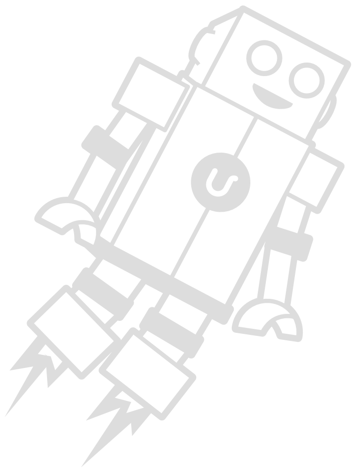
UDACITY

Ultimate Skills Checklist for Your First Front-End Developer Job

Welcome



Welcome to your ultimate skills checklist for getting your first job as a front-end web developer! You're embarking on a road to an exciting career, filled with opportunities to improve the lives of others and expand your own creative capabilities.

Having choices is always a good thing. But sometimes it's helpful to have a guide, so we're here to help you cut the noise.

We recently launched our Front-End Web Developer Nanodegree, which guides students along a project-based curriculum to learn the skills they need to get their first job in web development. We learned a TON from talking to employers to make sure our skills list is cutting edge, and we can't wait to pass this skills list on to *you*.

In this guide, you'll find the ultimate skills checklist for getting a job as a front-end web developer, as well as resources where you can get started.

Congratulations on taking a step towards advancing your web development career! Read on for the ultimate front-end web development skills checklist and recommended resources.

Front-End Web Developer Skills Checklist: What We’ll Cover

Here’s a breakdown of the skills you need to learn to be a data analyst. Take some time to review this list - how many boxes can you check off?

For more detail on these skills and for learning resources, navigate to the corresponding pages listed.

HTML

HTML is the first of the big three languages you need to learn to make websites—HTML, CSS and JavaScript. HTML isn't a programming language per se. It describes how elements on a website should be laid out and provides browsers with a list of all the other files, like CSS and JavaScript, that websites need. You can think of HTML like the blueprint of a house. It tells you how big the rooms are and what should be inside them. But it doesn't tell you how they should look.

```
<!DOCTYPE html>
<head>
  <meta charset=
  <title>Awesome
  <link rel="stylesh
</head>
<body>
  <div class="top">
    <div class="container">
      <div class="three columns info-top">
        <p class="font-icon">q</p>
        <p>(123)-456 -789-00</p>
      </div>
      <div class="four columns info-top">
        <p class="font-icon">w</p>
        <p>yourmail@company.com</p>
```

- ☐ **Semantic elements**: A semantic element clearly describes its meaning to the browser and the developer; elements like article and section, rather than using div everywhere
- ☐ **Block-level elements**: A block-level element occupies the entire space of its parent element
- ☐ **Inline elements**: An inline element occupies only the space bounded by the tags that define the inline element
- ☐ **Forms**: A form represents a document section that contains interactive controls to submit information to a web server
- ☐ **Input Types**: An input element is used to create interactive controls for web-based forms in order to accept data from the user

CSS

If HTML describes the layout of a house, CSS describes the look of a house. CSS, or Cascading Style Sheets, is responsible for the way a website looks. Colors, fonts, and even some animations are all controlled by CSS. Like HTML, CSS isn't a programming language. It's a text document that reads like an interior designer's instructions for making a website look great.

```
<!DOCTYPE html>
<head>
   <meta charset=
   <title>Awesome
   <link rel="stylesh
</head>
<body>
   <div class="top">
      <div class="container">
         <div class="three columns info-top">
            <p class="font-icon">q</p>
            <p>(123)-456 -789-00</p>
         </div>
         <div class="four columns info-top">
            <p class="font-icon">w</p>
            <p>yourmail@company.com</p>
```

- ☐ **Display Value Types**: The display attribute lets you control the rendering of graphical or container elements
- ☐ **The Box Model**: The box model defines the size of the rectangular box representing an element in a document
- ☐ **Basic Positioning**: The position property chooses alternative rules for positioning elements
  - ☐ **Static:** Static positioning lets the element use the normal behavior
  - ☐ **Absolute**: Does not leave space for the element, instead position is at a specific position relative to its closest ancestor or the containing block
  - ☐ **Fixed**: Does not leave space for the element, instead position is specified relative to the screen's viewport
  - ☐ **Flexbox**: A layout mode providing for the arrangement of elements on a page so that the elements behave predictably when the page layout must accommodate different screen sizes and different display devices
  - ☐ **Float**: Specifies that an element should be taken from the normal flow and placed along the left or right of its container
- ☐ **Font Styling and Web Fonts**: Font styling allows the ability to change the appearance of text; web fonts allow for loading web-based font files that are usable by all clients
- ☐ **Backgrounds**: Backgrounds allow you to define a color or image to be used as a container's background
- ☐ **Pseudo-selectors**: Pseudo-selectors allow you to select hypothetical elements that exist around elements you have defined within your HTML
- ☐ **Animations and transitions**: Animations and transitions allow you to animate elements or define the transition between two states of an element

JavaScript

Of the big three web languages, JavaScript is the only one that's an actual programming language. JavaScript controls interactions on a website. It's like a handyman that can tear down walls, build new rooms and redecorate your house. For a simple, static website, you won't need to use much JavaScript. But for a dynamic web app you'll need an in-depth understanding of the language.

```
<!DOCTYPE html>
<head>
  <meta charset=
  <title>Awesome
  <link rel="stylesh
</head>
<body>
  <div class="top">
    <div class="container">
      <div class="three columns info-top">
        <p class="font-icon">q</p>
        <p>(123)-456 -789-00</p>
      </div>
      <div class="four columns info-top">
        <p class="font-icon">w</p>
        <p>yourmail@company.com</p>
```

- ☐ **Syntax**: The general rules defining how a language is typed
- ☐ **Data types**: The different types of variables the language supports (for example, strings and integers)
- ☐ **Functions**: A block of code designed to perform a particular task
- ☐ **Object Literals**: Everything in JavaScript is an object but writing your own object literals can simplify your code
- ☐ **Object-Oriented Programming**: JavaScript provides a number of ways to implement object-oriented programming including functional, prototypal and pseudoclassical
- ☐ **Design Patterns**: A design pattern is a reusable solution to a commonly-occurring problem
- ☐ **AJAX**: AJAX provides the ability to asynchronously request data from a web server without requiring a page reload
- ☐ **jQuery**: jQuery is an extremely popular library that makes cross-browser DOM traversal and manipulation, event handling and AJAX much simpler

Responsive Web Design

Open a website and make your browser smaller. Did the page content change its layout to fit the new screen? That's responsive design. People expect modern websites to look great on their phones, tablets and laptops. By diving into responsive design principles, you'll learn how to make websites that scale and adjust themselves to offer amazing experiences regardless of the device.

- ☐ **@media queries:** Media queries let the presentation of content be tailored to a specific range of output devices without having to change the content itself
- ☐ **Relative sizing units**: CSS provides many other units of measurement than just pixels (px), such as em, rem, vw, vh and vmin

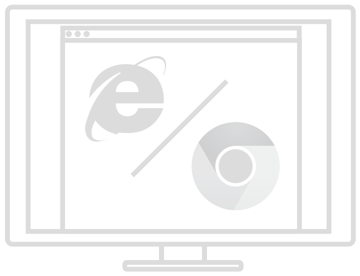
CSS Frameworks

Bootstrap is a good example of a CSS framework. Frameworks make it easy to structure and build websites. They provide custom CSS classes that simplify laying out content and ensure that your content looks great no matter the device. Frameworks can help you follow industry best practices and modern design principles.

- ☐ **Bootstrap**: Bootstrap is a CSS framework, originally developed at Twitter, that makes creating responsive designs much simpler
- ☐ **Foundation**: Foundation is another CSS framework, developed by Zurb, that also makes creating responsive designs easier

JavaScript Libraries and Frameworks

JavaScript libraries and frameworks make it easier to write your web application by enforcing various best practices and, often, an organizational pattern for the various files you'll be working with. They also address the majority of cross-browser compatibility issues you may experience and include various performance optimizations. Examples are AngularJS, EmberJS and KnockoutJS.



- ☐ **AngularJS**: features two-way data binding and allows you to extend HTML vocabulary to create front-end web applications
- ☐ **EmberJS**: removes the need for boilerplate code through the use of strict file and object naming conventions
- ☐ **KnockoutJS**: makes creating data-driven applications easier through its declarative binding system

Version Control

With complex projects, how do you maintain your code? Version control software like Git helps software developers around the world save and maintain their code, even as projects grow to hundreds of developers and dozens of subprojects.

```
<!DOCTYPE html>
<head>
    <meta charset="utf-8">
    <title>Awesome Site</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="top">
        <div class="container">
            <div class="three columns info-top">
                <p class="font-icon">q</p>
                <p>(123)-456 -789-00</p>
            </div>
            <div class="four columns info-top">
                <p class="font-icon">w</p>
                <p>yourmail@company.com</p>
```

- [ ] **Git**: Git is a distributed version control system
    - [ ] clone
    - [ ] add
    - [ ] commit
    - [ ] push
    - [ ] pull
    - [ ] branch
    - [ ] log
- [ ] **GitHub**: GitHub is a web-based hosting service for Git repositories that offers a number of other features to support collaboration between developers
    - [ ] forking
    - [ ] pull requests

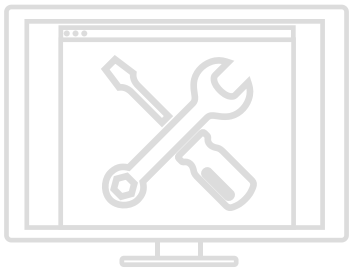
Web Performance

Once you've made your website, how will you ensure it's fast? By understanding a few simple principles of browser rendering, you'll be able to make sure that you deliver fast, efficient websites to your users.

- ☐ **Critical rendering path**: The critical rendering path is the process that browsers use to transform your HTML, CSS and JavaScript into actual pixels that are sent to the user's screen
- ☐ **Image optimization**: Image optimization is the process of using proper image types for the content of the image as well as removing the extra metadata within an image file
- ☐ **JavaScript minification**: JavaScript minification is the process of removing unnecessary characters from your JavaScript files to reduce their filesize

Browser Developer Tools

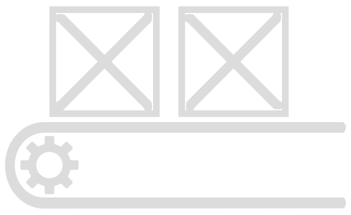
In building your web applications you are destined to experience bugs in your code, performance issues, or quirks in how your pages are rendered by the browser. The only way to fix these issues is to understand how the browser is actually interpreting your code. Browser developer tools provide all the nitty-gritty details and are your window “under the hood” to test, measure and iterate on your code.



- ☐ **Element inspecting**: The elements panel lets you see everything in one DOM tree and allows inspection and on-the-fly editing of DOM elements
- ☐ **Network**: The network panel records information about each network operation in your application
- ☐ **Timeline**: The timeline panel lets you record and analyze all the activity in your application as it runs
- ☐ **Application profiling**: The profiling panel lets you observe the memory usage of your application as it runs
- ☐ **Resources**: The resources panel lets you inspect resources that are loaded into your application

Building and Automation Tools

There's a lot more to building a web application than just writing code! You have to run test suites, optimize images, adhere to your organization's code style guides and even prepare your code for deployment on a production server. That's a lot of extra, repetitive, and often mundane work to be done. Building and Automation Tools like Grunt and Gulp can handle running all of these tasks for you in the background so you can focus on building a great web application.



- ☐ **npm**: npm is the default package manager for Node.js, which is the framework that most build and automation tools are written in
- ☐ **Grunt**: Grunt is a task-based command line build tool that interacts with files on your hard disk
- ☐ **Gulp**: Gulp is a program-based command line build tool reads files on your hard disk, then interacts with those files as streams
- ☐ **Bower**: Bower is a package manager for HTML, CSS and JavaScript libraries that allows you to define, version and retrieve your dependencies
- ☐ **Yeoman**: Yeoman is a scaffolding application that will automatically generate boilerplate code for a variety of applications based on the frameworks and libraries you define

Testing

As your application becomes more complex it become quite easy to introduce bugs or completely break existing functionality. Unit, integration and behavior testing are excellent ways to ensure that you don't break your application when you are adding new features. Examples of excellent testing frameworks include Mocha and Jasmine.

- ☐ **Mocha**: Mocha is a JavaScript testing framework that runs in Node.js and the browser making asynchronous testing simple
- ☐ **Jasmine**: Jasmine is an open source, behavior-driven testing framework for JavaScript

Soft Skills

It’s important to have the right technical skills for the job, but just as important are the soft skills that enable you to produce high quality work both in a team environment and on your own. Cultivating your ability to share insights, teach and inspire others, and see the bigger picture beyond the job make you a much more desirable choice for potential employers.

- ☐ **Strong communication**: Convey goals, progress and issues to the wider team, including management, peers and clients
- ☐ **Agile problem solving:** Understand the question behind the question and break down the problem into smaller, more easily solvable pieces
- ☐ **Healthy Passion**: Keep up-to-date on advances in the field and be aware of the latest technological advances
- ☐ **Self-Starting Motivation**: Experiment and explore; take risks, but stay persistent while facing roadblocks

What Next? Learning Resources

You made it to the end of the checklist - congratulations!

Whether you were able to check off many skills, or whether you're going to start tackling the checklist from the very beginning, pat yourself on the back for taking a big step by reading this guide.

As we mentioned at the beginning of this guide, we're here to help you cut the noise when it comes to navigating your learning choices.

We invite you to check out our Front-End Web Developer Nanodegree for a structured program to help you learn all these skills with the support of Coaches and fellow students:

In the Front-End Web Developer Nanodegree, you'll work your way through five projects designed to teach you web development fundamentals - as you build a portfolio that will demonstrate your new skills to employers. You can think of this skills checklist as your learning blueprint, and the nanodegree as an action plan.

The nanodegree is a new type of credential designed to prepare you for a career, and it's a big commitment at a minimum of 10 hours a week for 9 to 12 months.

If you are looking for a learning plan with lower time commitment, or if you're looking to fill a specific gap in your skill set, check out our individual courses:

Intro to HTML and CSS - Learn how to convert digital design mockups into static web pages and how to build a responsive portfolio site to showcase your work.

JavaScript Basics - Learn JavaScript syntax and coding conventions that web developers use to create interactive and dynamic websites while you create an online résumé for your portfolio.

Intro to jQuery - Learn how to access and modify the DOM with ease using jQuery! This course will teach you how to use jQuery's core features - DOM element selections, traversal and manipulation.

Object-Oriented JavaScript - Build a variety of JavaScript objects and explore how their inheritance models affect your app's in-memory model. Gain simplicity and modularity in your own code.

HTML5 Canvas - Learn how you can use HTML5 Canvas to create and modify images or even interactive animations.

Website Performance Optimization - Learn how browsers convert HTML, CSS and JavaScript into websites while you experiment with Chrome Developer Tools to measure and optimize website speed!

Intro to AJAX - In this course you will learn how to make asynchronous requests with JavaScript using jQuery's AJAX functionality, and how to perform asynchronous requests as you build a web app with data from Google Street View, the New York Times and Wikipedia!

Check out these resources and communities to stay up-to-date in this exciting field:

Our good friends

- ☐ Ilya Grigorik
- ☐ Paul Lewis
- ☐ Paul Kinlan
- ☐ Pete LePage

Books

- ☐ Dive Into HTML5
- ☐ Eloquent JavaScript
- ☐ JavaScript Design Patterns

Newsletters

- ☐ HTML5 Weekly
- ☐ Tales from the Front End
- ☐ JavaScript Weekly
- ☐ Hacker Newsletter

Blogs/Websites

- ☐ HTML5 Doctor
- ☐ HTML5 Hub
- ☐ HTML5 Rocks
- ☐ Smashing Magazine
- ☐ A List Apart
- ☐ CSS Tricks

UDACITY