**Front-End Fact Sheet:**

Here are some quick facts that can help you evaluate whether Front-End Programming is the right path for you!

**What are the different types of Front-End developers?**

* There really isn’t a clear distinction between different types of Front-End developers. Some might happen to do more of one kind of task than others, but all Front-End developers need the same general body of skills and knowledge.
* Design focused developers create sites that look and feel intuitive to the user. They spend most of their time writing HTML and CSS.
* Application focused developers use frameworks to create logic that powers interesting interactive user experiences on websites. These developers most often use JavaScript.

**What are the essential Front-End programming skills?**

* Empathy - The ability to see the thing you’re creating from your users’ perspectives. Remember, you aren’t developing a website for yourself, you’re creating it for other people.
* Breaking down problems - Breaking problems into manageable chunks is an essential problem-solving skill. You need to think about communication, design and architecting an application. In order to do all of these tasks, you need to be able to step back and plan out how you’ll turn a seemingly monolithic task into logical and reasonable chunks.

**What are important concepts Front-End programmers need to know:**

* Front-End developers focus on parts of the web that users see and interact with.
* Front-End developers access and present data from multiple sources on the internet through Application Program Interfaces (APIs).
* A Front-End developer is someone who builds the interface on an applications that runs on a variety of devices. These interfaces should responsively transition from mobile to tablet to desktop.
* Every web page and web app is created with HTML, CSS, and JavaScript. All of the images, buttons, transitions, animations, drop-down menus, contact forms, and styling elements are all controllable using these technologies.

**What are some Front-End Frameworks Front-End developers use?**

* Applications are built on top of frameworks like Angular.js, Ember.js, Backbone.js, Knockout.js, React.js and Polymer.js. There are similarities and differences, however if you can pick up one, you can pick up all of them.
* Front-End developers use frameworks like Angular.js and Ember.js to organize data using industry-standard design patterns which help synchronize user actions and data with a web server.

**How will I know if I'm ready for a path in Front-End programming?**

If you've completed the previous lessons in this Nanodegree Program, you already have an understanding of HTML syntax, and the structure of HTML, CSS and Python. This will enable you to pick up Javascript libraries like JQuery and to use CSS frameworks like Bootstrap. While Javascript and Python are different, they share similarities as well: functions, variables, data-types, loops, and conditional statements are used by both languages.

Check out our Blog post [**"Web Dev Building Blocks"**](http://blog.udacity.com/2015/03/web-dev-building-blocks-need-to-know-where-to-learn.html) for front end building blocks and where to learn them.

What about APIs you ask? Read more about APIs in this [**blog post**](http://sproutsocial.com/insights/what-is-an-api/).

Front-End Developers specifically make calls or requests to APIs but most often Back-End Programmers are the people who build these APIs.

NEXT

**Back-End Fact Sheet**

Here are some quick facts that can help you evaluate whether Back-End Programming is the right path for you!

**What are the different types of back-end programmers?**

* Back-end programmers are differentiated by the languages and technology stacks they use. Back-end programmers can be generalists or they can specialize in areas like devops/infrastructure, internal tooling, api design, and database administration.

**What are essential back-end programming skills?**

* Software architecture- The ability to translate project requirements into technical specifications.
* Software deployment- The ability to make a software program ready and available for public use on the web.

**What are the differences and similarities of back-end programming languages?**

* Considering the broad range of tasks that a back-end programmer can perform, it makes sense that there are a large range of programming languages and software that can be employed to perform these tasks. These can operate on a variety of levels, and it can be useful for a programmer to be comfortable with a variety of tools to accomplish their tasks. Different languages are typically associated with different stacks, although some components are interchangeable between stacks. For example Ruby is closely associated with the Rails framework and ActiveRecord Object Relational Mapping (ORM), while Python has several competing web frameworks and ORMs. The technologies used by back-end programmers tend to stick around longer than those used on the front-end because rearchitecting an application is a major undertaking. As a result back-end technologies change more slowly.

**Why might a back-end language be more suitable for certain projects?**

* Languages differ in how quickly a programmer can produce a working product, but ease of development generally comes at the cost of performance and maintainability. Languages like Java and Go are popular among large teams and large projects because they enforce code organization and structure. Languages like Python, Ruby and Javascript allow rapid development and easy prototyping, but they don’t necessarily scale to heavy loads as easily. This implies cultural differences between the communities around different languages, and in the teams that use the different languages. Java in particular is used at larger enterprises. PHP tends to be seen in legacy applications.

**What are the important concepts back-end programmers need to know:**

* The concept of a **stack**: The ‘stack’ refers to the combination of technologies a developer uses to implement a particular project. One of the first major stacks was the LAMP stack (Linux, Apache, MySQL, and PHP). Over time more stacks have emerged that use different components. For example nginx is a popular alternative to Apache for the webserver component of the stack. There are many SQL and nonSQL alternatives to MySQL in use today, and there are even more choices of programming language and framework. Today the stack might also include the front-end technologies used by an application.
* The concept of **full-stack**: A programmer who is competent in everything from front-end to back-end programming, including all elements of the stack. In practice very few people are truly experts in every component of a large application. It’s reasonable for a single programmer to be familiar with the entire stack, and to implement small applications and prototypes single handedly. A full-stack programmer should be able to research the topics they need to work with any part of an application despite not being a daily user of any one piece of the stack.

**How will I know if I'm ready for a path in back-end programming?**

If you've completed the previous lessons in this Nanodegree Program, you already have an understanding of Python and through the movie website project you learned how Python (the back-end) and HTML/CSS (the front-end) interact. Your next step is to learn how to store information in databases.

Check out our blog post [**"Front-End vs. Back-End vs. Full-Stack Web Developers"**](http://blog.udacity.com/2014/12/front-end-vs-back-end-vs-full-stack-web-developers.html) to learn more about the differences between front-end and back-end programmers.

NEXT