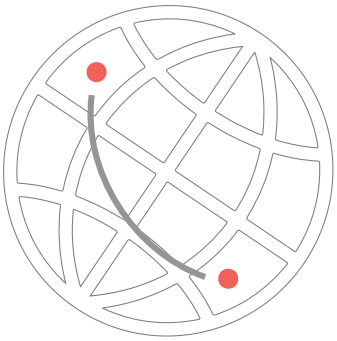
The Ultimate Guide to

CODING FOR BEGINNERS

CodeScholar.ly

DON'T BLOW THIS OPPORTUNITY: YOU MUST LEARN HOW TO CODE!



You are in possession of the most powerful machine man has ever made.

More powerful than the wheel.

More powerful than the printing machine.

More powerful than a freight train.

More powerful than an atom bomb (and hopefully a lot less destructive).

What makes this machine so powerful?

Simple: it can connect you to anyone, anywhere on the planet.

In fact, it can connect you to everyone, everywhere on the planet. In a matter of milliseconds.

And that's barely the beginning of what it can do.

It can dismantle nation states, it can perform calculations at speeds that are beyond our human comprehension, it can and is fundamentally changing the way we live, work, and play.

It can do all of this, while still being small and inexpensive enough that you can own it (and sometimes even hold it in your hand). But, if you're like most people, chances are you're taking advantage of less than 1% of what this machine can do.

BEFORE YOU BEGIN

GET DOWN WITH TECH LINGO

One of the first things you need to learn when you're thinking about starting a career in tech is the lingo. You've probably heard the basics before: terms like HTML, CSS, WordPress, etc. But do you know what those things actually are?

You don't have to memorize every single tech term out there. But being familiar with the major ones is really important, if for no other reason than that you know what to search for when you're learning new things.

Knowing the tech terms you're most likely to encounter makes it easier to ask the right questions from the right people. After all, if you know what the difference between UX and UI is, then you'll know an employer is looking for when they say they're hiring a UI designer.

AGILE, OR AGILE SOFTWARE DEVELOPMENT

A set of principles for coding software that prioritizes "continuous improvement" by launching as soon as possible and releasing frequent updates to a piece of software instead of waiting until it's perfect.

BACK END
Part of a website or web service that makes it work and includes applications, web servers, and databases.

BUG
Mistake or unwanted piece of code that keeps a website or program from working like it should. More specifically, you call something a bug when it's not working as expected.

CLOUD COMPUTING
Storing and accessing information and services via the Internet.

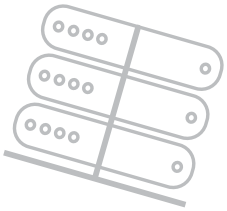
CODE
A simplified form of language with very strict rules and syntax used by humans to tell computers what to do.

CODING LANGUAGE
A *specific* set of rules and syntax for writing the code that tells computers what to do. Includes programming, assembly, and markup languages such as Ruby, PHP, and HTML.

COLOR THEORY
Characteristics of colors and the relationships between them.

COMPUTER PROGRAMMING
The process of writing and implementing various instructions for a computer to do a particular task (or set of tasks), using code.

CSS (CASCADING STYLE SHEETS)
Code that tells browsers how to format and style HTML for a web page; controls things such as font type and colors.



CSS3
The most current version of CSS.

DATABASE
Collection of electronic information (data) stored on a web server.

FRONT END
The part of a website that can be seen by users and is made up of HTML, CSS, and JavaScript code files.

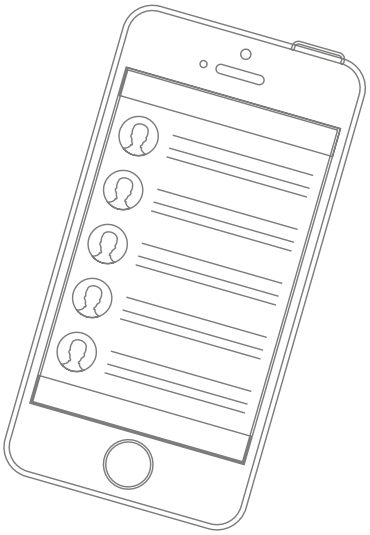
GRID SYSTEM
Set of columns and rows that can be used as guidelines to arrange content on a web page.

HTML (HYPERTEXT MARKUP LANGUAGE)
A coding language used to put content on a web page and give it structure. Since HTML doesn't tell computers to do anything, it's not considered a programming language (this is a distinction that only matters in job interviews when an interviewer asks if you can "program").

HTML ELEMENT
HTML code made up of an opening tag, a closing tag, and information between them.
Example: This is my paragraph element!

HTML5
The most current version of HTML.

HTML5 APP
A web application designed specifically for use on mobile phones using the latest HTML5 and JavaScript technologies.



INTERNET
A network of interconnected computers all over the world, not to be confused with the Web.

LEAN OR LEAN STARTUP
A popular process for launching products and quickly iterating on them to better meet customer needs, based on continuous customer feedback. Think of it like agile but for companies. Popularized by the book *The Lean Startup*.

MOOD BOARD
Collection of content showing the visual style for a website including color palette, images, icons, fonts, etc.

MINIMUM VIABLE PRODUCT, MVP
A product with the minimally adequate features to meet the needs of early adopters, often used to test a concept or idea without a huge outlay of resources. Popular among lean startups.

NATIVE APP
A mobile app built using the software development kit (SDK) native to a specific mobile device.
Example: any app coded for the iOS (Apple) operating system

OBJECT-ORIENTED PROGRAMMING (OOP)
A popular way to design software programs (commonly known as a design pattern) where code is organized into objects that have specific and unique attributes and abilities.
Example: A blog might include a blog post object that has a title, date, and content attribute
Examples of OOP language: Ruby, PHP, Python

PROGRAMMING LANGUAGE

Technically a subset of coding languages that specifically tell computers what to do vs. how to display something. For example, HTML and CSS are NOT considered programming languages but instead are markup languages.

RESPONSIVE DESIGN & DEVELOPMENT

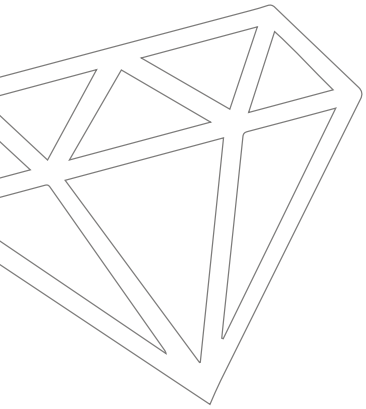
A way to design and code websites such that they can adapt to different-sized devices like phones, tablets, wearable devices, etc.

SDK (SOFTWARE DEVELOPMENT KIT)

Set of tools for creating a specific kind of software.

SEMANTIC ELEMENT

HTML element that gives the browser more information about the content in it.
Examples: aside (for sidebars), header, footer.



SEMANTIC WEB

A design theory for the web whose premise is that all data should be properly named and stored so that it can be more easily accessed and reused in the future.

SITEMAP

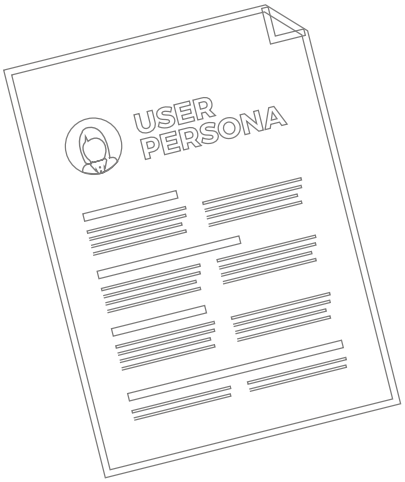
In web development, an outline, or map, of the pages needed for a website. Usually drawn using lines and boxes to visualize the hierarchy of pages.

SOFTWARE DEVELOPMENT

The process of programming, documenting, testing, and bug fixing involved in creating and maintaining all manner of software applications and frameworks.

TEXT EDITOR

Software used to write plain text (text with no formatting) that's used for coding and programming. Examples: SublimeText, TextEdit, TextWrangler, Notepad++



UI (USER INTERFACE)

How a website is laid out and how users interact with it.

USER FLOW

Map of the path users take from getting to a website to taking an action on the site.

USER PERSONA

Profile of an imaginary person who would use a website; used to define who a site is for and what their needs are.

USER RESEARCH

Investigating how users act and what they need and want in order to better design a website for them.

UX (USER EXPERIENCE)

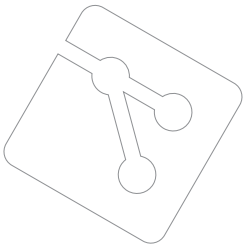
What a user experiences when they browse a website; this can range from straightforward usability (can they accomplish a given task?) to the less tangible (what do they feel when they're on the website?).

VERSION CONTROL

A software used to keep track of changes to code files, similar to the track changes feature of Word. Used by software teams so that they can work on the same code files at the same time without overwriting one another's work.
Example: Git, Subversion

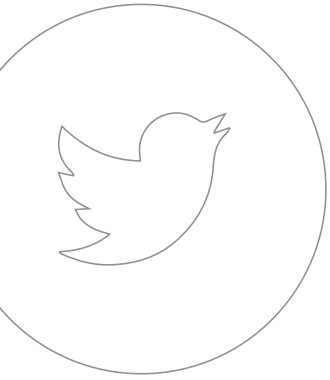
VIRTUAL REALITY OR VR

A computer-generated simulation of a three-dimensional environment that users can interact with in a somewhat realistic way, often using equipment like a helmet with a screen or interactive gloves.



WEB APP OR WEB APPLICATION

A website with complex functionality and heavy interactivity.
Example: Twitter, Facebook, Bank of America



WEB APPLICATION FRAMEWORK

A series of pre-written code that is used by developers as a starting point to building their web applications.
Examples: Ruby on Rails, Bootstrap, AngularJS

WEB OR WEB BROWSER

The Web is everything you access using a web browser. Web browsers are software applications that allow you to access information (websites) stored on other computers connected to the Internet. Not to be confused with the Internet itself ;) Also the Web is always capitalized!

WEB DESIGNER

A designer who specializes in designing websites and web applications for use on the Web.

WEB DEVELOPER

A software developer who specializes in coding websites and web applications for use on the Web.

WEB SERVER

A computer that can be accessed through the Internet and stores information in the form of websites. Whereas your computer only makes orders ("Give me google.com!"), web servers can give AND receive orders ("Here is google.com.").

WIREFRAME

A simple sketch of the key information that goes on each web page, usually done in black and white with boxes, line, and placeholder text.

WHAT YOU SHOULD KNOW NOW

You've familiarized yourself with key tech terms that you'll hear in the industry. And you've taken the Codescholarly Bootcamp to get a more comprehensive view of the basics of tech.

You're probably getting an idea of just how many opportunities tech skills can open up for you, not to mention the job security that goes along with knowing tech. Before you dive into learning the skills that will really make you an invaluable employee (or rockstar freelancer), you should make sure you've got the basics down.

At the end of this phase, you should be familiar with the following:

- Common terms you'll hear in tech
- How the web works
- What HTML, CSS, and JavaScript are and how they work
- The difference between website front ends and back ends
- How all the parts of a website come together into what you see when you're browsing the web

BUILD YOUR FOUNDATION

step 1

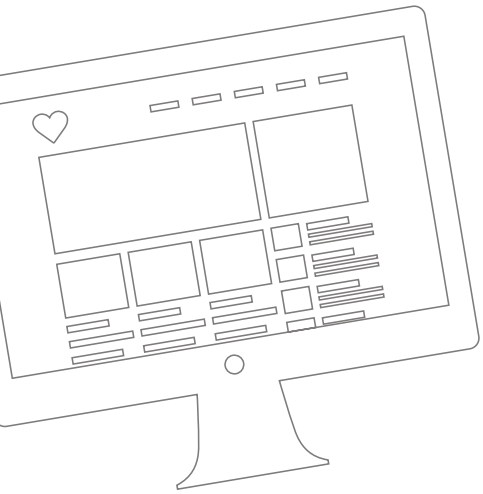
LEARN ABOUT WEB DESIGN

You might have a rough idea of what “web design” is, but there’s more to it than just creating a pretty design. Websites aren’t the same as art you'd hang on your wall.

Websites are built with a purpose: it could be to provide information, to sell something, to entertain the visitor, to challenge the visitor, or something else entirely. And more importantly, they’re interactive.

The form of the website and the way it looks has to serve the function of the site. When the form and the function work together, the site is considered to have good UX, or User Experience.

The fundamentals of design are the same regardless of the medium. Considerations like balance, harmony, and color theory apply whether you’re designing a building, a T-shirt, or a website. But as mentioned above, websites have to function in ways that other types of design don’t have to worry about.

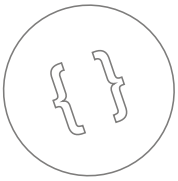


Learning about the basics of web design is an important first step in any tech career. Even if you later decide you want to work on the coding end of things, understanding how web design works and what makes a good design is invaluable in any area of tech.

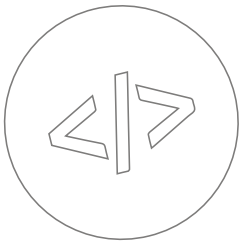
step 2

HTML & CSS

HTML and CSS are the building blocks of the web. You can build entire websites with just HTML and CSS. In fact, with the newest versions of both, you can also build games, animations, and more.



Think of HTML like the framing for the walls and roof of a house. They form the structure of the house and the basics of how it's laid out. You can tell it's a house by looking at it, but it's not necessarily very functional or beautiful like that. It's just a shell.



CSS adds things like the finishes on the walls and floors, the windows and doors, and all the other things that make the house comfortable and livable. You can even use CSS to move the parts around and configure them differently (just like swapping the furniture in a house can transform the purpose of different rooms and alter the layout).

By separating the content (HTML) from the presentation (CSS), you can change the way your page looks without having to rebuild everything from scratch, and you can easily add new content without having to design the whole page every time you want to add something.

HTML and CSS are vital skills to learn, whether you want to be a web designer or web developer. They're the most basic parts of the web, and understanding what you can (and can't) do with each one makes your projects that much easier to complete with minimal frustration.

UNDERSTANDING HTML, CSS & JAVASCRIPT

Think of HTML, CSS & JavaScript as parts of a building.

1

HTML creates the structure of the building. It's the foundation, the walls, and the roof. With just these parts, you can recognize it as a building (even if it's not a particularly inviting one!).

Here's what some of the HTML that makes up the Codescholarly website looks like (the HTML is the tags that have a letter and sometimes a number inside two bracket like this <h2>):

```
<h2>After completing one of our Career Blueprints you
will be able to:</h2>
<ul>
    <li>Make more money</li>
    <li>Feel confident in your job security</li>
    <li>Work the hours you want</li>
    <li>Build the career of your dreams</li>
</ul>
```

As you can see, HTML wraps the content of a website and gives it structure. Here you see a second level headline and an unordered list with 4 list items. Nice, right?

2

CSS makes the building more attractive and inviting. Think of CSS as like the paint color, the flooring, the trim details, and the interior design. It can turn that barebones building into something people actually want to live and work in.

Here's what some of the CSS from the Codescholarly site looks like:

```
.blog-landing p {
   font-size: 16px;
   line-height: 25px;
}

.blog-landing .entry-excerpt + p {
   max-height: 100px;
   overflow: hidden;
}
```

As you can see, the CSS dictates what the HTML should look like, what its font-size is, its line height, and its width.

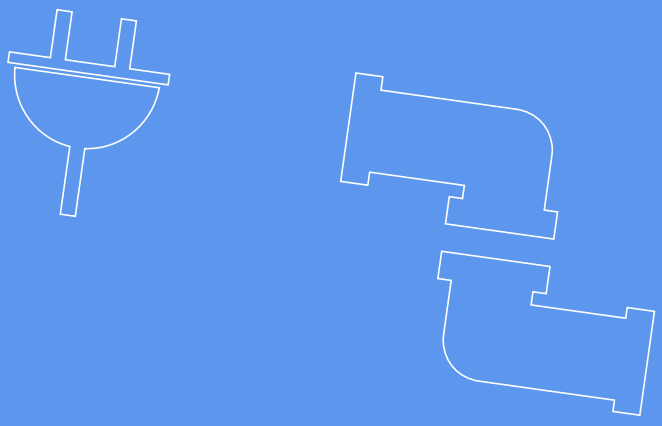
3

JavaScript is another important part of many modern websites. Like CSS, it's not required, but it sure can be nice. In a building, think of JavaScript like the electrical and plumbing systems, the parts of a house that are active and DO things for you. You don't need electricity or plumbing in your house, but you're gonna want 'em!

Here's an example of JavaScript from the Codescholarly site:

```
$('.show-transcript').on('click', function(){
$('.transcript-wrap').removeClass('full').slideToggle()
.toggleClass('show');
   if($('.transcript-wrap').hasClass('show')){
         $(this).text('Hide Transcript');
   } else {
         $(this).text('View transcript');});
```

What this JavaScript code does is show (or hide) the transcripts below a video. Nice, right?

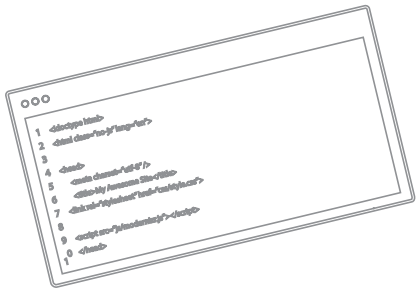


step 3

THE TOOLS OF THE TRADE

As in any industry, there are certain tools you'll need to get comfortable with for a successful tech career. Here are the most common tools you'll want to learn, though different companies might use additional tools or alternatives to the ones mentioned below.

Git: Git is a version control software that developers use for keeping track of code history and changes. Version control software makes it easier to see how code has changed and evolved over time, and also makes it possible to rollback to earlier versions in the event that a new version of the code creates problems or bugs. Git also makes it possible for multiple developers to work on the same code without having to worry about overwriting each other's work.



Text Editor: Professional web developers generally use text editors specifically designed for coding to write and edit the code they create. The difference between a developer-friendly text editor and a plain text app is that the former uses syntax highlighting (usually in the form of different colors for things like HTML tags, CSS, elements, PHP, comments, and the like) to help keep your code organized. Syntax highlighting also makes code about a billion times easier to read and write properly (since it will usually highlight when your code is incomplete or, in some instances, incorrect).

UNDERSTANDING GIT

1. Git is a version control system that keeps track of all the changes you make to your code files. It works a lot like track changes in Word.

   Git can be used via fancy Git software, but most often, you will use Git via your computer's terminal by typing commands like this one:

```
git commit -m "This is a git commit message. It's where
I write a note to myself about the work I just did."
```

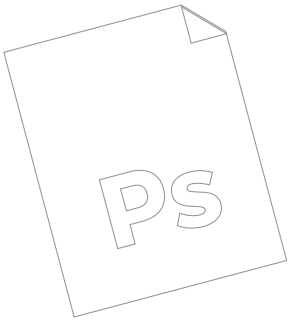
2. When you work as a professional developer you'll want to use Git from the very beginning of every project. What you'll do is:

   - Start tracking your code files with Git
   - Make updates to your code files
   - Save those updates & log those changes in Git with a short note to yourself about the code edits you made
   - Rinse & repeat until your project is done

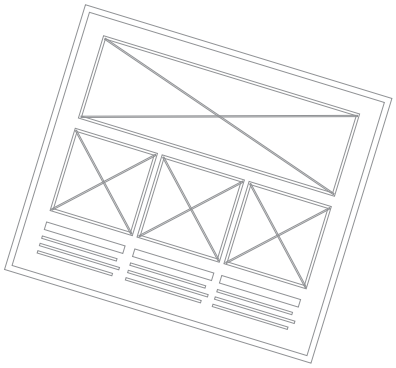
   This will help you keep track of your code as you work on it, organize your changes, and make sure you have a copy of your work in case anything goes wrong.

3. Where Git will be especially useful, however, is when you work with other developers on the same codebase. By using Git you can all work on the same code files at the same time without worrying about overwriting or accidentally losing each other's work.

   You can learn more about Git at: git-scm.com.



Graphics Software: If you're interested in design, you'll need to learn to rock a graphics program to create all of those amazing visuals that you see online. The two most popular ones in the world of web design are Sketch and Photoshop (there are others, but these are the two you're most likely to encounter). Even if you're more interested in web development, becoming familiar with these programs is key so that you can more easily work with the designers on your team.



Prototyping and Mockup Tools: Creating wireframes, prototypes, and mockups is a key part of designing and developing any website or app. Wireframes are essentially a sketch of the layout a site will have (with little or no indication of how the interactive parts work).

Mockups are generally a bit more polished and give a clearer idea of what the final site will look like. Prototypes are usually interactive, and show how an app or website will actually function, often with fake user data and a full picture of the user experience.

A lot of designers still start out with pen and paper for wireframes, but eventually those need to be translated into some kind of digital format. There are dozens of tools out there for creating prototypes and mockups, but most work in similar ways, so once you've learned to use one, the curve for learning others isn't bad.

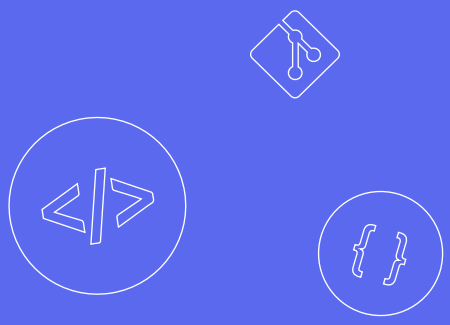
It's vital that you learn to use industry-standard tools when embarking on a tech career. As a hobbyist working alone on projects, it doesn't really matter what you use. But when you're working with a team of other designers and developers, they'll expect that you know how to use the standard programs that the rest of the industry uses.

YOUR PHASE 1 CHEATSHEET

1

SKILLS TO LEARN:

- [ ] User Experience Design
- [ ] Web Design
- [ ] HTML
- [ ] CSS
- [ ] Git



2

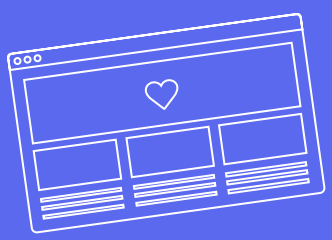
SOFTWARE TO TRY:

- [ ] Text Editor
- [ ] Adobe Photoshop
- [ ] A wireframing tool such as Balsamiq or Axure

3

YOUR PHASE 1 GOAL:

- [ ] To build, design, and launch your own portfolio website



HOW TO LEARN SKILLS

The great news is that these days there are SO many different ways to learn tech skills. Here are a few of our favorite resources for learning HTML, CSS, and the basics of Web Design:

🔗 CODESCHOLARLY BOOTCAMP FREE

Take the free Coding Bootcamp to learn how to talk like a techy,

🔗 CODECADEMY HTML & CSS FREE

This course teaches you the basics of HTML and CSS, as well as how to structure and style a webpage.

🔗 LEARN TO CODE HTML & CSS - SHAY HOWE FREE

This easy-to-understand and comprehensive guide helps beginners learn the fundamentals of HTML and CSS, as well as common elements of front end design and development.

🔗 HTML & CSS, DESIGN & BUILD WEBSITES BY JON DUCKETT

This book will teach you how to read and write HTML and CSS, structure and design web pages, prepare media for the web, and more.

🔗 TRY GIT FREE

This 15-minute interactive tutorial will teach you all the basics of how Git works and how to use it.

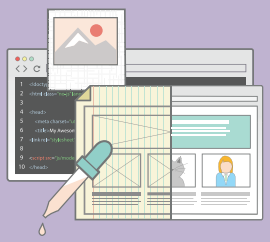
WEB DESIGNER BLUEPRINT

AVERAGE SALARY: $61,000

YOUR BLUEPRINT IS STRUCTURED AROUND

UX AND WEB DESIGN

HTML AND CSS



WEB DESIGNER APPRENTICESHIP

DOES YOUR CREATIVITY DRAW A CROWD?

You love making new things, from inventions to craft projects to delicious dishes, and you (not-so-secretly) dream of sharing them with the world. After all, does it exist if no one ever sees it?

MEET YOUR BLUEPRINT.

The Web Designer Blueprint includes everything you need to get started in web design in just three months. We'll take you from Photoshop to basic web tech to freelance domination--and everywhere in between.

YOU'LL LEARN:

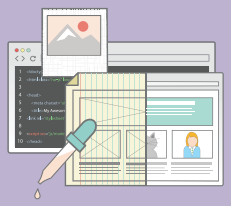
- The artistic principles of web design including color theory, grid systems, and typography
- How to use industry standard design software such as Photoshop
- How to design and code simple websites (including your very own portfolio website!) using HTML and CSS
- How to crush it as a freelancer, including how to find your first client and how to determine what to charge them

FRONT END DEVELOPER BLUEPRINT

AVERAGE SALARY: $92,000

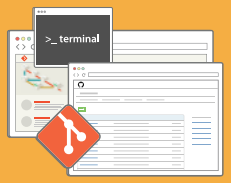
YOUR BLUEPRINT IS STRUCTURED AROUND

HTML AND CSS



RESPONSIVE WEB DEVELOPMENT

GIT SAFARI



JAVASCRIPT & JQUERY

WHY SURF THE WEB WHEN YOU CAN BUILD IT?

You love getting your hands dirty, you're always asking pesky questions like why and how, and you see beauty in the functional: a finished product that works just as it should. Most of all, you're ready for a career that rewards hard work.

MEET YOUR BLUEPRINT.

The Front End Developer Blueprint is a program that includes everything you need to start building and coding websites in just three months! We know what you need to know, and we're here to help you learn it.

YOU'LL LEARN:

- How to design and launch websites (including your very own portfolio!) using HTML and CSS
- How to turn static, old-fashioned sites into dynamic, responsive ones with CSS3, Flexbox, and Bootstrap
- How to use industry-standard version control Git via the Command Line
- How to build fun and engaging user experiences with the programming language JavaScript

WHAT YOU SHOULD KNOW

At this point, you should be able to create a basic website using your design, HTML, and CSS skills. This is a big accomplishment! You can create sites for other people at this point, or just to highlight your own skills.

SKILLS WEB DESIGNERS NEED

1

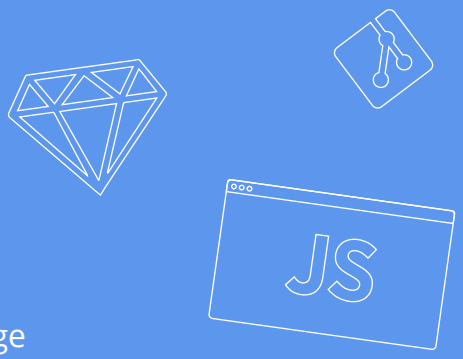
MUST-HAVE SKILLS:

- [ ] Typography
- [ ] Color Theory
- [ ] Branding
- [ ] Responsive Design
- [ ] UX Design (including how to create wireframes, mockups, mood boards, etc.)
- [ ] HTML
- [ ] CSS

2

OPTIONAL SKILLS:

- [ ] JavaScript
- [ ] jQuery
- [ ] Git
- [ ] Sass or LESS
- [ ] Web programming language such as PHP, Ruby, or Python



3

SOFTWARE:

- [ ] Adobe Photoshop and/or Sketch
- [ ] Adobe Illustrator
- [ ] Text Editor
- [ ] Wireframing software such as Balsamiq or Axure
- [ ] Command Line

SKILLS DEVELOPERS NEED

1

MUST-HAVE SKILLS:

- [ ] HTML
- [ ] CSS
- [ ] JavaScript & jQuery
- [ ] Git + Github
- [ ] A back-end programming language such as PHP, Ruby, or Python
- [ ] MySQL and/or another database querying language



2

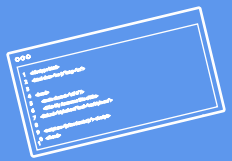
OPTIONAL SKILLS:

- [ ] Sass or LESS
- [ ] Responsive Design
- [ ] Photoshop and/or Sketch
- [ ] UX Design
- [ ] Ruby/Ruby on Rails
- [ ] PHP/WordPress
- [ ] Agile best practices
- [ ] Object Oriented Programming


3

SOFTWARE:

- [ ] Text editor
- [ ] Command Line
- [ ] Adobe Photoshop and/or Sketch




THANKS FOR JOINING US.

Feel free to email us with any questions at
info@codescholarly.com