Web Development

Concepts For Everyone

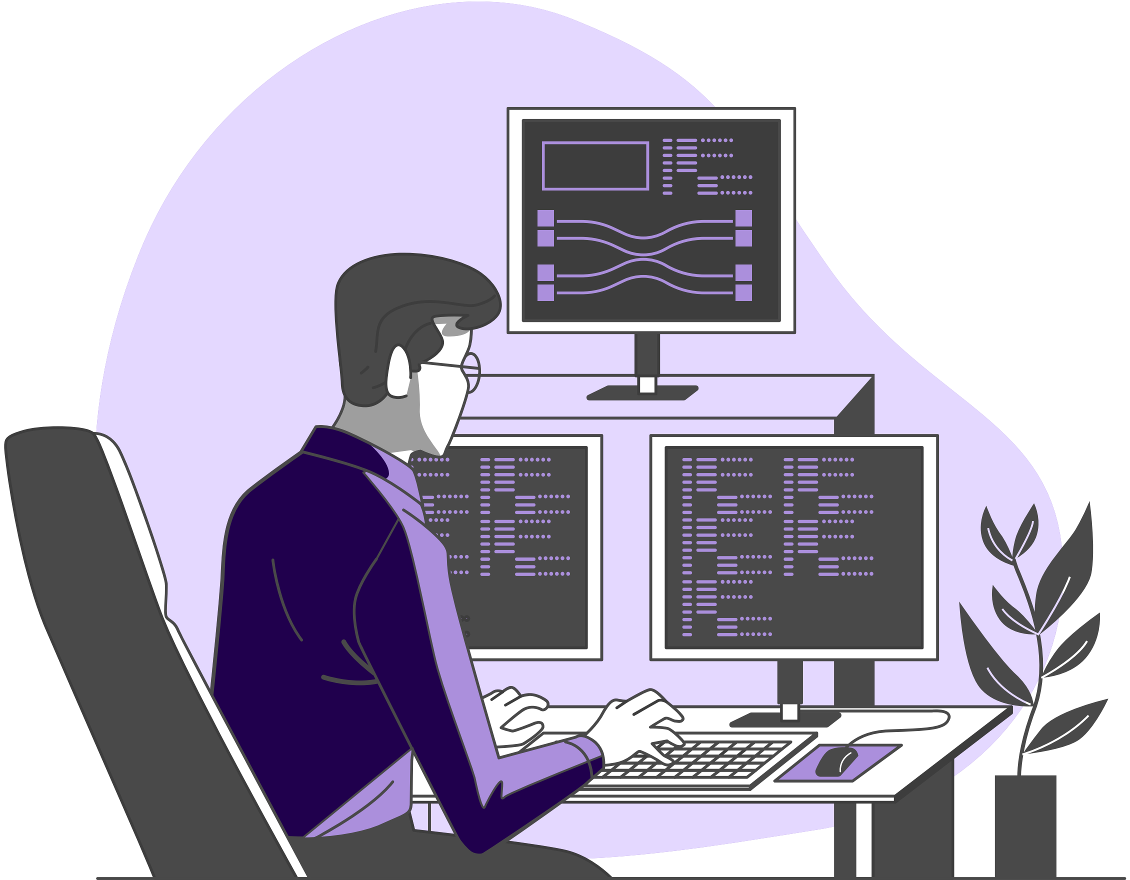
Chapter

01

Introduction

Course Note

What to expect



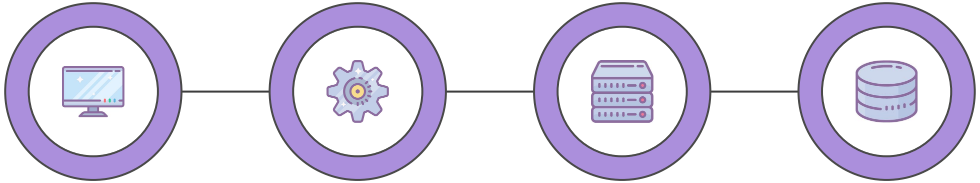
Course Tools

Here is what I will be using

Chapter

02

The Full Stack



Full Stack

The “Full Stack” is a term used to describe the system of applications that speak to each other across a networks, often comprised of a Front End, Client, Server, and Database.

...

Front End

The Front End is everything we can see on a Web Page. It is a combination of layout and the style of all the elements on the screen. The Front End is made of up HTML and CSS.

Client

The Client is the logic that makes a webpage “smart”. It handles User Interactions and the flow of data between the User and the Server. The Client is what turns a Web Page into a Web Application.

Server

The Server handles data exchanges between Clients and Databases. Users have the Client on their device, so there are many clients. Those clients talk to the same server per application.

...

Database

Holds onto data for a web application long term. The browser can only hold on information for a limited time. The data in Databases are persistent. The data does not go away if the database or application are “turned off”.

Chapter

03

Version Control

Version Control

Technology that supports versioning of in progress software. It is more advanced than just about saving files. Version Control gives developers flexibility to safely experiment and collaborate with other developers.

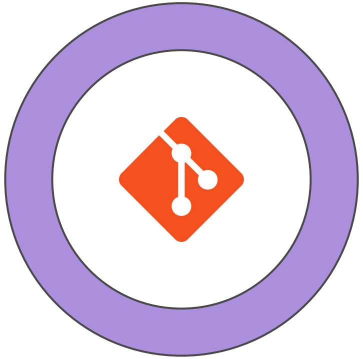
...

Developers can work together!

With Version Control, developers can:

- Developers can work on the same code base at the same time,
- Isolate “branches” of their code to try out and solidify ideas,
- Use the technology to deploy to multiple locations.

Version Control is the backbone of modern multi-developer development.



Git

Git is the most popular version control technology in 2021. Pairs with popular platforms such as GitHub, Bitbucket, and GitLab. Allows for version branching.

Commit

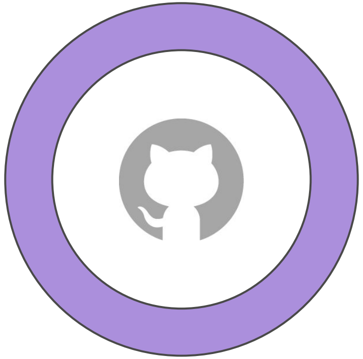
A commit is a “snapshot” of a code base that a developer can make when they choose. They typically do so as they are working through an idea and get to a point where part or all of their idea is established and functional in the code.

Branch

A branch is a copy of the code base of a commit. The branch is a copy of the last commit, and all the commits that came before it. It starts a new “timeline” of events that allows the developer to work with the code in isolation.

Merge

A merge is when a branch is introduced (or reintroduced) to another branch. Often, this happens when a branch of new functionality is completed and ready to be brought back into the main development branch and ready to use.



GitHub

GitHub is a popular Git cloud platform used by individual developers and companies. On GitHub, developers can store, share, and collaborate with other developers, and a whole lot more. Purchased by Microsoft in 2018.

Chapter

04

Programming Languages

Runtime Environments

A Runtime Environment is where an application, which is the result of writing code, is executed. Node and the Web Browsers like Chrome are both examples of JavaScript runtime environments that have a different purpose.

Example of where some Languages “Run”

Java

Spring (Back End)
Android (Mobile)

Python

Flask (Back End)
Django (Back End)

Javascript

Chrome (Front End)
Node/Express (Back End)

Swift

iOS (Mobile)

Ruby

Ruby on Rails (Back End)

C# (“C Sharp”)

.NET (Back End)
Unity (Game Engine)

Types in Programming

One of the biggest differences in programming languages is how that language handles “types”. Types in programming are descriptions of declared information. Some examples of basic types in programming would be: **Numbers**, **Strings**, and **Booleans**.

Programming Types: Weak vs Strong

Weak

Can change Type

A unit of information, often called a “Variable” is allowed to change from one type to another within a computer program.

Strong

Cannot change Type

A Variable is NOT allowed to change its type in the program. The variable is “locked” into a type and will cause an error in the application if a change is attempted.

Programming Types: Dynamic vs Static

Dynamic

Runtime Errors

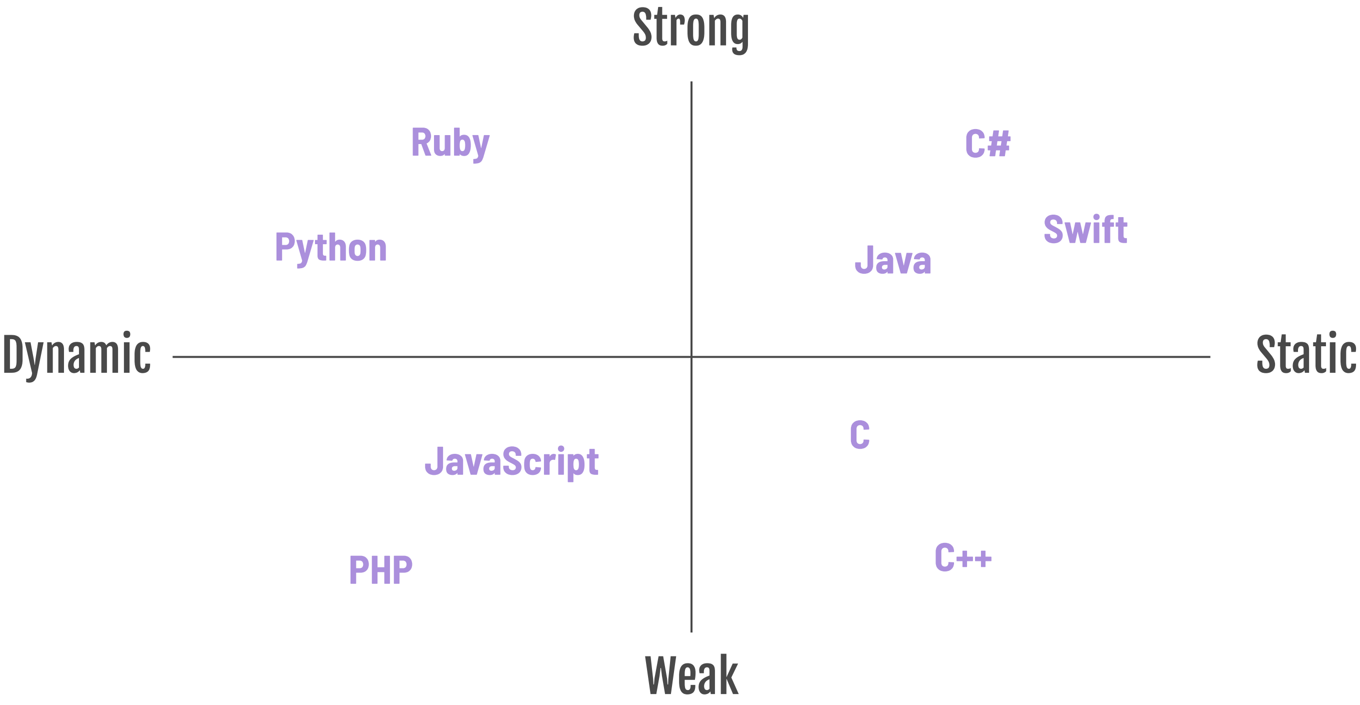
Errors are surfaced when an application is running. As a potential advantage the developer is allowed to iterate and "discover" their application a bit more.

Static

Compile Time Errors

Compile time error checking of types while the application compiles. Giving the developer a "heads up" that their logic may be flawed.

Strong vs Weak | Dynamic vs Static



Strong

Ruby

C#

Python

Swift

Java

Dynamic

Static

C

JavaScript

PHP

C++

Weak

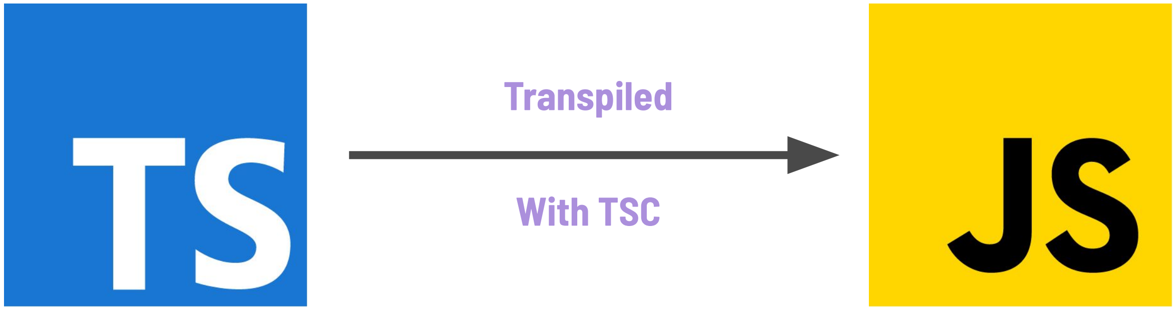
Supersets

A Superset language, is one that contains all of the features of a given language, but then is expanded and/or enhanced to include other features as well.

Superset Example – TypeScript

TypeScript is a Superset of JavaScript developed and maintained by Microsoft. It is a strict syntactical superset and adds optional static typing to the language. Its feature implementation is configurable and can be intermixed with JavaScript.

TypeScript turns into JavaScript



Transpiled

With TSC

TypeScript

JavaScript

Markup Languages

A Markup language is a system for annotating a document in a way that is visually distinguishable from the content. It is used only to format text, so that when the document is processed for display, the markup language does not appear. Examples include HTML and XML.

SDKs

XXX

Chapter

05

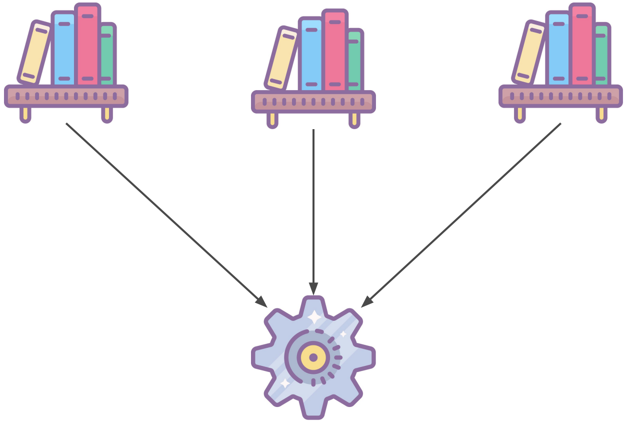
Frameworks and Libraries

(Code) Library

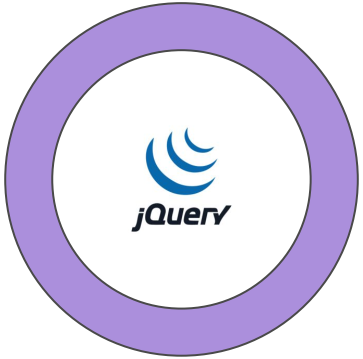
A Library is a collection of either Front End or Back End code that is “un-opinionated”. Developers bring a library into their project and decide when to use the code supplied by the Library.

Libraries support our Applications!

Supporting Libraries



Our Application



(Code) Library Example – jQuery

Wildly popular Library that rounds its rise before the popularization of technologies like Angular and React. jQuery makes interacting with “The DOM” much more manageable while also simplifying client to server interactions.

(Code) Library Example – Bootstrap

jQuery works with the Client JavaScript Code - But there are also libraries that support the Front End HTML and CSS. Bootstrap is a popular CSS library, that also has some functionality pre-developed in JavaScript.

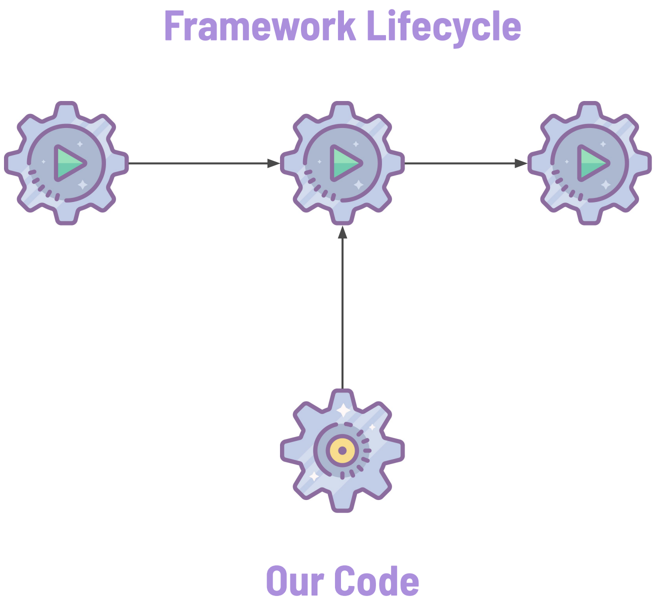
(Code) Library Example – Chart.js

A popular charting Library for JavaScript. It makes what would be a difficult combination of styling, layout, and data positioning and turns it into an easy to manage ordeal. Saving developers loads of time in creating graphs and charts.

(Code) Framework

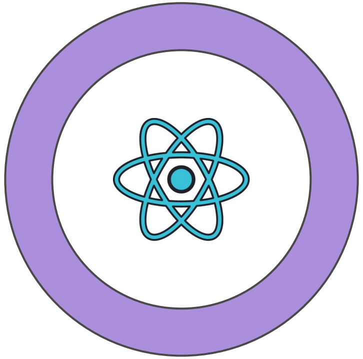
A Framework is a collection of either Front-End or Back-End (Server) code that is “opinionated”. Developers develop their code to work with the frameworks life cycle of operations that is provided.

Our code is plugged into the Framework Lifecycle!



(Code) Framework Example – Angular

Developed by Google, Angular is a Front-End framework that utilizes the TypeScript Superset. It is a opinionated framework that has many architectural decisions already made for developers.



React

A JavaScript library developed by Facebook. While it has framework like features, it is not fully featured, and therefore considered a library. There are still some architectural decisions that are left unmade.

AngularJS and Angular

AngularJS and Angular are NOT the same framework. AngularJS was the initial version of Angular and uses regular semantic versioning. Angular, started as Angular 2, they skipped Angular 3?, Angular 4, and so on. Angular 13 is set to release in November 2021.

AngularJS versus Angular

AngularJS

Written using JavaScript.

Reduces development time, but takes longer to load.

Angular

Written using TypeScript.

Provides better structure and faster loading times.

Chapter

06

Client/Server Communications

HTTP Protocol

HTTP is a communication standard for between Client and Server over the internet. It is an important standard as not all technology used to build Client and Server applications are the same, but all of those technologies have a solution for HTTP built-in to meet the standard.

HTTP Methods

Predefined verbs used to describe the interaction the Client wishes to engage in with the Server, often called a “Request”. While there are 39 different possible verbs, the 4 most common are: “**GET**”, “**POST**”, “**PUT**”, and “**DELETE**”.

APIs

An API or “Application Programming Interface” is a declaration of how an application can communicate with another application, library, or framework. In the context of the web, a Web API, is a collection of HTTP methods a Client can use to communicate to a Server/Service.

REST APIs

A REST API is a collection of six architectural constraints. In common language between developers, REST APIs point to a collection of HTTP methods that are **resource based**.

Web Sockets

The WebSocket API makes it possible to open a bi-directional, interactive session between a client and server. The browser and server can both “broadcast” messages that the other listens for. Websockets are the backbone for applications that have chat functionality.

GraphQL

Publicly released in 2015 by Facebook, GraphQL describes itself as a “Query Language for APIs”. GraphQL allows developers to create more precise Client to Server calls, retrieving only data the developer needs at the time of the call, rather than a predefined block of data with REST.

RPC

Similar to REST, RPC is an API Interaction standard. However, instead of being resource based like REST, RPC focuses on **execution of processes**. It’s like saying “execute this instruction”, rather than “give me this resource”.

SOAP

The Simple Object Access Protocol, is an older API protocol. It uses an XML schema and Web Services Description Language (WSDL) to define what data needs to be exchanged and where it should go.

Chapter

07

Authentication & Authorization

Authentication (AuthN)

Upgrading an application to allow it to collect information and validate **WHO** the user is.

Authorization (AuthZ)

In working with Authentication, after the application determines who the user is, Authorization determines **WHAT** the user is allowed to do in the application.

OAuth and OpenID Connect

OAuth is a system which grants third-party websites limited access into user accounts. OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol.

Chapter

08

Back-End Frameworks

Back End Frameworks

Back End frameworks give developers the tools to create applications that are capable of receiving requests over the internet, turning those applications into servers. They generally come in one of two scales, as **Micro-Frameworks** or regular **Web Frameworks**.

...

Microframeworks

Micro frameworks are a framework in the idea that they typically have a lifecycle that a developer builds on top of, but leaves some decisions and implementations to the developer.

Microframeworks

Java

Spark
Micronaut

JavaScript

Express.js

Python

Flask
Bottle

PHP

Phalcon
Lumen

Ruby

Sinatra

Modules and Packages

Microframeworks tend to be as small as possible by design. But there are still some common problems that developers have solved, that other developers do not want to create custom solutions for. Packages are common solutions that are bundled up and made available to other developers.

Web Frameworks

Web Frameworks were the major player in creating more modern web experiences. They handled interactivity and customized experiences before JavaScript matured and made the web as we know it today. Web Frameworks like C#'s .NET and Java's Spring still remain an incredibly popular option even today.

Web Frameworks commonly solve:

01

Accounts

AuthN, AuthZ, and Roles for the application.

02

Database Abstraction

Leveraging ORMs (Object Relational Mapping)

03

Input Validation and Sanitation

Makes sure our data is clean!

04

Leverages Template Engines

The Back End framework renders “intelligent” HTML.

Web Frameworks

Java

Spring
Struts

JavaScript

Nest.js

Python

Django

PHP

Laravel
Symfony

Ruby

Ruby on Rails

C#

.NET

Template Engines

Template Engines are a Web Frameworks way of rendering HTML. Originally, the Client Logic layer was much “thinner”, in that the logic of an application was housed primarily in the Server. Templates created hooks for interaction and data to flow between the front end and the back end. Each framework has their own preferred Template Engine.

Chapter

09

Databases

SQL – Relational Databases

Data that is organized into Rows and Columns on a Table. The entries in the table can have relationships to other tables and the developer can choose to structure ways to “query” or group that data.

In SQL, Data is in Tables!

Dog Table

ID	Name	Breed	Age
1	Baxter	French Bulldog	4
2	Ori	Shiba Inu	2
3	Jeffery	Boxer	10

In SQL, Data is in Tables!

Dog Table

ID	Name	Breed	Age	Mood Id
1	Baxter	French Bulldog	4	2
2	Ori	Shiba Inu	2	3
3	Jeffery	Boxer	10	1

Mood Table

ID	Mood Name
1	Happy
2	Sleepy
3	Playful

Dog Table and Mood Table, joined and filtered!

ID	Name	Breed	Age	Mood Id	Mood Name
1	Baxter	French Bulldog	4	2	Sleepy

Relational Database Example – PostgreSQL

PostgreSQL, known as “Postgres” is a free and open-source relational database management system that emphasizes extensibility and SQL compliance.

NoSQL – Document Databases

Data is “schemaless” and organized into “JSON-like” objects. Instead of defined rows of needed information for an entry, entries can be more free-form. Entries can contain different information from other entries in the ‘collection’.

Document Database Example – MongoDB

MongoDB is a source-available cross-platform document-orientated database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

In-Memory Databases

Databases that store information in Computer Memory, rather than on a Disk or Solid State Drive. Makes the data extremely quick to access, but the data is usually not persistent between sessions.

...

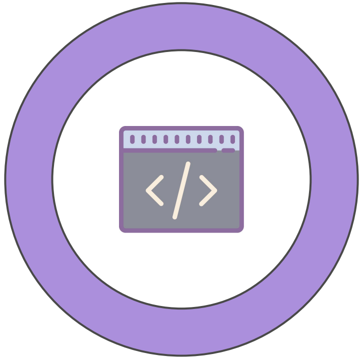
In-memory Database Example – Redis

Redis is an in-memory data structure store, used as a distributed, in-memory key-value database, cache and message broker, with optional durability.

Chapter

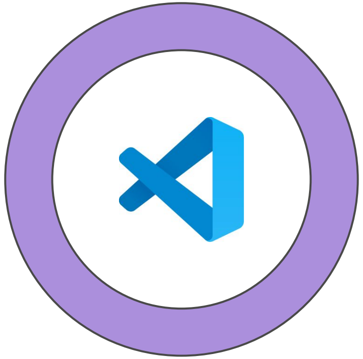
10

Developer Tools



Code Editor

A Code Editor is an application that is used by programmers to develop applications. More than a simple text-editor, a code editor is optimized for programming languages rather than spoken languages.



Code Editor Example – Visual Studio Code

Visual Studio Code or VSC is a popular code editor used by many web developers. Created by Microsoft, it is popular since it includes not only support for debugging and a host of other features, but it is also upgradable through an integrated marketplace.

IDEs

An Integrated Development Environment is an application that facilitates software development for programmers. Usually includes a code editor, build tools, and debugger.

IDE Example – IntelliJ Idea

IntelliJ Idea is an IDE for specifically the Java Programming language. It is developed by Jetbrains and streamlines the process of creating Java applications.

Console

A low-level command line interface included in Operating Systems that allow users to execute more discrete tasks than a regular computer user. Programmers use the console regularly for a wide range of software development related tasks.

Package Managers

Package managers are often language or framework specific tools that manage the retrieval and organization of supporting code needed by developers that are created by other developers.

Build Automation

Build Automation tools are supporting tools that can execute a number of different tasks for developers creating application. Examples of tasks that could be completed are code minification and code transpiling.

ORMs

The common programmer version of Object Relational Mappers (ORMs) is a library that allows you to programmatically change data in a database using the programming language rather than the query language.

Postman

Postman is an application used for testings APIs. It is a Client application that tests HTTP requests using a graphical interface. Great for Back End developers who want to test their APIs without building a Front End, or looking for something more than a console.

Chapter

11

Developer Concepts

Software Design Patterns

Software design patterns are a general, reusable solution to common problems that need to be solved in programming. They are abstract ideas that are programming language agnostic.

Software Testing

Software Testing is the process of evaluating and verifying that a software or application does what it is supposed to do. This is often done with supporting technologies and test-specific code that can be executed as the application is developed.

SOLID Principles

SOLID is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible, and maintainable.

SDLC

The Software Development Life Cycle (SDLC) is the process the software industry uses to create software. It typically includes phases of development such as planning, creating, testing, and deploying applications.

SPAs

Single Page Architecture (SPAs) is a web application style that is common with the advent of React and Angular. Instead of using an older method of having the Server serve entirely new webpages as a user navigates a site, the Client controls what is shown on the screen.

Algorithms

An Algorithm is a series of programmed instructions. Typically, algorithms are executed to perform actions like collect data or interactions, transform data, and produce data or visual outputs.

Code Standards

Code Standards are a set of guidelines for a given programming language that recommend programming style, practices, and methods for each aspect of a program written in that language.

12 Factor Application

A methodology for the delivery of Web Applications. It can be applied to applications written in any language and that use any combination of services.

Chapter

12

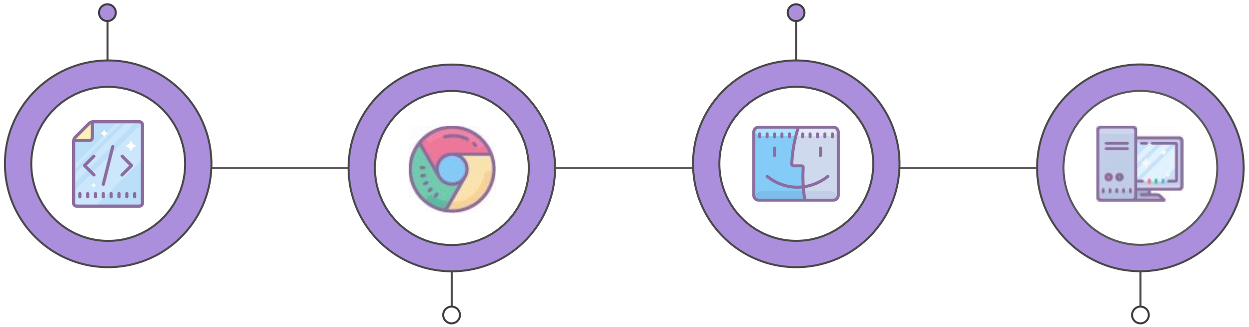
Cloud Platforms

How Applications Run

Code creates Applications.
Applications run in Runtime Environments.
Runtime Environments run on Operating Systems.
Operating Systems run on Hardware.

How our Application actually runs!

(An Example of a Client Application Running)



Our Application runs on...
(Client.js/App.js, etc.)

A Runtime Environment,
That runs on...

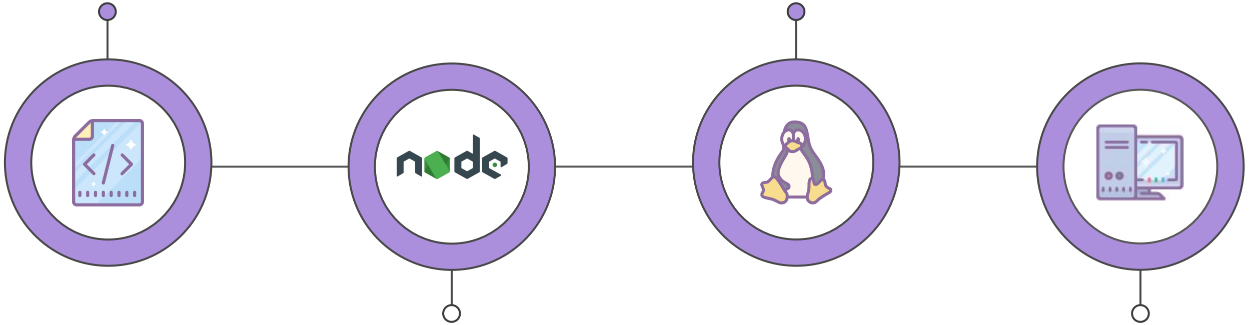
An Operating System,
That runs on...

Our Hardware!

How our Application actually runs!

(An Example of a Server Application Running)

Our Application runs on...
(Server.js/App.js, etc.)



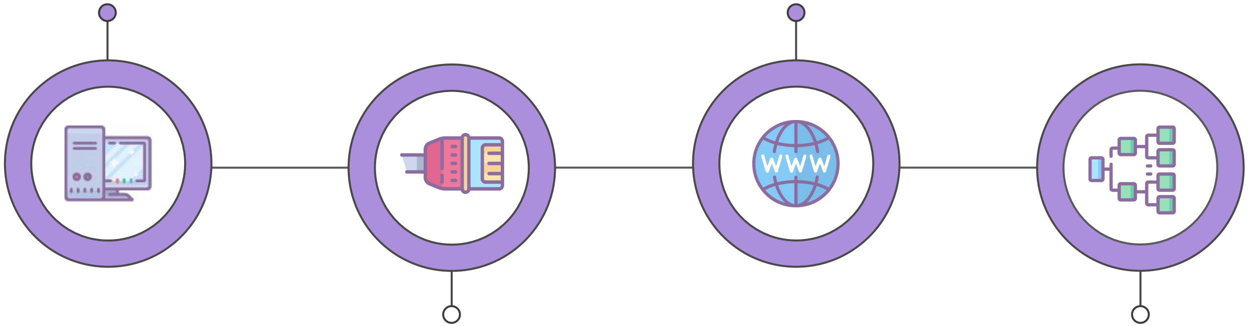
A Runtime Environment
That runs on...

An Operating System
That runs on...

Our Hardware!

How we get to Other Computers

(The simple way we get to the Internet!)



Finally, our Hardware,
Connects to...

Our Wifi or Wired connection
to our home Networks,
Which Connects to...

Our Internet Service Provider,
Which connects to...

Other Computers/Servers
Wired into the Internet!

Our Computer Hardware

In order to understand the Cloud, we need to understand the basics of how our computer works. Just the core parts. Our computer has a processor, memory (or RAM), and a hard drive. These core components leverage many other components in modern computers, but these are the core to discuss.

Key Hardware Components

Processor

Speed / Power,
Measured in GHz
and Cores

...

Memory

Short Term Memory,
linked to Speed.
Measured in MHz and GB

...

Storage Space

Storage Space,
Measured in GB/TB
and HDD or SSD

...

Cloud Platforms

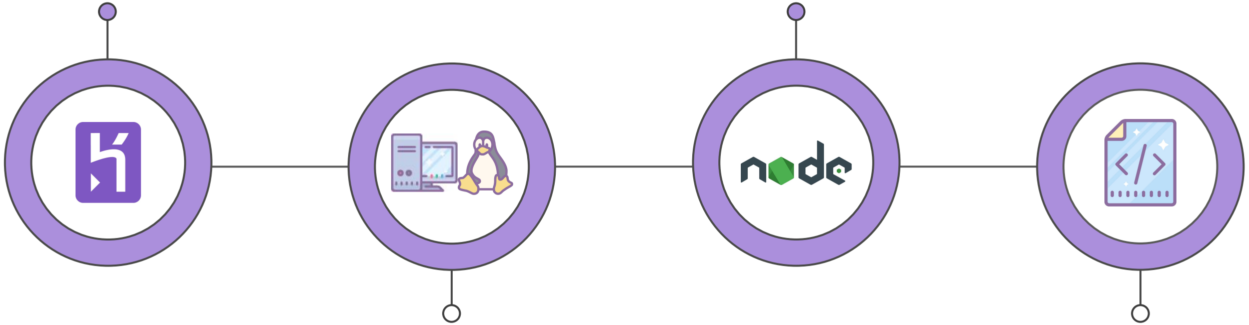
A Cloud Platform is a suite of services made available by a Provider. The most common service made available by Cloud Platforms is site and application hosting, and in recent years has expanded to be a wide range of infrastructure and application services.

Heroku – A Cloud Platform

Heroku is classified as a PaaS that enables developers to build, run, and operate applications entirely in the cloud. Its an awesome cloud provider for those learning development because of its simple setup.

Cloud Providers take care of things for us!

(That way, we can focus on the code!)



Heroku provides...

Hardware and an Operating System that runs...

A Runtime Environment that runs...

Our Application!

Amazon Web Services (AWS)

AWS features a MASSIVE list of services that developers and cloud engineers to pick from. Each service is designed to take a portion of an applications ecosystem. We can bite off on all or some services depending on what make sense for the project.

Serverless

Just a fancy term for choosing to use Cloud Services rather than writing a server “by hand”. The truth is, is that there is still a server, it’s just that the Cloud Services provider has written a portion of it for us, often providing a User Interface to plug in values.

Cloud Function

A Cloud Function is a Serverless concept where a service is enlisted to carry out a small piece of application functionality. It is accessible to the developer via API. As a developer, you decide the Programming Language you wish to use, but the rest of the hosting details happen behind the scenes.

Cloud Certifications

Cloud Providers offer certification paths for both developers and non-developers. They are scoped to the services they provide and often offer different levels. Lack of certification does not mean unskilled however. Some choose simply not to get them.

Chapter

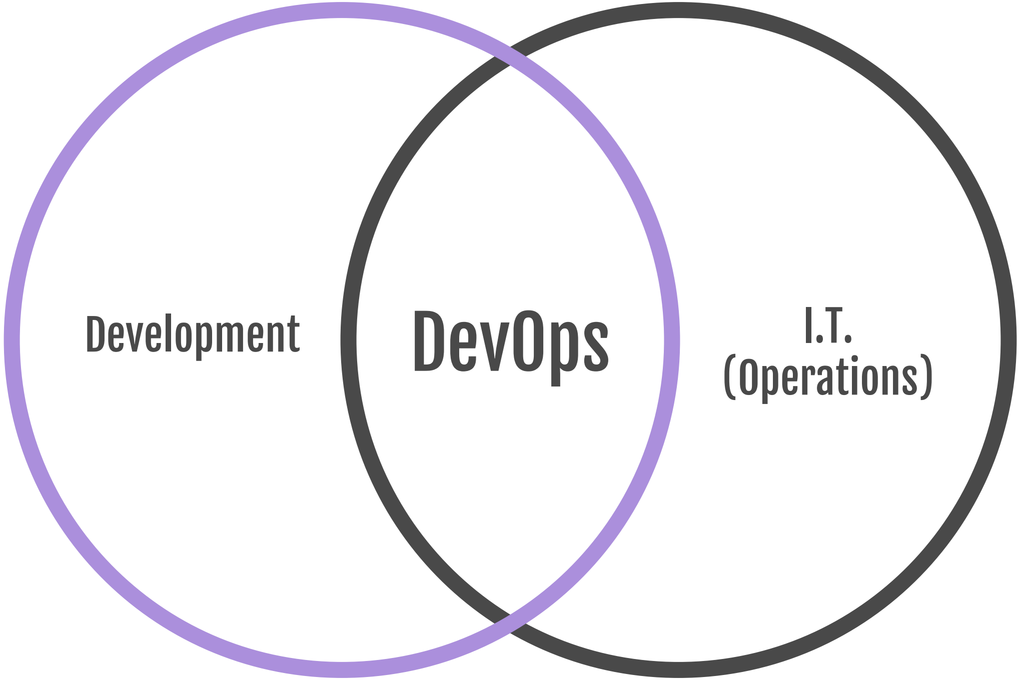
13

DevOps

DevOps (Development Operations)

DevOps is a set of practices that combines software development and IT (Information Technology) operations. It aims to shorten the systems development lifecycle and provide continuous delivery of high software quality.

...



Development

DevOps

I.T.
(Operations)

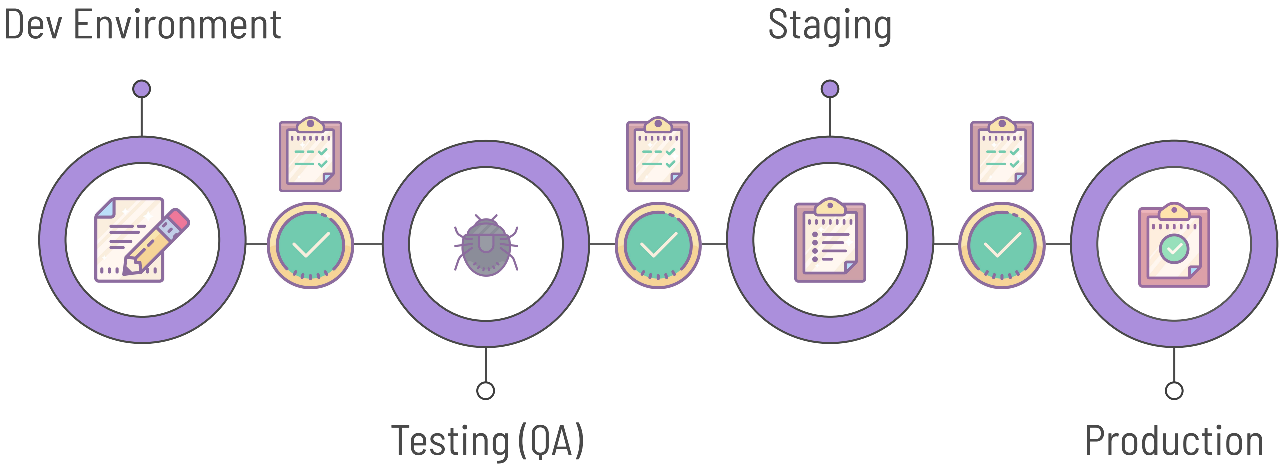
DevOps Goals

Release new code faster!
More responsive to business needs!
Better quality and uptime!

Continuous Integration (CI)

The automated building and testing of an application on each new commit. It is the practice of merging all developers' code to a composite of that code, several times a day.

Tested along the way



Dev Environment

Testing (QA)

Staging

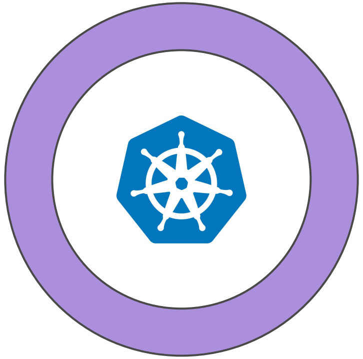
Production

Continuous Delivery (CD)

A software approach in which teams produce software in short cycles. As opposed to long drawn out period between when code is done and when it is actually put into production.

Containerization and Docker

Containerization is a way of packaging the code up complete with libraries, files, dependencies in an environment that is predictable and portable. Docker is a suite of technologies that accomplishes this.



Container Orchestration and Kubernetes

Container Orchestration is the process of automating application deployment, scaling, and management. In a nutshell, it is a way to manage sets of containers. Kubernetes was originally developer by Google, and often referred to K8 or Kube.

Continuous Monitoring

Continuous monitoring is the process and technology used to detect compliance and risk issues associated with an operational environment.

DevOps Maturity Model

While there are many versions of the DevOps Maturity Model, it is meant to be a model that describes a company's journey along the path of building a DevOps culture. Most models provide a step/progression layout to help identify where a company is, and what the next step may be.

Thank You!