play as a Backend

By Dilum De Silva

(Software Engineer at Circles.Life Singapore)

So, what we gonna know by the end of 3 weeks...

Week One	Week Two	Week Three
● Intro to ‘frontend vs backend’ ● Intro to Play as a backend ● Combining Play with front end tech (Concepts) ● What is MVC pattern ● Play setup and base play project structure	● What the heck is REST APIs? ● Concepts of REST APIs ● Developing REST APIs with play.	● Combining Play with a frontend (angular) and exposing an API (coding).

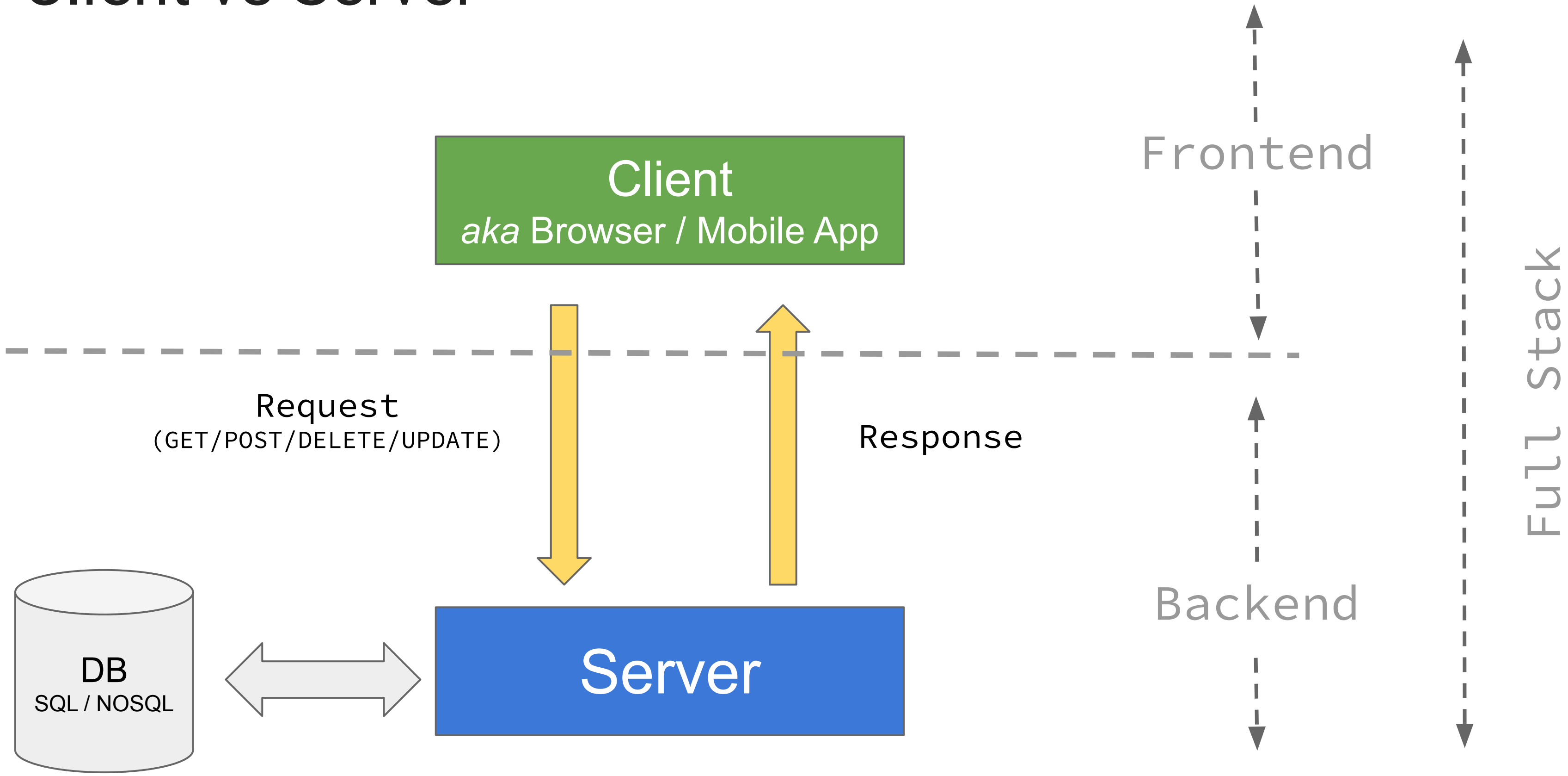
Frontend vs backend

Frontend vs Backend

Front-End
Back-End

Frontend (client) vs Backend (server)

Client vs Server



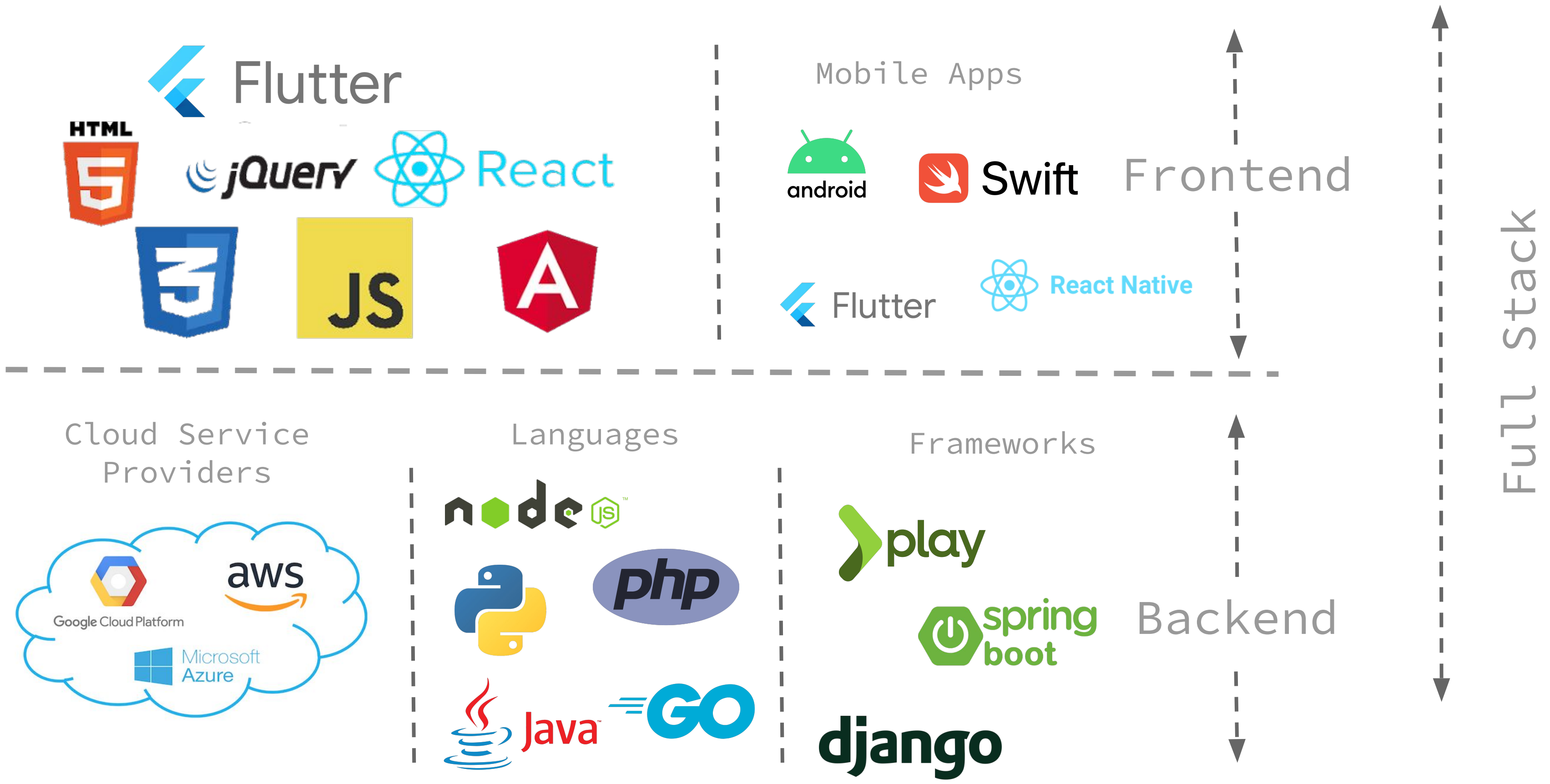
Client vs Server in Real World

Frontend

Backend

Full Stack

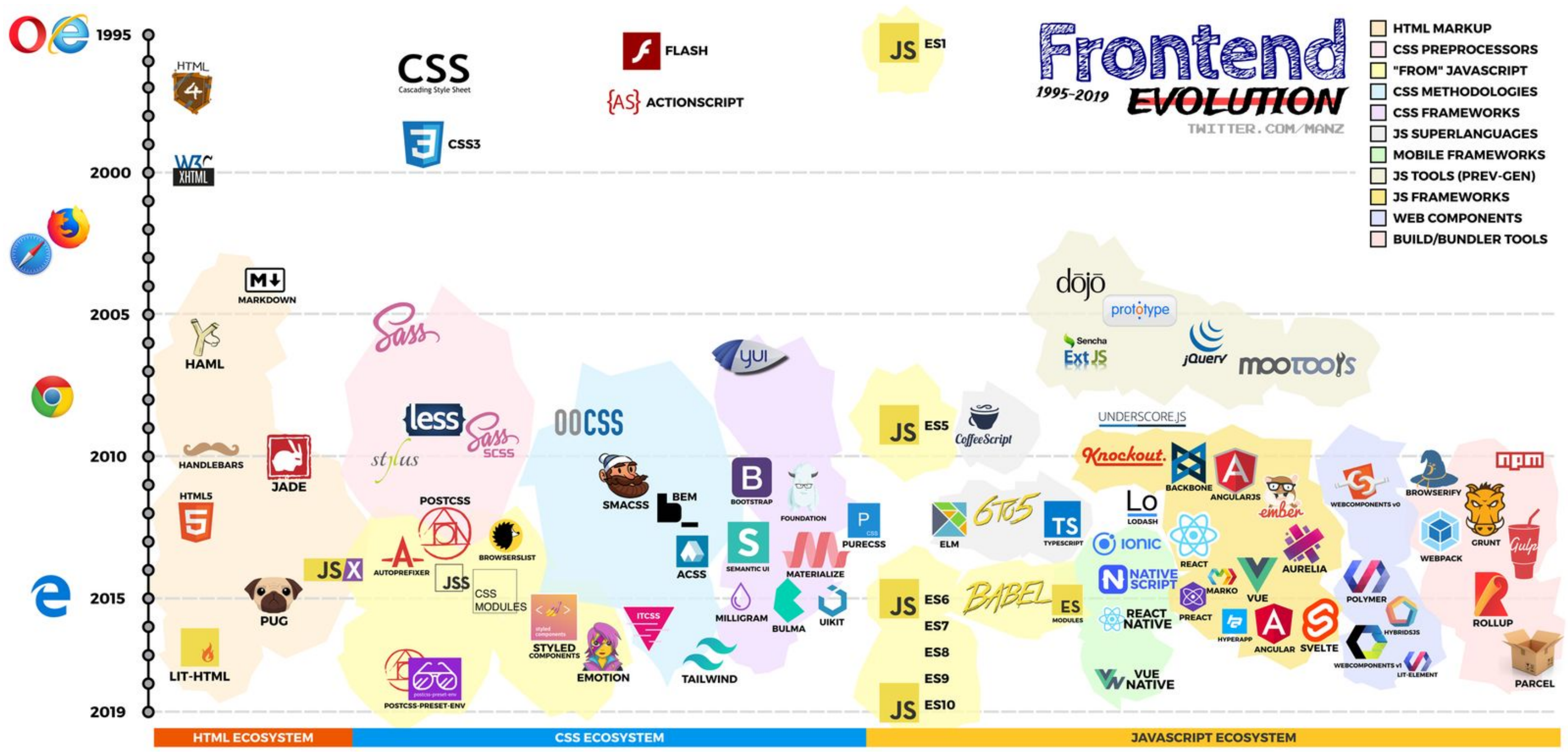
Client vs Server in Tech Perspective



Frontend Development

#client

Evolution of Client-Side Technologies



If you want to become a frontend developer...

Technologies/ Languages

- HTML
- CSS
- JavaScript
- CSS Pre-processors (Sass, Stylus...)
- JavaScript Libraries (e.g. lodash) and Frameworks (Angular, React, Vue)
- Build Tools (npm, Webpack, ...)

You'll work on ...

- JS-driven User Interfaces
- Re-usable UI Components with JS logic and CSS Styling
- Forms & Input Validation
- Backend Communication Channels
- UX Strategies (PWAs, Live Updates)

Less Relevant Technologies/ Languages

- Server-side Languages (e.g. Node, PHP)
- Databases/ Query Languages (e.g. SQL)
- Server Configuration

You'll NOT work on ...

- Server-side Business Logic (e.g. User Authentication, Order Handling)
- Automatic E-Mail Notifications
- Database Access

Backend Development

#server

If you want to become a backend developer...

Technologies/ Languages

- Server-side Languages like Node, PHP
- Frameworks like Express, Laravel
- Databases & Query Languages
- Partly: Server Configuration
- Basic HTML, CSS, JavaScript

You'll work on ...

- Server-side Business Logic (e.g. User Authentication, Order Handling)
- Automatic Notifications
- Data Validation
- Data Storage/ Database Access
- Scheduled Processes

Less Relevant Technologies/ Languages

- Advanced JavaScript & CSS
- JavaScript Libraries & Frameworks
- Build Tools (npm, Webpack)

You'll NOT work on ...

- Client-side Validation
- Complex User Interfaces
- Advanced UX Strategies (PWAs, ...)

It’s a matter of your choice and passion...

Frontend	Backend	Fullstack
Extremely Fast Development Technologies & the Ecosystem	Security stays important, ever more Data and User Interactions require “better” Algorithms and Processes	No/ Little Dependencies on other Developers, Full Understanding of the Complete Tech Stack
↓	↓	↓
Growing Demand in Developers	Growing Demand in Developers	Perfect for Small Companies & Freelancers
“Just the User Interface”	“No direct Connection to the User/ Customer”	“Jack of all Trades”

Ruby
console.log("Hi")
CSS
FRONT-END
VS
BACK-END
VAR
Hello world
PHP
LET
For (i = 0)
<HTML>
<code>
Const

Let's dive Into

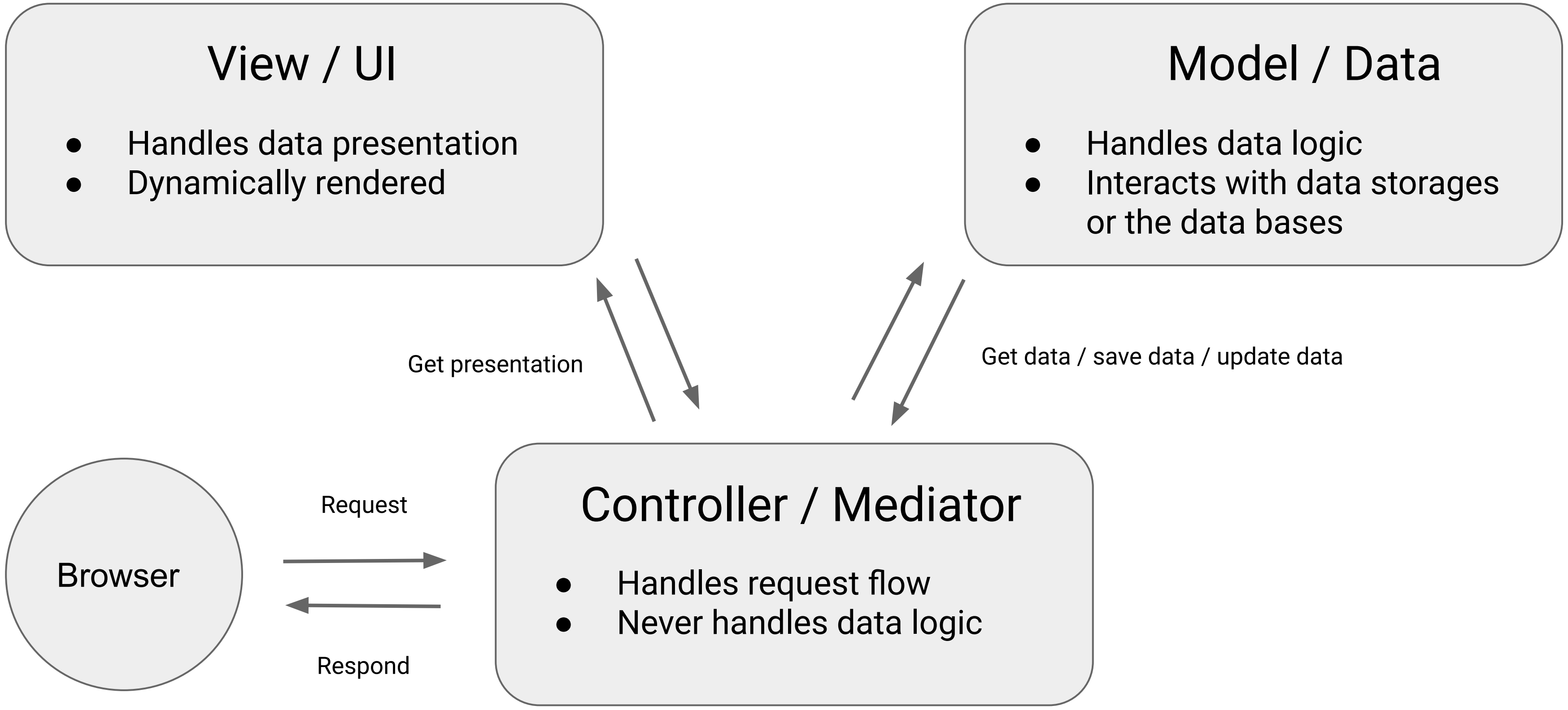
play

What is play ?

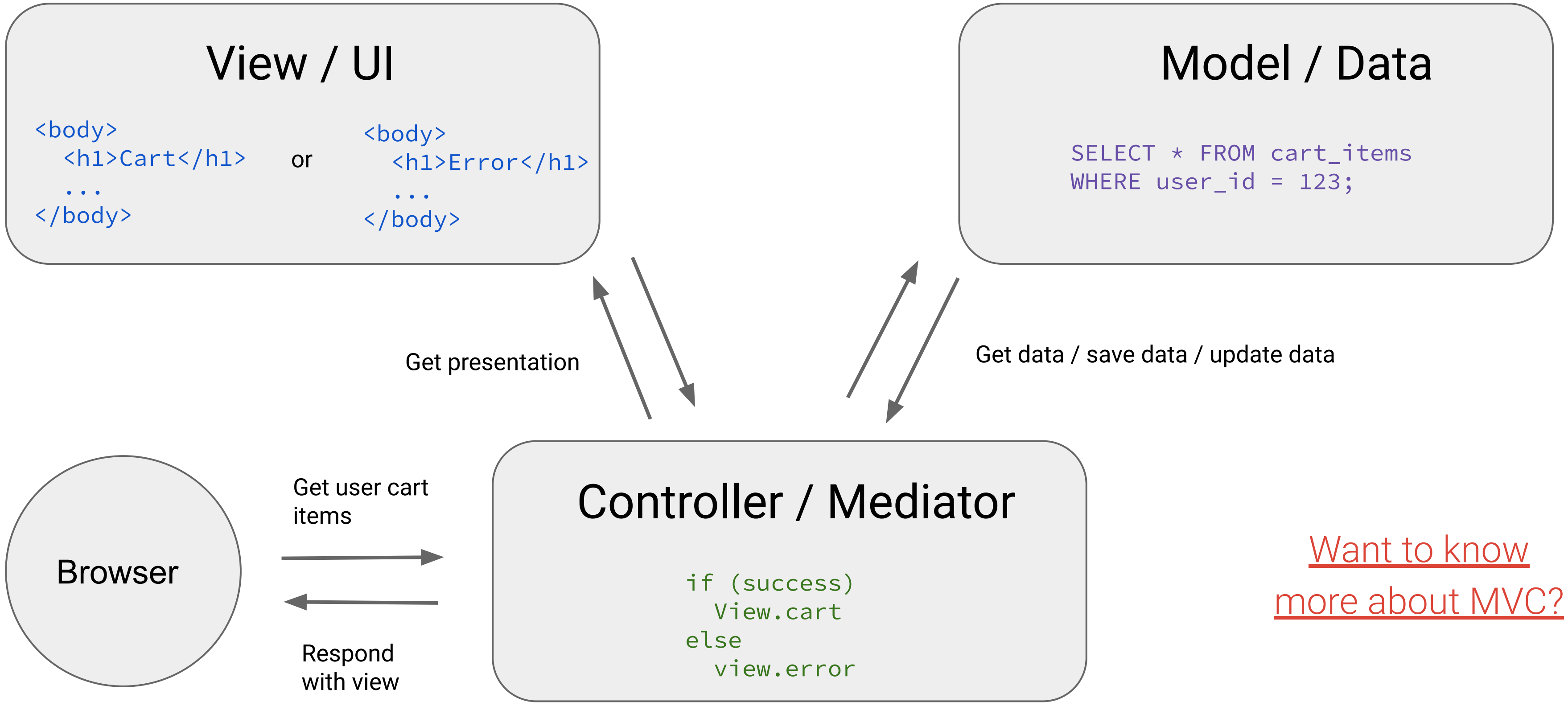
Play Framework is an **open-source web application framework** which follows the **model–view–controller (MVC)** architectural pattern. It is written in Scala and usable from other programming languages that are compiled to JVM Bytecode, e.g. Java

Want to know more?

What is MVC pattern ?



What is MVC pattern ?



```
<body>
  <h1>Cart</h1>
  ...
</body>
```

```
<body>
  <h1>Error</h1>
  ...
</body>
```

```
SELECT * FROM cart_items
WHERE user_id = 123;
```

```
if (success)
  View.cart
else
  view.error
```

Want to know more about MVC?

MVC
Explained
in
8 Minutes

Why we should consider play ?

- Developer friendly
- Scalability and language compatibility
- Eco-System support (Java)
- Performance (compiled code runs on jvm)
- Modern web and mobile support
- Production ready and proven

Why we should consider play ?

EA

eero

theguardian

Walmart

iHeart MEDIA

verizon

LinkedIn

NCL NORWEGIAN CRUISE LINE

SAMSUNG

UniCredit Group

weightwatchers

zalando

Let's install play

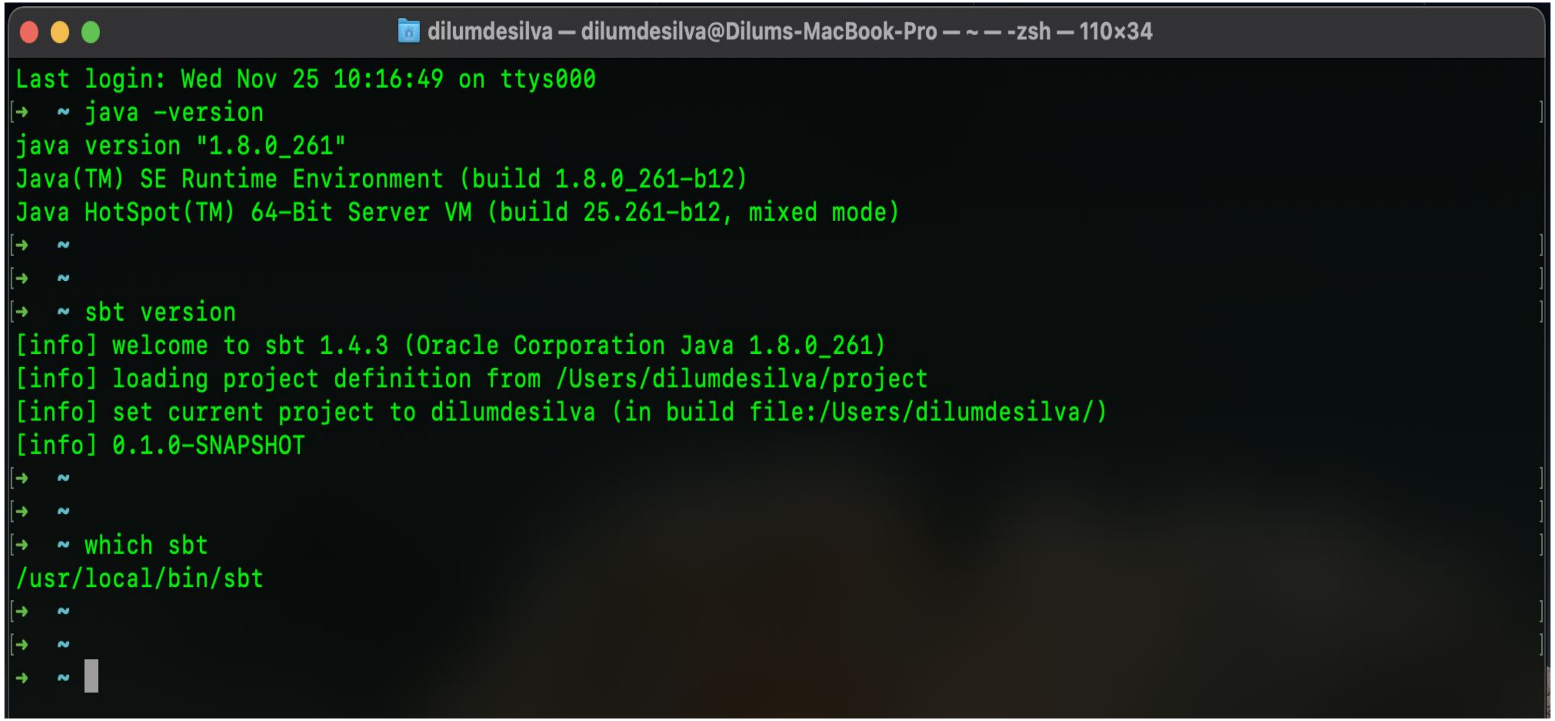
You need to have,

- Java 1.8 build (need to switch between multiple java version?)
- sbt latest - https://www.scala-sbt.org/index.html
- IDE (Prefer Intellij) - https://www.jetbrains.com/idea
- IDE plugins - scala and play

If you're on macOS

You need to have **homebrew** installed and if you recently updated to macOS **Big Sur** with **homebrew** you need to reinstall terminal tools.
https://apple.stackexchange.com/questions/401899/homebrew-your-clt-does-not-support-macos-11-0

Let’s verify our installations...


dilumdesilva — dilumdesilva@Dilums-MacBook-Pro — ~ — -zsh — 110×34

```
Last login: Wed Nov 25 10:16:49 on ttys000
→  ~ java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
→  ~
→  ~
→  ~ sbt version
[info] welcome to sbt 1.4.3 (Oracle Corporation Java 1.8.0_261)
[info] loading project definition from /Users/dilumdesilva/project
[info] set current project to dilumdesilva (in build file:/Users/dilumdesilva/)
[info] 0.1.0-SNAPSHOT
→  ~
→  ~
→  ~ which sbt
/usr/local/bin/sbt
→  ~
→  ~
→  ~
```

```
> sbt new playframework/play-java-seed.g8
```

```
 app                      → Application sources
  └ assets                 → Compiled asset sources
     └ stylesheets         → Typically LESS CSS sources
     └ javascripts         → Typically CoffeeScript sources
  └ controllers            → Application controllers
  └ models                 → Application business layer
  └ views                  → Templates
build.sbt                  → Application build script
conf                       → Configurations files and other non-compil
ed resources (on classpath)
  └ application.conf      → Main configuration file
  └ routes                 → Routes definition
dist                       → Arbitrary files to be included in your pr
ojects distribution
public                     → Public assets
  └ stylesheets            → CSS files
  └ javascripts            → Javascript files
  └ images                 → Image files
project                    → sbt configuration files
  └ build.properties       → Marker for sbt project
  └ plugins.sbt            → sbt plugins including the declaration for
 Play itself
lib                        → Unmanaged libraries dependencies
logs                       → Logs folder
  └ application.log        → Default log file
target                     → Generated stuff
  └ resolution-cache       → Info about dependencies
  └ scala-2.13
     └ api                 → Generated API docs
     └ classes             → Compiled class files
     └ routes              → Sources generated from routes
     └ twirl               → Sources generated from templates
  └ universal              → Application packaging
  └ web                    → Compiled web assets
test                       → source folder for unit or functional test
s
```

Combine
play with angular

Configuration 01

Build backend and frontend isolated in different projects and use REST interface to communicate.

Configuration 02

Build both backend and frontend in the same project, Use scala views to expose frontend entry point and communicate with backend using the REST interface.

Configuration 03

Build both frontend and backend in the same project: Use play static routes to serve frontend and communicate with backend using the REST interface.

This is the approach that we are planning to use.

Other resources

- Installing and maintaining multiple java versions using **jenv.**
  - https://github.com/jenv/jenv
- Installing and maintaining multiple node versions using **nvm.**
  - https://github.com/nvm-sh/nvm
- Play framework docs
  - https://www.playframework.com/documentation/2.8.x/Home

So, what's Coming next

Homework