高效前端：从前端菜鸟到大V的成长经验分享

首先这个标题有点想骗你点进来的意思，起得有点浮夸，我主要还是介绍我的成长经历和一些经验。我从事前端快三年，从一个刚入门前端的小伙到现在出了一本前端的书的大伙，这个过程我觉得有一些比较好的东西分享一下，说不定对你有帮助。

有人说要多看别人失败的经验，少看别人成功的经验，因为每个人的先天条件和环境是不一样的，成功往往是不可复制的。所以在接受鸡汤的同时要保持警惕，我说的这些都只是参考，不一定适合你。但是我肯定觉得有用才会列出来，否则不会随便拿出来误导大众。

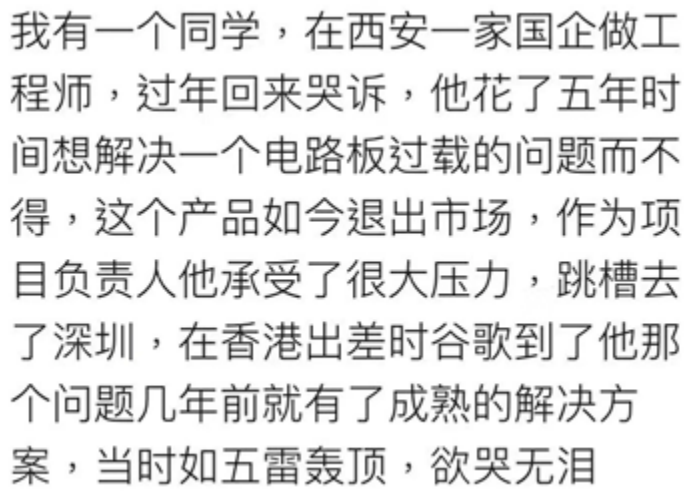
我先介绍下自己，我在学校里面主要编程语言是 C++，后来到了人人做 Java Web 服务，做了半年又觉得前端比较好玩，所以转了前端，前端还挺适合我。发博客已经有两年半，在知乎、掘金等地累计发了 50 多篇文章，有很多篇有上过前端热门、被很多公众号转发。现在还出了一本书叫《高效前端》，这本书在京东获得一致好评，上市一个多月，现在准备要重新印刷了。

为什么要写博客呢？这是因为读大学的时候有个师姐推荐我去关注一个叫刘未鹏（这个才是真大 V 哈）的博客，他上面发表了很多很有思考深度的文章，其中有一篇《为什么你应该（从现在开始就）写博客》看了之后感触很深，发现原来还有写博客这样的操作，不过在接下来的两三年也都没付诸行动，但心里一直有这个想法，直到转了前端之后才开始写博客（在 2015 年的国庆），而且前端本身开源的性质很大，能写的东西特别多。

通过写博客让我成长了不少，除了写博客之外还有其它一些经验，请上车。

1. 使用谷歌搜索

最近刚好在朋友圈看到一张图：



大意是说有位老兄遇到个电路板过载的问题搞了五年不得解，后来这个产品退出市场了，几年后他去了香港用谷歌搜了一下，发现那个问题当时就有了成熟的解决方案了。这个看起来有点夸张，但是还是有一定可信度的，因为谷歌搜索真的可以解决很多问题。做为一名工程师，如果还停留在只会百度的层面，那就真的是。。。很乖巧。

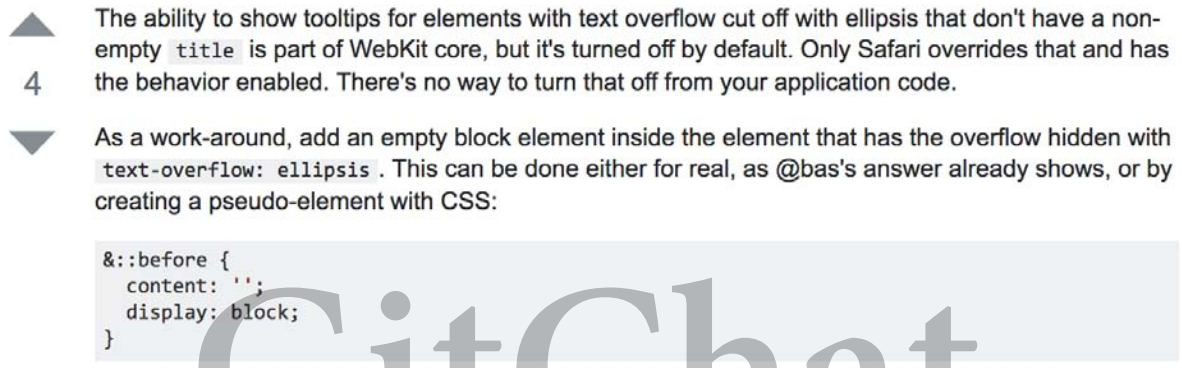
因为程序员是为了解决问题而生的，很多外行人觉得写代码的人黑科技很了不起，但实际上他们却不知，其实很多人是一边写代码一边各种查，查问题、查 API。我自己在写代码的时候就遇到很多问题，某个 API 不太熟，某个效果比较复杂不太好做，出了一个奇怪的 bug，理论上是没问题的，实际上跑不通，等等。这个时候你可能会去请教别人，但通常我们会先自己尝试解决，搞不定再去问别人。所以会用搜索引擎就很重要了。

刚开始的时候谷歌还是可以访问. hk 域名的，但是不稳定，后来连. hk 都访问不了，我也渐渐忘了还有谷歌这个东西了，所以用百度比较多，但用百度的时候总是有一些问题搜不到答案，搜来搜去很多网站的内容都是累同的。我在大三第二学期的时候去了深圳一家公司实习，那家公司很有外企的氛围，在那里学到了一项很重要的技能：科学上网（fq）。用了稳定的谷歌之后发现各种清爽，很多问题都可以很快地找到解决方法和相关代码。下面我举一些例子。

我们经常使用 text-overflow: ellipsis 来做单行文本… 的省略隐藏，这个在 Safari 上当鼠标悬浮的时候，它会显示一个 title 的提示文案，如下图所示：

如果 UI 想给它加一个 hover-tip 的话就会有双重提示，一个自己加的，另一个是浏览器加的，这个问题只有 Safari 会，其它浏览器不会。那怎么去掉浏览器加的提示文案，只要在谷歌搜索：safari text-overflow ellipsis hide title，打开第一条搜索结果：

就可以找到解决方法，有位老兄解释了 Safari 的问题，并给出了类似于 hack 的解决方法：



就是加一个块级的伪类，试了一下，果然可以。（下面评论有人指出了要用 after 兼容性更好）。你可以尝试用百度搜中文看能不能搜到相关的。比较庆幸的是百度搜英文能够搜到类似于 stackoverflow 高质量社区的文章了，但是排位比较靠后。而且如果你不会科学上网的话，很多外文的网站是打不开的，因为它们很多用了谷歌的 CDN。

另外还可以搜工具类，如 web 字体转换工具，找一个能够把桌面字体转成 ttf/woff2 的工具，因为 woff2 比 ttf 等要小一半的体积，用谷歌搜索 web font generator 就可以轻松找到。

还可以搜索效果类的，如做一个闪光的动画、星星的动画等等，这种经常会搜到 codepen/js fiddel，代码直接就可以拿过来用了。这里有一些小技巧：

1. 使用合适的关键词，不可否认，英文社区的文档质量要比中文的高很多
2. 谷歌关键词的提示，如输入web font会出来web font generator的下拉
3. 如果搜不到结果，切换关键词，如css blink不对的话换成css shine
4. 问题类优先查看Stackoverflow，直接看回答，不用看问题.
5. API类的如css的filter属性优先看MDN或W3school（英文版）
6. CSS Trick/Code pen/Github issue 也是一些高质量社区

可能你会觉得自己英文不行，看不懂英文的，实际上这些东西不需要多高深的英文，你只要直接看代码看 demo 看一下是不是你需要的就行，如果不确定就套过来试一试，一试便知，然后再细细去看。不用重头到尾把每个英文单词都细细看一遍，把每句话都理解了。（当然要是有时候要是找了好久没找到，确实需要去细看，看他说的话对你是否有启发）

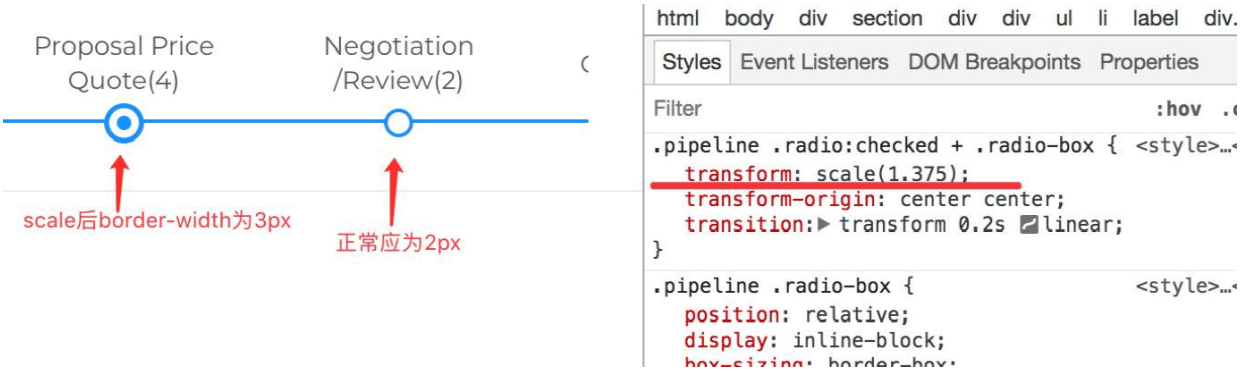
这种东西看多了也是可以培养英文能力了，有空可以找一本英文原著的书看一下，如 *Javascript: The Good Parts*（请问这是哪本书呢？^^），网上有 pdf 的，看完一两本之后基本英文阅读能力就没什么问题了。

2. 多尝试不同的解决方案

不要反复使用同一套经验，有些人习惯用了某一种方式之后，就一直用同一种解决方式，反正能完成需求就好了，工期又那么紧，先解决了再说。但是你要是总是抱着这种想法的话，可能就不会有成长了，就真的是一套经验用三年，变成了一个“工厂”里面的熟练工。

例如说有些人布局喜欢用 inline-block，动不动就加个 inline-block 解决换行问题，还有人喜欢用 table，另外一些人喜欢用 flex，什么地方都用 flex。这里并不说用这些属性有问题，只是你一直保持用同一种就会有问题。这些布局包括 float 都有它的适用范围，只有在这几种布局灵活切换，那才是真的高手。

我再举一个具体的例子，怎么做一个圆环放大的动画，当鼠标悬浮的时候，需要做一个放大的动画，这个可以用 transform: scale 的方式，但是遇到一个问题：



就是 scale 之后，会导致 border 变粗了，看起来有点丑，应该是要保持 border-width 为 2px 的。如果改成用 width/height 的话会导致圆圈的动画非常不圆滑。所以常规的方法解决不了这个问题，这个时候你可能会找 UI / 产品看能不能妥协一下。但实际上这个问题是可以解决的，只要你换一种思维方式：用 SVG，很多人对 SVG 都很陌生，可能不会尝试。但其实我们可以试一下，结果 SVG 可能完美地做出这个动画，如下图所示：

```
<svg width="22px" height="22px">
    <circle r="8" cx="11" cy="11" fill="#fff" stroke="#2492fc">
        <animate
            attributeName="r"
            from="8"
            to="10"
            dur="0.3s"
            begin="mouseover"
            fill="freeze"
        />
    </circle>
</svg>
```

做半径r的动画

r从8px变到10px

动画时间为0.3s

hover的时候触发动画

动画结束时冻住

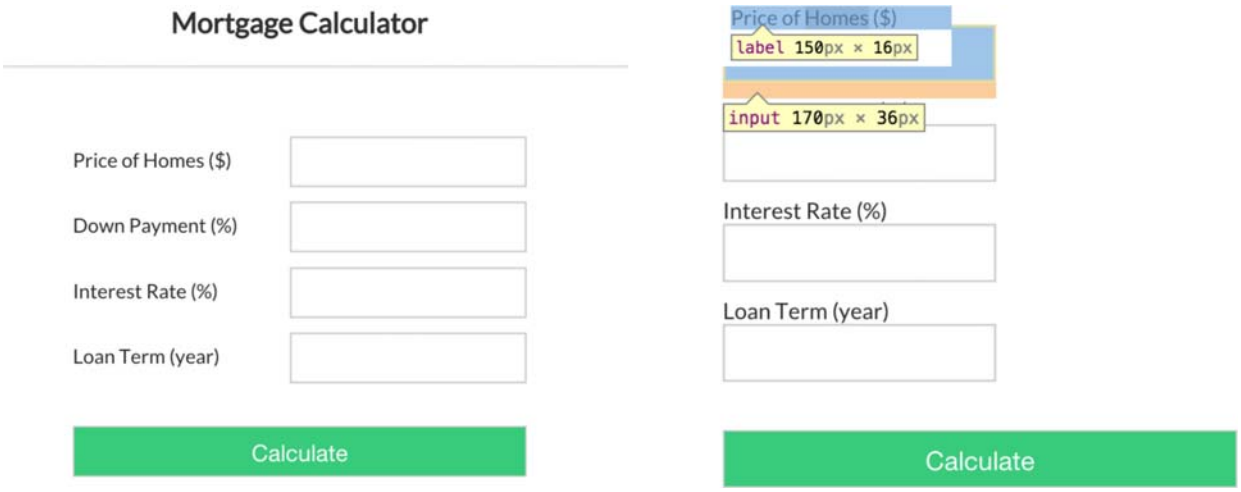
这样你就接触到了 SVG 动画，进而可以继续挖掘 SVG 可以做其它哪些 CSS 不好做的动画。以后你的技能列表里面做动画就多了一项 SVG。

做不同的项目或者不同的需求，你会遇到不同的场景，但是不要觉得需求小或者没有挑战，就是在写重复代码。当你尝试用不一样的方法去解决不同场景的问题的时候，你会发现在这里面可以学到很多东西，你可能不会遇到圆圈放大的动画的问题，但你可能会遇到其它的问题。我做了快三年前端，很大的成长利益于这种不断尝试新方案，或者发现老方案的一些问题做一些改进。

3. 遇到问题深挖到底

有些问题是很有价值，当你遇到一个问题的时候，不要轻易用一些治标的方法把它解决了，不知道为什么会这样。

例如很多人会遇到这样的问题：左边的 label 为 150px，右边的 input 为 170px，而它们的容器为 320px，但是这样的话会导致 input 换行了，如下图所示，难道 150px + 170px != 320px?



这个时候你可能会想到把容器调大一点，例如调到 330px，惊奇地发现解决问题了，input 不会换行了，搞定，很开心！但是有没有想过这是为什么呢?

又如当你把排成一行的几个行内元素设置 margin-left 为 UI 的标注，但是会发现会比 UI 大了一点，这又是为什么呢？这个时候你可能会手动把间距调小点让它看起来和 UI 一样。但是每次你都这样解决问题的话，就真的学不到东西，当然你可能还会给个冠冕堂皇的理由：需求排期紧，没有办法呀，先解决了以后再说吧。（但是通常以后就没有以后了，要么现在解决，要么没有下一次了）

实际上这两个问题如果你深入挖掘的话会发现是因为两个行内标签中间的换行符引入了空格引起的，多余的空白是因为中间多渲染了一个空白字符。那么怎么解决这个问题呢，有几种解决方式？就可以继续去思考。

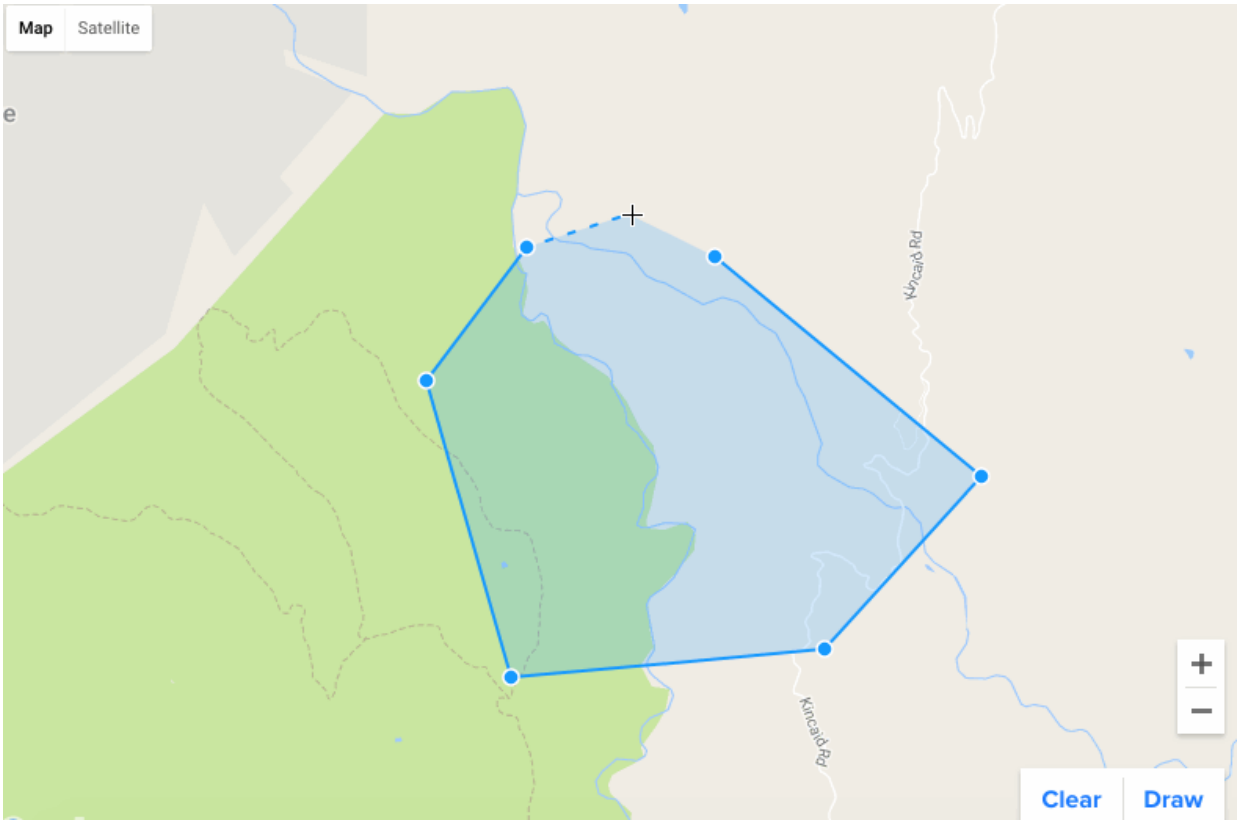
又如很多人会遇到图片底部空白的问题：

这个是因为父容器行高引起的，然后可以继续延伸下去，例如字号和行高的关系，为什么 height 和 line-height 一样可以垂直居中等等。

以上提到的 3 点我觉得是最为重要的，下面再介绍一些其它的。

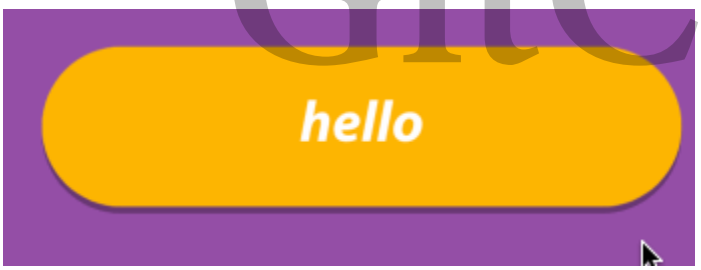
4. 追求细节

有些人抱着做做需求，过过日子的态度，把需求做了就好了。实际上我们可以给自己提高点要求，需求和项目不仅要做了，而且要做好了。例如我之前在做一个地图绘制应用的时候，做了一个吸附效果：



就是在最后一个点闭合的时候，当靠近最后一个点的时候有一个吸附效果。这个东西产品并没有要求，竞品也没有做这种效果，是我自己加上去的，最后效果还是挺好的。

再如做一个拖拽应用的时候，在拖住的时候手会变一个形状：



这个效果是我自己找 UI 加的，之前就给了一只手，但其实要是不变成另一只手的话看起来会有点奇怪，最后面他们发现这个问题可能也会找你做一下，就会显得比较被动。

这些效果可以起到画龙点睛的作用，而不是画蛇添足哈~

5. 代码洁癖

代码洁癖是说对自己的代码有高要求，首先在排版风格要一致，例如下面的代码，给人感觉就很乱：

```
if (a  === b){
     return true;
     }
```

你可能会说，没关系啊，反正我的编辑器会帮我格式化。但你总不能每次都依赖编辑器吧。你去面试代码写得很乱，给人第一印象就不好了吧。自己还是要养成一个习惯。

还有这种，代码拷来拷去的：

```
let full = first.replace(/ \([^)]+\)$/, "") + " "
        + last.replace(/ \([^)]+\)$/, "");
```

我们可以把相同的部分提取了出来：

```
let reg = / \([^)]+\)$/,
    full = first.replace(reg, "") + " "

                + last.replace(reg, "");
```

这样看起来好看一点。

6. 避免大段拷贝代码

我发现很多人喜欢拷代码，这个功能很像，把那段代码拷过来改一改，那里又要用到，又把十几行甚至几十行代码拷了过去。但这个是不可取的，如果你还停留在这个阶段，还是比较低级的。

应该要把相同点与不同点分离出来，把相同点当作一个函数或者一个类，不同点当作函数的传参或者类的实例数据，整体思路是这样的：重复代码 -> 函数 -> 模块 -> 插件，抽象级别不断提高，这个我在《高效前端》这本书里也有提及。

7. 回归技术基础

不要忽略了最基础的技术，例如在一个 div 里面水平居中两个 button：

我们可以用 flex 居中，也可以用 translate 负值法居中，但其实不用这么麻烦，只要给 div 加一个 text-align: middle 就行了，因为 button 是行内元素，text-align 是管用的，button 和 button 之间的间距可以直接用 margin-left 即可。

又如怎么垂直居中两个图片加一个文字：

这个时候你也可以用 flex，设置 align-item: center，但其实不用使用 CSS3 也可以实现，直接用 vertical-align: middle 就可以了。就会有上图的效果。

8. 不断学习新技术

接触 HTML5 的一些新技术，例如：

还有 CSS 变量 / CSS Houdini 等。
这些东西其实不算新了，有些已经发展多年。但其实很多人没怎么碰过，因为项目里面没用到。建议拿点学习框架的时间和热情来学点 HTML5 原生的技术。

9. 计算机基础知识的积累

包括不限于：

1. TCP/HTTP

2. HTTPS
3. 数据结构和算法
4. SQL
5. 多线程
6. 浏览器渲染 / CSS 布局

框架容易过时，但是计算机基础的东西不容易过时，多学一点不容易过时的东西，它们就像你的地基一样。要当一名优秀的前端前先当一名优秀的程序员。我在《高效前端》这本书里介绍了大量的基础知识，并和实践相结合，以达到修炼内功的目的。

10. 别总是重复造轮子

当要实现一个比较复杂的功能的话，我更偏向于先在网上找有没有现成的插件可以用，然后看它怎么实现的，拿过来改一改，适应产品的特点

从头造一个轮子确实可以学到很多东西，但是你去研究别人的代码其实是另一种级别的学习

11. 不要认为前端就是写 html/css/js

只会 Java 的 Java 程序员不是好的 Java 程序员，只会前端的前端不是好的前端。如果你认为前端就是写 html/css/js，那你的眼界就有点狭窄，自我设限。你可以接触点 nginx 的东西，数据库的东西，Java Web 的东西，你要知道后端是怎么建表怎么查表然后返回数据给你的。这样当你遇到问题的时候就会从前后端的思维思考这个问题，例如为什么我传的参数不对就会导致后端接口 500 了，这个时候你就可以根据后端返回的信息定位原因，例如可能是后端写数据库的时候数据长度超过了字段的最大长度值，导致数据库抛异常。所以除了 JS 之外，最好再会一门 Java/C++ 之类的强类型语言。你就会知道哪些特性是 JS 专有的，哪些是语言的共性，哪些是 JS 糟粕，哪些是 JS 的精华，这种前后端的思维对我的成长也是很有帮助。需要保持比较宽的技术视野。

12. 写博客、写文档

另外一个对我的成长起了很大作用的就是写博客了。程序员最烦两件事情，一件是别人的代码没有文档，第二件事是给自己的代码写文档。

养成写作的习惯是一个很好的事情，可以给代码写文档，写一些项目总结，遇到的问题，是怎么解决的，有没有其它的解决方案，如果你发现这些东西有共性，很多人也遇到过这种问题可以写成一篇博客，或者是自己主动研究的东西。写博客有很多好处，开篇我就提到了为什么要开始写博客。从我自己的感触来说的话：

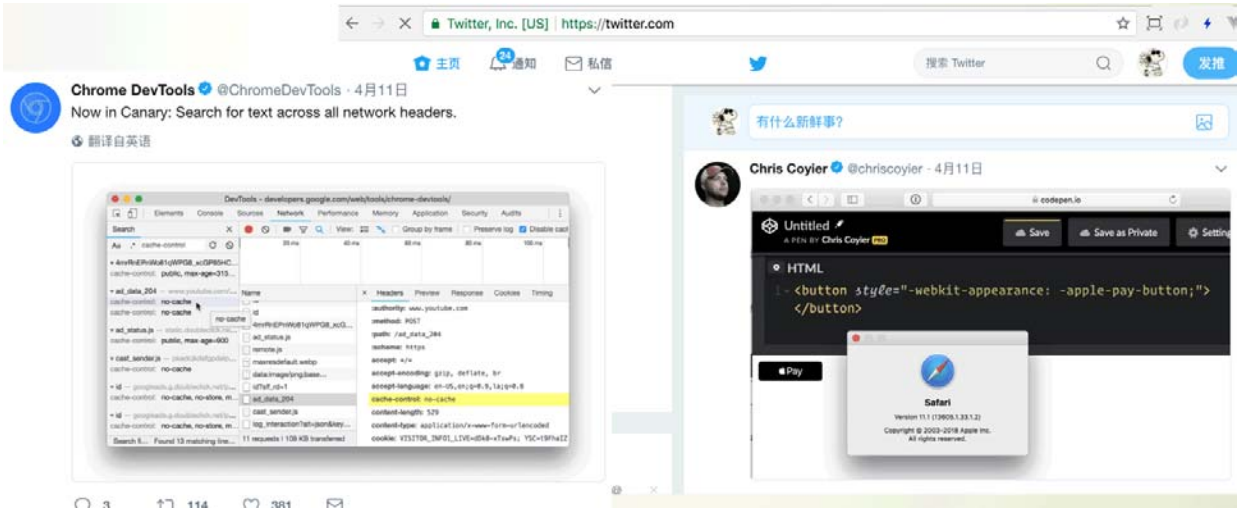
1. 有些东西你认为掌握了，但是当你写的时候，你发现说不明白，不透彻
2. 往往会一边写一边查一边实践，从而发现很多相关的自己还没有了解的东西
3. 在写的时候会把很多零碎的东西联系在一起，渐渐形成自己的知识体系
4. 培养钻研技术和写文章的能力，提高逻辑思维能力

另外就是分享的乐趣，和网友讨论知识，以及结识圈内的一些人：

不要害怕写得不好，其实我的文笔很差，高考 60 分作文总是写不出来凑 800 字的那种，但是现在通过写博客，我发现我的思维越来越清晰，写东西越来越流畅了。

13. 关注社区

中文的社区有掘金、知乎、前端的公众号（早读课、大全），英文的社区有英文的有 js/css weekly 邮件订阅、sitepoint、stackoverflow，Twitter 上面也有一些高质量的号，例如：



左边是 Chrome Devtools 介绍新的搜索 HTTP 头部字段的功能，右边是有个老兄在介绍 Safari 的 Apple Pay 按钮。

只有关注这些社区才能让自己保持在技术的前沿。

总的来说，也没什么经验，如果你对攻城狮这一行比较感兴趣的话自然会想办法提升自己。如果你需要一本全面提升软实力的书，那么《高效前端》很适合你，里面有很多案例和基础，凝聚了我 3 年前端的智慧。