、一个红色请求。其中绿色请求表示 `common.js` 的上游请求，即谁触发了 `common.js` 请求，红色请求表示 `common.js` 的下游请求，即 `common.js` 又触发了什么请求。



查看上下游请求

2. 抓包工具

（1）Wireshark

参考：Wireshark 抓包详细图文教程、一站式学习 Wireshark、图解 TCP 三次握手与四次分手

大学《计算机网络》课程曾有接触和使用。

使用 wireshark 的人必须了解网络协议，为了安全考虑，wireshark 只能查看封包，而不能修改封包的内容，或者发送封包。

Wireshark 与 Fiddler：Fiddler 是在 windows 上运行的程序，专门用来捕获 HTTP，HTTPS 的。wireshark 能获取 HTTP，也能获取 HTTPS，但是不能解密 HTTPS，所以 wireshark 看不懂 HTTPS 中的内容。如果是处理 HTTP,HTTPS 还是用 Fiddler, 其他协议比如 TCP,UDP 就用 wireshark。

（2） Fiddler

参考：十分钟学会 Fiddler、手把手 Fiddler 掌握

Fiddler 是一个 http 抓包改包工具，fiddle 英文中有“欺骗、伪造”之意，与 wireshark 相比它更轻量级，上手简单，因为只能抓 http 和 https 数据包，所以在针对 http 和 https 数据包的抓取上它更加专业。

不仅可以记录客户端和服务器的 http(s)请求，还能设置断点，修改请求和响应的数据，模拟弱网络环境。可以安装插件对 Fiddler 现有的功能进行扩展，甚至编写脚本实现一些自动化操作。

六、ES6

《JavaScript 高级程序设计》、重新介绍 JavaScript（JS 教程）

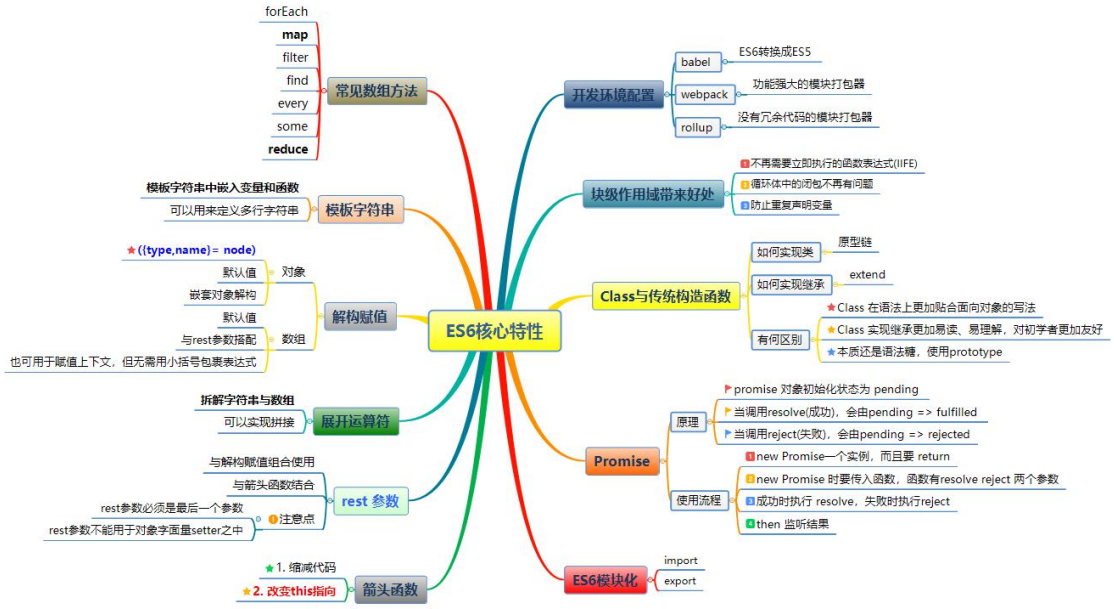
（MDN）、ECMAScript 6 入门、ES6 所改良的 javascript“缺陷”

ECMAScript 6（ES6）新特性可以分为新增语法（例如：class），增强 JavaScript 功能（例如：import），以及改良 JS“缺陷”（例如：let 关键字）。

最常用的 ES6 特性(上、下)：let, const, class, extends, super, arrow functions, template string, destructuring, default, rest arguments, import export, module。

React 中的 ES6：主要有箭头函数， 类， 模板字符串， let， 和 const 声明。可以使用 Babel REPL 来查看 ES6 的编译结果。

React Router 从第 1.0.3 版开始也使用 ES6 语法。



1. 改良 JS “缺陷”

块级作用域（闭包到块级作用域、let、const）、词法作用域的“this”(通过箭头函数)、处理“arguments”、类与继承（ ES6 之前使用 new 关

键字通过函数（也叫构造器）构造对象当做“类”来使用。）、严格模式（有助于防止问题用法和安全使用 JavaScript。）。

const 有一个很好的应用场景，就是当我们引用第三方库时声明的变量,用 const 来声明可以避免未来不小心重命名而导致出现 bug。

（1）rest 和 default

ES6 引入 rest 参数（形式为...变量名），用于获取函数的多余参数，这样就不需要使用 arguments 对象了。rest 参数搭配的变量是一个数组，该变量将多余的参数放入数组中。

注意：①每个函数最多只能声明一个rest参数，而且 rest参数必须是最后一个参数，否则报错。

②rest参数不能用于对象字面量setter之中

复制代码

```
let object = {
    set name(...value){   //报错
        //执行一些逻辑
    }
}
```

Default：默认值。

（2）箭头函数

超时调用的代码都是在全局作用域中执行的,因此函数中 this 的值在非严格模式下指向 window 对象，在严格模式下是 undefined。

--《javascript 高级程序设计》

注意：箭头函数this指向的固定化，并不是因为箭头函数内部有绑定this的机制，实际原因是箭头函数根本没有自己的this。而箭头函数根本没有自己的this，其内部的this也就是外层代码块的this。这就导致了其：

- 不能用作构造函数
- 不能用call()、apply()、bind()这些方法去改变this的指向

（3） ES5 和 ES6 中实现类

class, extends, super：原型、构造函数，继承（在最新版 React

中出现得很多）。

传统 ECMAScript 没类的概念，它描述了原型链的概念，并将原型链作为实现继承的主要方法。其基本思想是利用原型让一个引用类型继承另一个引用类型的属性和方法。而实现这一行为的传统方法便是通过构造函数。

ES6 提供了更接近传统语言的写法，引入了 Class（类）这个概念，作为对象的模板。通过 class 关键字，可以定义类。

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

Point.prototype.toString = function () {
  return '(' + this.x + ', ' + this.y + ')';
};

var p = new Point(1, 2);
```

```
//定义类
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  toString() {
    return '(' + this.x + ', ' + this.y + ')';
  }
}
var p = new Point(1, 2);
```

- 类里面有一个constructor方法，它是类的默认方法，通过new命令生成对象实例时，自动调用该方法。一个类必须有constructor方法，如果没有显式定义，一个空的constructor方法会被默认添加。
- constructor方法中的this关键字代表实例对象，
- 定义“类”的方法(如上例的toString)的时候，前面不需要加上function这个关键字，直接把函数定义放进去了就可以了。另外，方法之间不需要逗号分隔，加了会报错。
- 使用的时候，也是直接对类使用new命令，跟构造函数的用法完全一致
- 类的所有方法都定义在类的prototype属性上面

class的继承——extend

Class之间可以通过extends关键字实现继承，这比ES5的通过修改原型链实现继承，要清晰和方便很多。

```
class ColorPoint extends Point {
  constructor(x, y, color) {
    super(x, y); // 调用父类的constructor(x, y)
    this.color = color;
  }

  toString() {
    return this.color + ' ' + super.toString(); // 调用父类的toString()
  }
}
```

- super关键字，作为函数调用时（即super(...args)），它代表父类的构造函数；作为对象调用时（即super.prop或super.method()），它代表父类。在这里，它表示父类的构造函数，用来新建父类的this对象。
- 子类必须在constructor方法中调用super方法，否则新建实例时会报错。这是因为子类没有自己的this对象，而是继承父类的this对象，然后对其进行加工。如果不调用super方法，子类就得不到this对象。

（4）模块化

历史上，JavaScript 一直没有模块（module）体系，无法将一个大程序拆分成互相依赖的小文件，再用简单的方法拼装起来，这对开发大型的、复杂的项目形成了巨大障碍。为了适应大型模块的开发，社区制定了一些模块加载方案，比如CMD、AMD(浏览器端)和 CommonJs（服务器端）。

CommonJs是使用在服务器端，是同步加载的，NodeJs是对此规范的实践。

AMD，CMD是使用在浏览器端的，是异步加载的，require.js与sea.js是依赖此规范实现。

ES6 在语言标准的层面上，实现了模块功能，而且实现得相当简单，旨在成为浏览器和服务器**通用**的模块解决方案。

> ES6模块的设计思想，是尽量的静态化，使得编译时就能确定模块的依赖关系，以及输入和输出的变量。CommonJS和AMD模块，都只能在运行时确定这些东西。

ES6 的模块化写法：

```
import { stat, exists, readFile } from 'fs';
```

上面代码的实质是从 fs 模块加载 3 个方法，其他方法不加载。这种加载称为“编译时加载”，即 ES6 可以在编译时就完成模块加载，效率要比 CommonJS 模块（服务器端）的加载方式高。当然，这也导致了没法引用 ES6 模块本身，因为它不是对象。

模块功能：**export**（用于规定模块的对外接口，对外的接口必须与模块内部的变量建立一一对应关系。）、**import**（用于输入其他模块提供的功能，它接受一个对象（用大括号表示），里面指定要从其他模块导入的变量名（也可以使用*号整体加载））。

export default 和 export 区别：

- 1.export与export default均可用于导出常量、函数、文件、模块等
- 2.你可以在其它文件或模块中通过import+(常量 | 函数 | 文件 | 模块)名的方式，将其导入，以便能够对其进行使用
- 3.在一个文件或模块中，export、import可以有多个，export default仅有一个
- 4.通过export方式导出，在导入时要加{ }，export default则不需要

使用 export default 命令，为模块指定默认输出，这样就不需要知道所要加载模块的变量名。

```
1.export
//a.js
export const str = "blablabla~";
export function log(sth) {
  return sth;
}
对应的导入方式：

//b.js
import { str, log } from 'a'; //也可以分开写两次，导入的时候带花括号

2.export default
//a.js
const str = "blablabla~";
export default str;
对应的导入方式：

//b.js
import str from 'a'; //导入的时候没有花括号
```

```
//a.js
let sex = "boy";
export default sex（sex 不能加大括号）
//原本直接 export sex 外部是无法识别的，加上 default 就可以了.但是一个文件内最多只能有一个 export default。
其实此处相当于为 sex 变量值"boy"起了一个系统默认的变量名 default，自然 default 只能有一个值，所以一个文件内不能有多个 export default。
```

```
// b.js
//本质上，X.js 文件的 export default 输出一个叫做 default 的变量，然后系统允许你为它取任意名字。所以可以为 import 的模块起任何变量名，且不需要用大括号包含
import any from "./a.js"
import any12 from "./a.js"
console.log(any,any12)   // boy,boy
```

（5）模板字符串

JavaScript 用 “+” 号连接字符串和变量/表达式。

模板字符串是增强版的字符串，用反引号（`）标识。它可以当作普通字符串使用，也可以用来定义多行字符串，或者在字符串中嵌入

变量。 模板字符串中嵌入变量和函数，需要将变量名写在${}之中。

对于模板字符串，它：

- 使用反引号`包裹；
- 使用${}来输出值；
- ${}里的内容可以是任何 JavaScript 表达式，所以函数调用和算数运算等都是合法的；
- 如果一个值不是字符串，它将被转换为字符串；
- 保留所有的空格、换行和缩进，并输出到结果字符串中（可以书写多行字符串）
- 内部使用反引号和大括号需要转义，转义使用反斜杠 \

```
function sayHello(name){
    return "hello,my name is "+name+" I am "+getAge(18);
}
function getAge(age){
    return age;
}
sayHello("brand") //"hello,my name is brand I am 18"
```

```
function sayHello(name){
    return `hello,my name is ${name} I am ${getAge(18)}`;
}
function getAge(age){
    return age;
}
sayHello("brand") //"hello,my name is brandI am 18"
```

（6）严格模式

严格模式的目标之一是允许更快地调试错误。帮助开发者调试的最佳途径是当确定的问题发生时抛出相应的错误(throw errors when certain patterns occur)，而不是悄无声息地失败或者表现出奇怪的行为(非严格模式下经常发生)。严格模式下的代码会抛出更多的错误信息，能帮助开发者很快注意到一些必须立即解决的问题。在 ES5

中，严格模式是可选项，但是在 ES6 中，许多特性要求必须使用严格模式，这个习惯有助于我们书写更好的 JavaScript。

2.解构

ES6 允许按照一定模式，从数组和对象中提取值，对变量进行赋值，这被称为解构（Destructuring）。将一个数据结构分解为更小部分的过程。ES6 为数组和对象添加解构，省略相似代码。

（1）对象解构

必须用圆括号包裹解构赋值语句，这是因为暴露的花括号会被解析为代码块语句，而块语句不允许在赋值操作符（即等号）左侧出现。圆括号标示了里面的花括号并不是块语句、而应该被解释为表达式，从而允许完成赋值操作。

（2）数组解构

用{}解构返回数组个数：const {length} = names;

数组解构也可以用于赋值上下文，但不需要用小括号包裹表达式。

当使用解构来配合var、let、const来声明变量时，必须提供初始化程序（即等号右边的值）。下面的代码都会因为缺失初始化程序而抛出语法错误：

复制代码

```
var { type, name };//   语法错误!
let { type, name };//   语法错误!
const { type, name };// 语法错误!
```

3.Promise 的基本原理和使用

Promise 是异步编程的一种解决方案，比传统的解决方案(回调函

数和事件)更合理和更强大。

ES6 中的 promise 的出现给我们很好的解决了**回调地狱**的问题，所谓的回调地狱是指当太多的异步步骤需要一步一步执行，或者一个函数里有太多的异步操作，这时候就会产生大量嵌套的回调，使代码嵌套太深而难以阅读和维护。

（1）Promise 原理

Promise，简单说就是一个容器，里面保存着某个未来才会结束的事件（通常是一个异步操作）的结果。从语法上说，Promise 是一个对象，从它可以获取异步操作的消息。Promise 提供统一的 API，各种异步操作都可以用同样的方法进行处理。

promise 会让代码变得更容易维护，像写同步代码一样写异步代码,同时业务逻辑也更易懂。

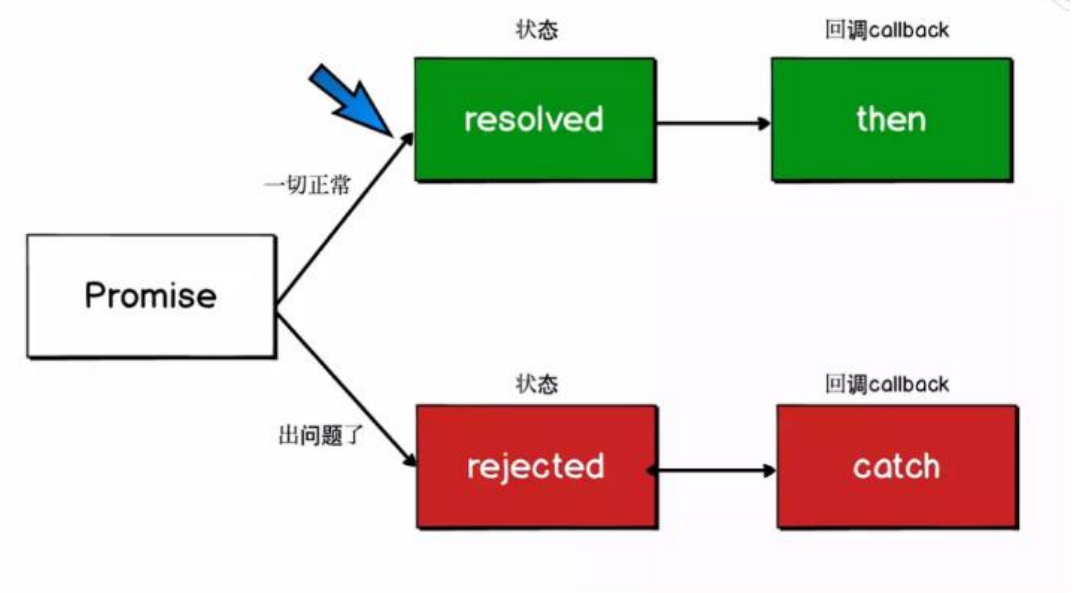
> Promise对象有以下两个特点。
>
> （1）对象的状态不受外界影响。Promise对象代表一个异步操作，有三种状态：pending（进行中）、fulfilled（已成功）和rejected（已失败）。只有异步操作的结果，可以决定当前是哪一种状态，任何其他操作都无法改变这个状态。这也是Promise这个名字的由来，它的英语意思就是“承诺”，表示其他手段无法改变。
>
> （2）一旦状态改变，就不会再变，任何时候都可以得到这个结果。Promise对象的状态改变，只有两种可能：从pending变为fulfilled和从pending变为rejected。只要这两种情况发生，状态就凝固了，不会再变了，会一直保持这个结果，这时就称为resolved（已定型）。如果改变已经发生了，你再对Promise对象添加回调函数，也会立即得到这个结果。这与事件（Event）完全不同，事件的特点是，如果你错过了它，再去监听，是得不到结果的。

有了Promise对象，就可以将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数。此外，Promise对象提供统一的接口，使得控制异步操作更加容易。

Promise也有一些缺点。首先，无法取消Promise，一旦新建它就会立即执行，无法中途取消。其次，如果不设置回调函数，Promise内部抛出的错误，不会反应到外部。第三，当处于pending状态时，无法得知目前进展到哪一个阶段（刚刚开始还是即将完成）。

Promise原理

promise 对象初始化状态为 pending ；当调用resolve(成功)，会由pending => fulfilled ；当调用reject(失败)，会由pending => rejected。具体流程见下图：



（2）Promise 的使用流程

①new Promise 一个实例，而且要 return

②new Promise 时要传入函数，函数有 resolve reject 两个参数

③成功时执行 resolve，失败时执行 reject

④then 监听结果

Promise.all(并行执行异步操作)接受一个 promise 对象的数组，

待全部完成之后，统一执行 success；

Promise.race（赛跑）接受一个包含多个 promise 对象的数组，只要有一个完成，就执行 success。

Node.js 的异步编程方式有效提高了应用性能；然而回调地狱却让人望而生畏，Promise 让我们告别回调函数，写出更优雅的异步代码；在实践过程中，却发现 Promise 并不完美；技术进步是无止境的，这时，我们有了 async/Await。

七、Npm、Node.js、Webpack 与 Express

参考：NPM 使用介绍、用 8 个 npm 技巧打动你的同事、一份关于 npm 的新手指南、10 个 NPM 使用技巧

1. Npm

NPM是随同NodeJS一起安装的包管理工具，能解决NodeJS代码部署上的很多问题，常见的使用场景有以下几种：

- 允许用户从NPM服务器下载别人编写的第三方包到本地使用。
- 允许用户从NPM服务器下载并安装别人编写的命令行程序到本地使用。
- 允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。

可能你已经注意到，运行npm命令有多种方式。以下是一些常用的npm别名的简要列表：

- `npm i <package>` - 安装本地包
- `npm i -g <package>` - 安装全局包
- `npm un <package>` - 卸载本地包
- `npm up` - npm更新包
- `npm t` - 运行测试
- `npm ls` - 列出已安装的模块
- `npm ll` 或 `npm la` - 在列出模块时打印附加包信息

安装淘宝 npm（cnpm）

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

2. Webpack

参考：webpack 入门、Webpack 入门教程、浅入浅出 webpack、Webpack 从入门到上线

webpack 是模块化管理工具，使用 webpack 可以对模块进行压缩、预处理、按需打包、按需加载等。

3. Express

参考：Express 使用手记：核心入门、Express - 简单介绍 Express

Express 是 Node.js 上最流行的 Web 开发框架，使用它我们可以快速的开发一个 Web 应用。

八、周末及下周学习计划

1. 复习消化、强化、补充完善以上内容
2. 深入 ES6，Promise、Fetch API、跨域、事件委托代理
3. react + redux + router（结合模块管理工具，webpack，fie）

2018.8.10（完）

下周部分内容（详见掘金 爱·学习 文件夹）：

Fetch API 初探：

http://coderlt.coding.me/2016/11/20/JS-Feach/

从 ajax 到 fetch、axios：

https://juejin.im/post/5acde23c5188255cb32e7e76

传统 Ajax 已死，Fetch 永生：

https://segmentfault.com/a/1190000003810652

九、react + redux + router 全家桶

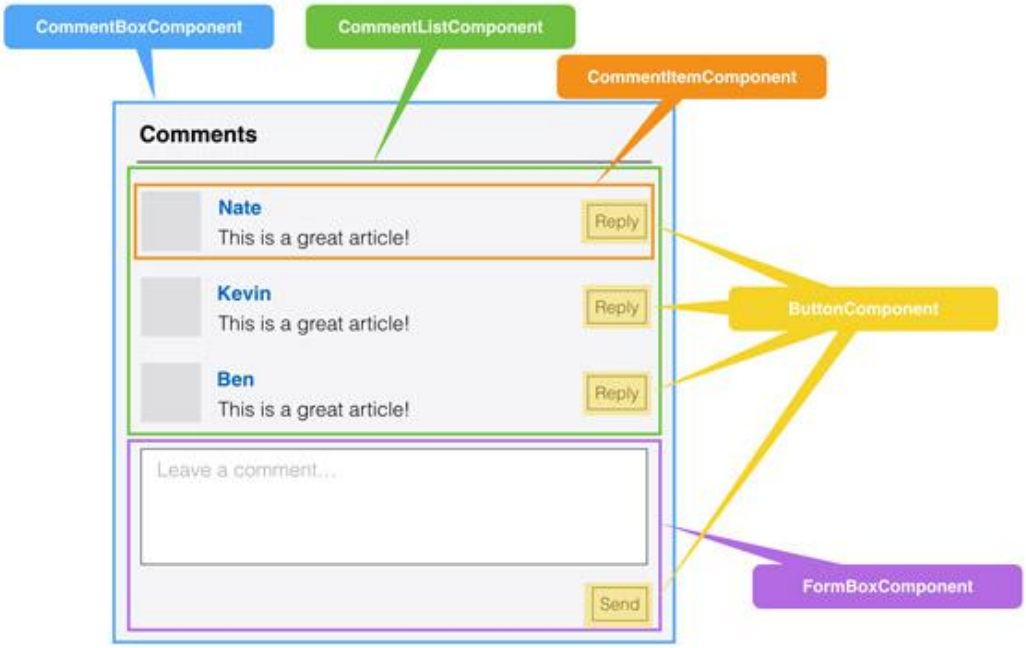
1. React

（1）什么是 react？

React 起源于 Facebook 的内部项目，由于设计思想极其独特，属于革命性创新，性能出众，代码逻辑却非常简单。所以，越来越多的人开始关注和使用，认为它可能是将来 Web 开发的主流工具。

React 不是一个完整的 MVC 框架，最多可以认为是 MVC 中的 V（View），甚至 React 并不非常认可 MVC 开发模式。

> 对于MVC开发模式来说，开发者将三者定义成不同的类，实现了表现，数据，控制的分离。开发者更多的是从技术的角度来对UI进行拆分，实现松耦合。
>
> 对于React而言，则完全是一个新的思路，开发者从功能的角度出发，将UI分成不同的组件，每个组件都独立封装。
>
> 在React中，你按照界面模块自然划分的方式来组织和编写你的代码，对于评论界面而言，整个UI是一个通过小组件构成的大组件，每个组件只关心自己部分的逻辑，彼此独立。



特点：

React 引入**虚拟 DOM**(Virtual DOM)机制，在浏览器端用 Javascript 实现了一套 DOM API，解决复杂或频繁 DOM 操作带来的性能瓶颈问题。

组件化：虚拟 DOM(virtual-dom)不仅带来了简单的 UI 开发逻辑，同时也带来了组件化开发的思想，所谓组件，即封装起来的具有独立功能的 UI 部件。组件具有**可组合**、**可重用**、**可维护**的特征。

React 不依赖 jQuery，当然我们可以使用 jQuery，但是 render 里面第二个参数必须使用 JavaScript 原生的 getElementByID 方法，不能使用 jQuery 来选取 DOM 节点。

React.js 基本环境安装:点这里。

参考：开发 react 应用最好用的脚手架 create-react-app

create-react-app 的使用及原理分析

create-react-app:facebook 官方开发。所有的源码放到 src 目录下，什么配置文件，各种乱七八糟都不用管，只需要专注开发就好

了，create-react-app 都给你处理好了。开发环境也有类似 webpack-dev-server 的--inline --hot 自动刷新的功能。

线上编译命令：这个是 create-react-app 的一个大亮点，它能让你的应用骗译出在线上生产环境运行的代码，编译出来的文件很小，且文件名还带 hash 值，方便我们做 cache，而且它还提供一个服务器，让我们在本地也能看到线上生产环境类似的效果。

只需一行命令：

```
npm run build
```

运行下面两条命令，可以查看线上生产环境的运行效果。

```
1 npm install -g pushstate-server
2 pushstate-server build
```

编译好的文件都会放到 build 目录中。

（2）组件间通信

（3）重要属性

props 是组件对外的接口，state 是组件对内的接口。

Props：属性传值；组件通信；不会进行改变。

State：

不是用来传值，用于改变组件内状态值，来自{Component}；和 render 方法一样，只能用于 class 内部；

不能直接修改 State:直接修改 state，组件并不会重新重发 render。正确的修改方式是使用 setState()；

State 的更新是异步的；

State 的更新是一个浅合并（Shallow Merge）的过程。当调用 setState 修改组件状态时，只需要传入发生改变的状态变量，而不是组件完整的 state，因为组件 state 的更新是一个浅合并（Shallow Merge）的过程；

函数传参：

箭头函数()=>XX(xxx)，不会自动执行；.bind（this，“xxx”）。

```
state={
}
render(){
    this.state.
}
```

（4）

2. Redux

3. Router

4.

Your Timeline for Learning React:

https://aleen42.github.io/PersonalWiki/translation/your_timeline_for_learning_react/your_timeline_for_learning_react.html

一看就懂的 ReactJs 入门教程（精华版）：

http://www.cocoachina.com/webapp/20150721/12692.html

React 入门实例教程：http://www.ruanyifeng.com/blog/2015/03/react.html

React 技术栈系列教程：

http://www.ruanyifeng.com/blog/2016/09/react-technology-stack.html

全面理解虚拟 DOM，实现虚拟 DOM：http://foio.github.io/virtual-dom/

爱用前端实习生资料索引（必看）：

https://aiyongbao.yuque.com/allhands/mb50kc/np1w61

Stackoverflow：https://stackoverflow.com/users/10207206/king-hcj