# 前端开发指导书

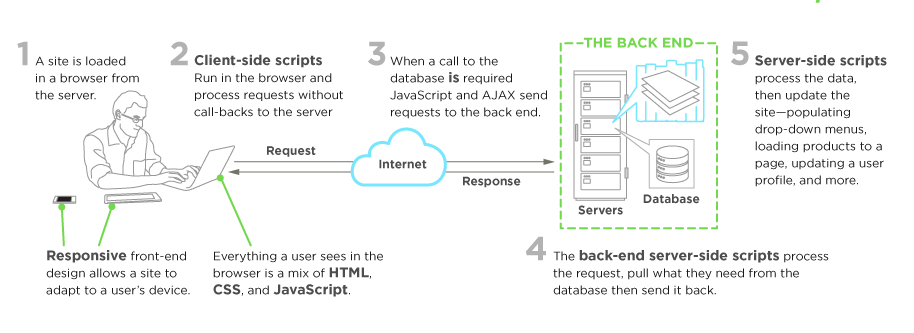
# 前端开发是什么？

WEB 前端开发，也作客户端开发，是给网站或网页应用生产 HTML、CSS 和 JavaScript 的过程，它让用户得以浏览网站或网页应用并与之交互。创建网站前端的工具与技术时常变革，因此开发者应对业内发展近况保持清醒，这是前端开发者面对的挑战。

保证用户打开网站时，以一种易读且关联的形式浏览信息，是设计网站的目的。目前这个需求变得更棘手：如今用户使用的设备种类繁多，设备对应的屏幕尺寸和方案亦然，这迫使设计者在开发站点时考虑这些方面。他们要保证站点在不同的浏览器（跨浏览器），不同的操作系统（跨平台）以及不同的设备（跨设备）上正确运行，这要求开发者缜密地制定方案。

## HTML、CSS 和 JavaScript:

前端开发者使用 WEB 技术（例如 HTML、CSS、 DOM 和 JavaScript）来建造网站和开发应用。他们使用 WEB 技术在 [WEB 平台](https://en.wikipedia.org/wiki/Open_Web_Platform)或非 WEB 框架（比如 [NativeScript 框架](https://www.nativescript.org/)）上设计程序。



通常前端开发者通过学习 HTML、CSS 和 JS 来入门。HTML、CSS 和 JS 代码在 [WEB 浏览器](https://en.wikipedia.org/wiki/Web_browser)、[无头浏览器（Headless Browsers）](https://en.wikipedia.org/wiki/Headless_browser)、[WebView](http://developer.telerik.com/featured/what-is-a-webview/) 上运行，或者被用于原生运行环境的程序设计。我稍后解释这四种运行场景。

### WEB 浏览器

WEB 浏览器用于从[万维网（WWW.）](https://en.wikipedia.org/wiki/World_Wide_Web)检索、呈现和遍历信息。通常浏览器在台式机、笔记本、平板或手机上运行，如今，浏览器可以在几乎所有物体（比如冰箱，汽车）上运行。

最常见的 WEB 浏览器如下（按使用度排序）：

* [Chrome](http://www.google.com/chrome/)
* [IE](https://en.wikipedia.org/wiki/Internet_Explorer)（注：非 [Edge](https://en.wikipedia.org/wiki/Internet_Explorer)，数据参考自 IE 9 至 IE 11）
* [火狐](https://www.mozilla.org/firefox/)
* [Safari](http://www.apple.com/safari/)

### 无头浏览器（Headless Browsers）

无头浏览器是一种没有用户图形界面的 WEB 浏览器，我们可以用命令行界面编程控制该浏览器，达到自动化运行 Web 页面（比如功能测试、网站检索、单元测试等）的目的。可将无头浏览器看作网页浏览器，不过你用命令行来检索、遍历网页。

最常见的无头浏览器：

* [PhantomJS](http://phantomjs.org/)
* [slimerjs](http://slimerjs.org/)
* [trifleJS](http://triflejs.org/)

### Webviews

原生操作系统的原生应用中，用 [Webviews](http://developer.telerik.com/featured/what-is-a-webview/) 来运行网页。不妨把 [webview](http://developer.telerik.com/featured/what-is-a-webview/) 想成一个嵌进原生应用的 iframe 或 WEB 浏览器标签，而该原生应用运行在设备的系统上（比如 IOS、安卓、windows）。

最常见的 webview 开发解决方案如下：

* [Cordova](https://cordova.apache.org/) (通常用于手机、平板的原生应用)
* [NW.js](https://github.com/nwjs/nw.js) (通常用于桌面应用)
* [Electron](https://electron.atom.io/) (通常用于桌面应用)

### 基于 WEB 技术的原生应用

最终，前端开发者从 WEB 浏览器开发中得到经验，并可以脱离浏览器引擎环境编码。近来，人们正在构思如何脱离 web 引擎，用 web 技术（比如 CSS 和 JS）来构建原生应用。

该环境的例子：

* [NativeScript](https://www.nativescript.org/)
* [React Native](https://facebook.github.io/react-native/)

## 注：

请确认自己明白 “web platform” 的准确含义。查阅[“The Web platform: what it is”](http://tess.oconnor.cx/2009/05/what-the-web-platform-is)和维基百科 [“Open Web Platform”](https://en.wikipedia.org/wiki/Open_Web_Platform)

# 前端开发者所使用的技术



前端开发者所使用的核心技术有如下这些（建议按顺序进行学习）：

1. 统一资源定位符 (URLs)
2. 超文本传输协议 (HTTP)
3. 超文本标记语言 (HTML)
4. 层叠样式表 (CSS)
5. JavaScript 编程语言 (ECMAScript 262)
6. JavaScript 对象表示法 (JSON)
7. 文档对象模型(DOM)
8. 网络 APIs (HTML5 或者浏览器 APIs)
9. 网络内容可达性指南 (WCAG) & 可访问的富互联网应用 (ARIA)

下面的介绍涵盖了以上技术的定义，相关文档以及具体规范。至于更加详尽的网络开发规范请参看 [platform.html5.org](https://platform.html5.org/)。

##### 超文本标记语言 (HTML)

超文本标记语言，通常被称为 HTML，被用作创建网页的标准标记语言。网络浏览器可以读取 HTML 文件并且把它们渲染成可见或可听的网页。HTML 在语义上描述了一个网站的结构，并且隐含了其表现形式，因此是一种标记语言，而非程序语言。

— [*维基百科*](https://en.wikipedia.org/wiki/HTML)

相关规范／文档：

* [W3C HTML 规范大全](http://www.w3.org/standards/techs/html#w3c_all)
* [动态标准中的 HTML 元素](https://html.spec.whatwg.org/multipage)
* [全局属性](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes)
* [W3C 的 HTML 5.2](http://w3c.github.io/html/)
* [HTML 属性参考资料](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes)
* [HTML 元素参考资料](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)
* [HTML 语法](https://html.spec.whatwg.org/multipage/syntax.html#syntax) from the Living Standard

##### 层叠样式表 (CSS)

层叠样式表（CSS）是一种样式语言，用来描述使用标记语言编写的文档的外观和格式。尽管样式表通常被用来改变以 HTML 和 XHTML 的方式编写的网页和用户界面的样式，它也能被运用在任何使用 XML 编写的文档中，其中包括 XML，SVG 和 XUL。同 HTML 和 Javascript 一样，CSS 是用以构建具有视觉冲击力的网页和用户界面的基础技术。

— [*维基百科*](https://en.wikipedia.org/wiki/Cascading_Style_Sheets)

相关规范／文档：

* [W3C CSS 规范大全](http://www.w3.org/Style/CSS/current-work#roadmap)
* [层叠样式表 2.2 (CSS 2.2) 规范](https://drafts.csswg.org/css2/)
* [CSS 参考资料](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)
* [第三代选择器](http://www.w3.org/TR/css3-selectors/)

##### 文档对象模型 (DOM)

文档对象模型（DOM）是一个跨平台并且具有语言无关性的概念，用来表示 HTML，XTHML 以及 XML 文档中的对象以及这类对象的交互方式。每份文档中按照树形结构进行组织的节点，被称为 DOM 树。可以使用 DOM 树中的对象所拥有的方法对该对象进行处理和操作。DOM 的 API 规定了它的公共接口。

— [*维基百科*](https://en.wikipedia.org/wiki/Document_Object_Model)

相关规范／文档：

* [文档对象模型 (DOM) 的第三代事件规范](https://www.w3.org/TR/DOM-Level-3-Events/)
* [DOM 动态标准](https://dom.spec.whatwg.org/)
* [W3C DOM4](https://www.w3.org/TR/2015/REC-dom-20151119/)

##### JavaScript 编程语言 (ECMAScript 262)

JavaScript 是一个高级的、动态的、弱类型的解释性编程语言，被包含在 ECMAScript 的语言规范中。同 HTML 和 CSS 一样，它是万维网内容生产环节必不可少的三种技术之一，被大多数网站所使用，并且在不需要使用插件的情况下被所有现代的浏览器所支持。Javascript 基于原型并且把函数视为头等公民，因此是一种多范式的编程语言，支持面向对象，命令式以及函数式编程风格。它有一个可被用来操作文字，数组，日期以及正则表达式的 API，然而并不包含任何 I/O，因此像建网，存储或者图形工具之类的功能就需要依赖它所在的开发环境。

— [*维基百科*](https://en.wikipedia.org/wiki/JavaScript)

相关规范／文档：

* [ECMAScript® 2017 语言规范](https://tc39.github.io/ecma262/)

##### Web APIs (HTML5 及其他)

当使用 Javascript 给网页编写代码的时候，有许多的 API 可供使用。以下是一张关于所有在开发网站或者网络应用时可以使用的公共接口的表。

— [*Mozilla*](https://developer.mozilla.org/en-US/docs/Web/API)

相关文档：

* [Web API 公共接口](https://developer.mozilla.org/en-US/docs/Web/API)

##### 超文本传输协议 (HTTP)

超文本传输协议（HTTP）是一个为分布式的、协作的、多媒体的信息系统指定的应用协议。HTTP 是万维网数据交流的基础。

— [*维基百科*](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

相关规范：

* [超文本传输协议 -- HTTP/1.1](https://tools.ietf.org/html/rfc2616)
* [HTTP/2](https://http2.github.io/)

##### 统一资源定位符 (URL)

统一资源定位符（URL）也被称为网址，是关于资源的引用，明确了计算机网络资源的地址以及检索该资源的机制。URL 是统一资源标志符（URI）的一种特定类型，尽管许多人认为这两个概念可以互换。URL 表明了获取所需资源的方式，然而并非每个 URI 都会如此。URLs 通常被用于网页（http），与此同时也被用作文件传输（ftp），邮件（mailto），数据库接入（JDBC），以及许多其他的应用。

— [*维基百科*](https://en.wikipedia.org/wiki/Uniform_Resource_Locator)

相关规范：

* [统一资源定位符 (URL)](http://www.w3.org/Addressing/URL/url-spec.txt)
* [URL 动态标准](https://url.spec.whatwg.org/)

##### JavaScript 对象表示法 (JSON)

Javascript 对象表示法是在异步的浏览器／服务器交流方式（AJAJ）中所使用的主流的数据格式，并且基本上已经取代 XML（被 AJAX 所使用）。虽然 JSON 是从 Javascript 中衍生出的数据格式，但它实际上是独立于语言的。在许多的编程语言中都有现成的，用于解析和生成 JSON 数据的代码。JSON 数据格式的规范最开始是由 Douglas Crockford 制定的，现在被两个互相竞争的标准所描述：RFC 7159 以及 ECMA-404。ECMA 标准更轻量，仅仅规定了可以使用的语法规则；而 RFC 则基于句法上的以及安全上的考量提供了更细致的规范。JSON 官方的网络媒体类型是 application/json。JSON 文件名的后缀是 .json。

— [*维基百科*](https://en.wikipedia.org/wiki/JSON)

相关规范：

* [介绍 JSON](http://json.org/)
* [JSON API](http://jsonapi.org/)
* [数据交换格式 JSON](http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf)

##### 网络内容无障碍指南 (WCAG) & 可无障碍访问的富互联网应用 (ARIA)

可达性讨论的是对残障人士友好的产品设计，设备，服务或者环境。无障碍设计通过使用对残障人士友好的技术（例如计算机屏幕读取仪）对“直接访问”（例如，普通人的访问）和“间接访问”进行兼容。

— [*维基百科*](https://en.wikipedia.org/wiki/Accessibility)

* [可无障碍访问的富互联网应用 (WAI-ARIA) 现状](http://www.w3.org/standards/techs/aria#w3c_all)
* [网络无障碍倡议 (WAI)](http://www.w3.org/WAI/)
* [网络内容无障碍指南 (WCAG) 的现状](http://www.w3.org/standards/techs/wcag#w3c_all)

# 前端开发技能



假设每个领域的前端开发者都使用基础的高级 HTML、CSS、DOM、JavaScript、HTTP/URL和浏览器技能。

除了 HTML、CSS、DOM、JavaScript、HTTP/URL以及浏览器开发的专业知识，一个前端开发者还需要掌握以下一项或多项技能：

* 内容管理系统（亦称 CMS）
* Node.js
* 跨浏览器测试
* 跨平台测试
* 单元测试
* 跨设备测试
* Accessibility / WAI-ARIA
* 搜索引擎优化（亦称 SEO）
* 交互或用户界面设计
* 用户体验
* 适用性
* 电子商务系统
* 门户系统
* 线框绘制
* CSS 布局/ Grids
* DOM 操作（比如 jQuery）
* 移动 Web 性能
* 负载测试
* 性能测试
* 渐进增强/优雅降级
* 版本控制（比如 GIT）
* MVC / MVVM / MV\*
* 函数式编程
* 数据格式（比如 JSON，XML）
* 数据API（比如 Restful API）
* Web 字体嵌入
* 可缩放矢量图形（亦称 SVG）
* 正则表达式
* 内容策略
* Microdata / Microformats
* 任务管理器，构建工具，过程自动化工具
* 自适应网页设计
* 面向对象的程序设计
* 应用程序构建
* 模块
* 依赖管理
* 包管理
* JavaScript 动画
* CSS 动画
* 图表/图形
* UI 控件
* 代码质量测试
* 代码覆盖率测试
* 代码复杂性分析
* 集成测试
* 命令行/命令行界面
* 模板策略
* 模板引擎
* 单页应用
* XHR 请求（亦称 AJAX）
* Web /浏览器 安全
* HTML 语义
* 浏览器开发工具

# 团队里的前端

通常，前端开发者是团队中的唯一角色，他们设计并开发 web 站点、web 应用或基于 web 技术的原生应用。

为了构建**专业**网站或 web 平台软件应用，一个基本的开发团队通常至少包含以下岗位。

* 视觉设计师（也就是字体、颜色、间距、情调、视觉概念和主题）
* UI／交互设计师／信息架构师（也就是线框、所有用户交互和 UI 功能、的指定，信息的架构）
* 前端开发者（也就是编写运行在客户端／设备上的代码）
* 后端开发者（也就是编写运行在服务端的代码）

这些岗位参照技能重叠来排序。通常前端开发者和后端开发者一样能较好地处理 UI／交互设计。团队成员通过接手重叠部分的职责，来担任多于单人的职责，这并不是稀罕的事。

假设上述团队是由项目经理或一些项目委托人（也就是参与人、项目管理者、项目经理等等）来指挥。

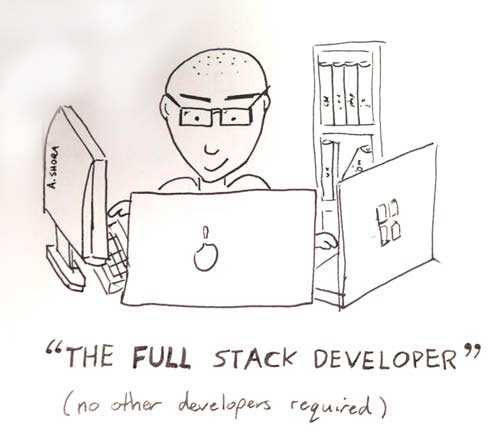
大型 web 团队可能包含下列岗位，这些岗位尚未提及：

* SEO 策略师
* 运维工程师
* API 开发者
* 数据库管理
* QA 工程师／测试者

###### 注意:

一个小趋势似乎正在发生：“全栈开发者” 同时担任前端和后端开发者的职责。

# 全能型人才/全栈神话



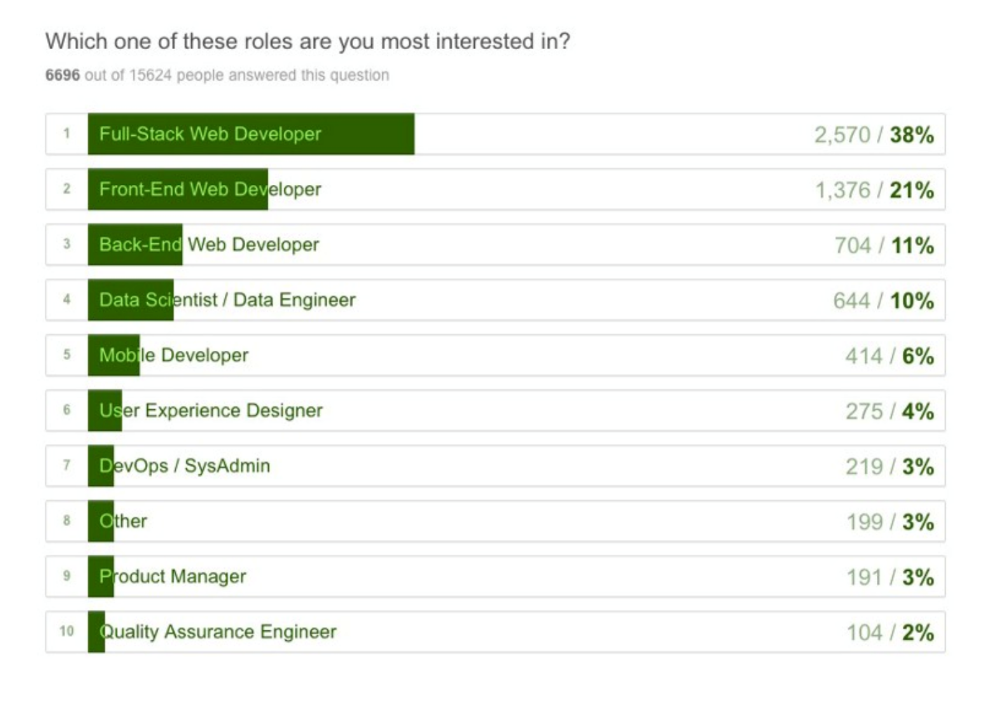
一个能够设计和开发整个 web 解决方案的人，需要具备非常强的专业技能和在视觉设计、UI/交互设计、前端开发、后端开发等领域非常丰富的经验。能这四个领域里精通一门或多门技术的人，就已经可以说是非常稀有的人才了。

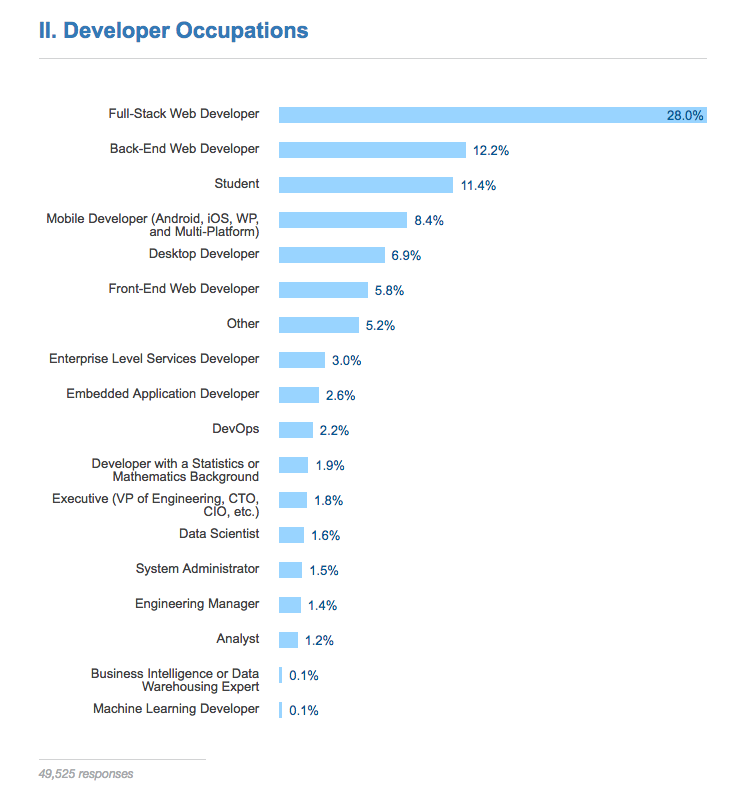
老实说，你应当努力成为，或者尽力聘请到这些领域之一的一个专家（例如：视觉设计、交互设计/信息架构、前端开发、后端开发）的专家。声称自己在上述领域是专家的人非常稀少，甚至少的出奇。

不过，考虑到 JavaScript 已经渗透到了整套技术栈的所有层面（例如：React, node.js, express, couchDB, gulp.js 等），找到一个会前端开发和后端开发的 JavaScript 开发者并没有那么难。通常，这些全栈开发人员只需要关心 JavaScript —— 不像先前那样荒谬（还要关心视觉设计、交互设计和 CSS）。虽说在我看来依然少的出奇，但至少不像以前那么难找。因此，我并不建议开发者开始转向全栈工程师。在少数情况下可能有好处，但就职业发展来说，我认为前端开发工程师还是应当着重关注前端相关的技术。

###### 备注:

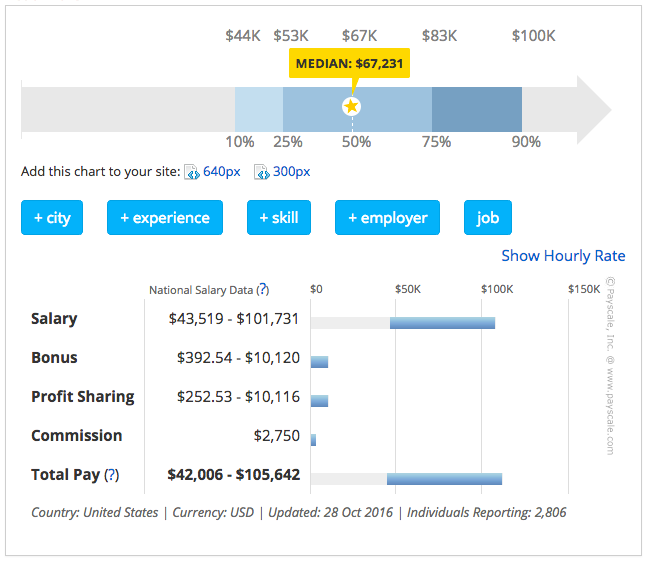
“全栈开发者”已经变为一个拥有多种含义的术语。也就是说使用这个术语时，它可能包含不止一层含义。分析一下下面的两个调查，从调查结果可以看出大多数开发者都是全栈开发者。但以我近 20 年的经验来看，这并不是真的。





# 前端开发薪资

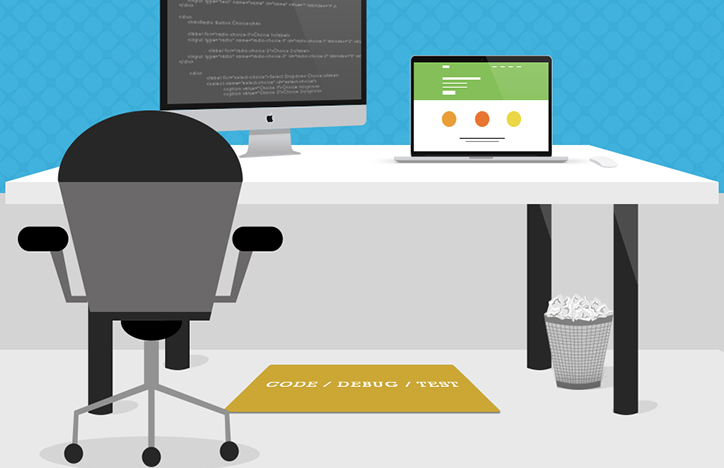
在美国，中等水平的前端开发者的平均年薪大约为 [75K 美元](http://www.glassdoor.com/Salaries/front-end-web-developer-salary-SRCH_KO0,23.htm)。当然，在刚开始进入前端开发领域时，根据开发者的所在地区和工作经验，期望年薪应该在 35K 美元左右。



###### 注意：

高级前端开发者或工程师通常可以自愿选择办公地点（例如在家远程办公）并且年薪超过 150K 美元（登陆 [angel.co](https://angel.co/jobs) 或在 [Stack Overflow Jobs](https://stackoverflow.com/jobs?q=front-end&sort=y) 上查看年薪 150K+ 美元的前端开发工作）。

# 前端开发者是如何诞生的



到底如何才能成长为前端开发者？这很难说。现如今，你仍然不能指望在大学毕业的时候拿到一个关于前端工程的学位。此外，我几乎没听说或见过一个前端开发者需要遭受一种过时的计算机科学学位或图形设计学位的折磨，才能专业地完成 HTML、CSS 和 JavaScript 编码工作。在我看来，如今大部分在前端圈工作的人，通常像是自学成才或来自非认证机构、课程及训练营。

如果你打算成为一个前端开发者，我将尽量让你跟随以下概括好的步骤（第二部分，["学习前端开发"](https://frontendmasters.gitbooks.io/front-end-handbook-2017/content/learning/self-direct-learning.html)，深入了解更多学习资源的细节）。

1. 请大致了解一下 WEB 的工作方式，并确认自己了解以下概念是什么，会在哪里出现：域名、DNS、URL、HTTP、网络、浏览器、服务器／托管、JSON、数据 API、HTML、CSS、DOM 和 JavaScript。你可以不求甚解，只需要大致理解各部分是如何协作的。请关注前端架构的主目录（high level outlines）。从编写简单的 [web 页面](https://github.com/h5bp/html5-boilerplate/blob/master/dist/index.html)和简要地学习[前端应用（又叫 SPA）](http://developer.telerik.com/featured/front-end-driven-applications-new-approach-applications/)开始。
2. 学习 HTML。
3. 学习 CSS。
4. [学习 JavaScript](https://youtu.be/QjKH1J77gjI?list=PL055Epbe6d5bQubu5EWf_kUNA3ef_qbmL)。
5. 学习 DOM。
6. 学习 JSON 和数据 API。
7. 学习用户交互设计（也就是 UI 模式、UI 设计、用户习惯设计和可用性）的基础。
8. 学习命令行界面／命令行。
9. 学习软件工程的实践（也就是应用设计／架构、模版、Git、测试、监控、自动化、代码质量、开发方法论）。
10. 使用你觉得有用的东西（也就是 Webpack、React 和 Redux）配置和定制自己的工具箱。
11. 学习 Node.js。

对学习的简短建议：[在学习工具和框架前应当先学习更底层的实现技术](https://youtu.be/QjKH1J77gjI?list=PL055Epbe6d5bQubu5EWf_kUNA3ef_qbmL)，学 DOM 而非 jQuery。学 CSS 而非 SASS。学 HTML 而非 HAML。学 JavaScript 而非 CoffeeScript。学 JavaScript ES6 模版而非 Handlebars。学 UI 模式而非使用 Bootstrap。

刚起步的时候，你应当对那些把 WEB 开发的复杂性掩盖掉的工具和框架保持警惕。高级工具和框架的不当运用会给人一种使用了高级技能的表象，而掩盖了这样的事实：开发者对于基础和底层概念所知寥寥。

本书的剩余部分会为你指出可能的资源，供你学习前端开发和实践开发所需的工具。在此过程中，我们的前提假设是你不仅学习新知识，而且学以致用并研究工具的使用方法。有人说实践出真知，还有人说学以致用，而我的建议是，请结合以上两个观点，找到合适自己的工作方式并践行它。但毫无疑问的是，要结合起来！因此别止步于看，要实践起来！学习，实践，学习，实践，学习，实践。要不断重复两者，以应对飞速变化的东西。这就是要学习基础知识而非高级工具的原因，学习基础知识十分重要。

近来出现了很多非认证的，收费昂贵的前端编码学院和训练营。这些成为前端开发者的途径，往往是由老师主导，由官方讲师制作的课程，它们遵循更传统的学习方式（也就是教学大纲、测试、小测验、项目、团队项目、分数等等）。如果你正在考虑是否要报一个收费昂贵的课程大纲，请牢记你是在互联网上！你可以在网上免费获得所有需要学习的东西。然而，如果你需要一个老师，要求这个老师能教你如何获取并学习到真正免费的知识，而且还要对你的学习负责，那你可能就要考虑一下系统化的课程了。否则，你学习任何专业技术所需的网络资源，每月收费几百块的视频资源会员，以及使你对知识怀有强烈的渴望，这些实际上都不是免费的。