Three Tier Web Architecture

Contents

- Why n-tier?
- Layers.
- Monolithic or 1-tier architecture.
- 2-tier architecture.
- 3-tier architecture.
- Need of MVC.
- MVC architecture.
- MVC components.
- Comparison between MVC and 3-tier.

Why n-tier?

- Need of e-commerce solutions; increase in users and merchant sites all over the world.
- Applications should be scalable, user-friendly, have tight security and easily maintainable.

Layers

- Layer means logical.
- Tier means physical.
- Generally there are three layers:-
  - ✓ Presentation
  - ✓ Business
  - ✓ Data access layer

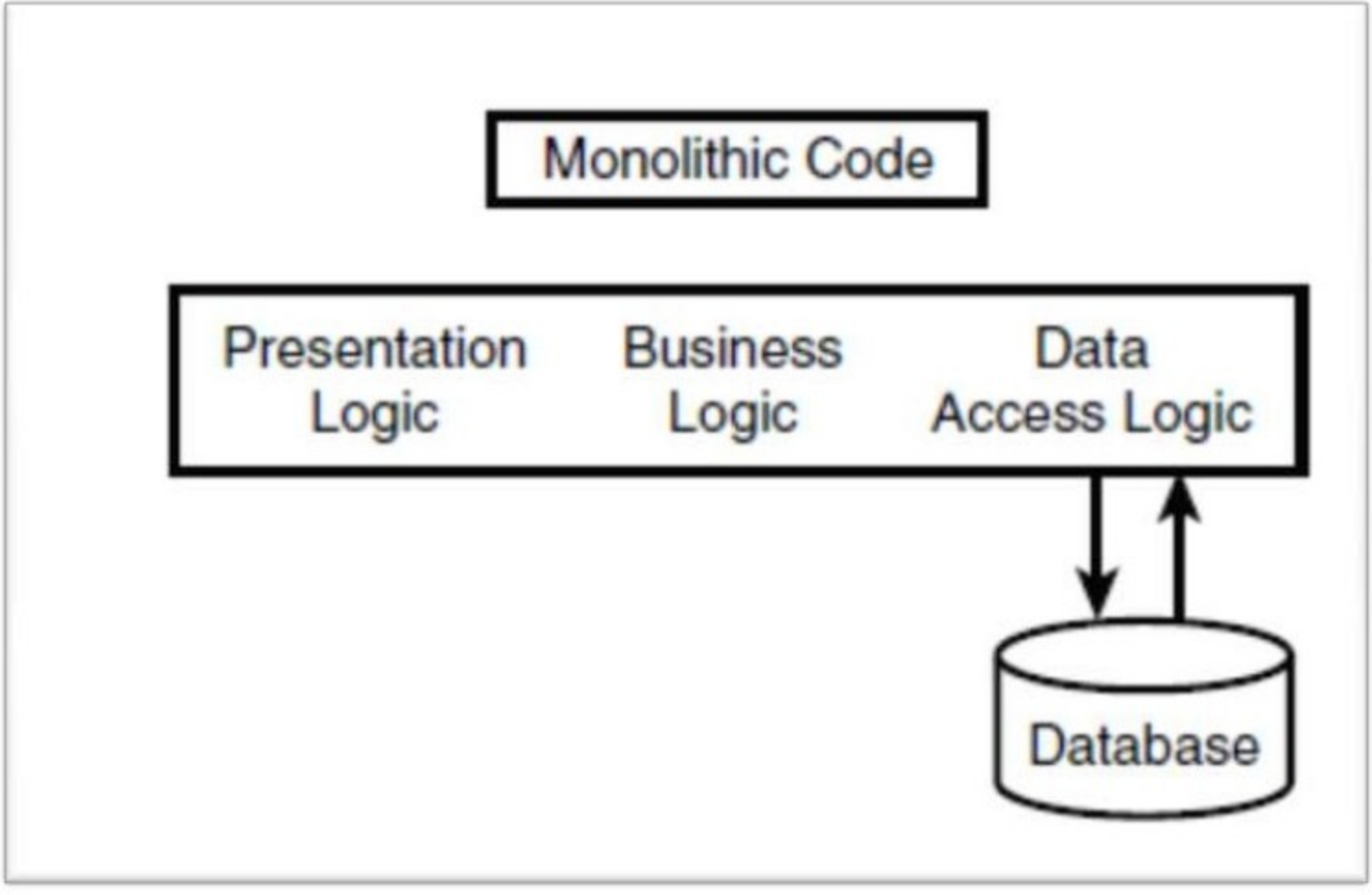
Layers

Presentation Layer:- involves with client and application interaction. Provides user friendly interface for the clients.

Business Layer:- contains the application code or the core functionalities of the application or what the application will perform.

Data access Layer:- involves with the maintaining database and storage of data.

Monolithic or 1-tier



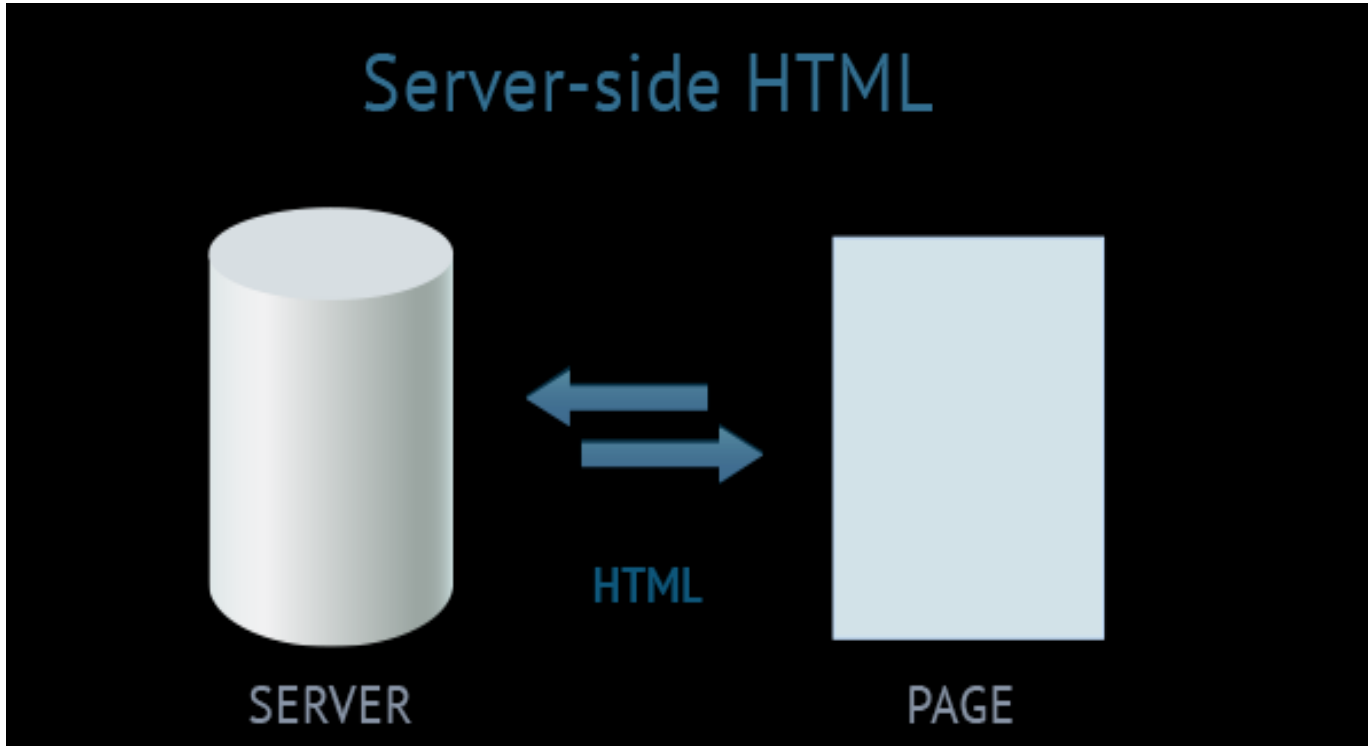
Monolithic or 1-tier

- Presentation layer, Business layer and Data Access layer are tightly connected.
- As the layers are tightly connected(depends on each other), if any of the layer code are changed then other layers should be affected and the codes in other layers need to be changed.

Monolithic or 1-tier

- Traditional approaches of the applications are based on this type of architecture.
- Typically implementation of 1-tier architecture can be a C program running in a computer.

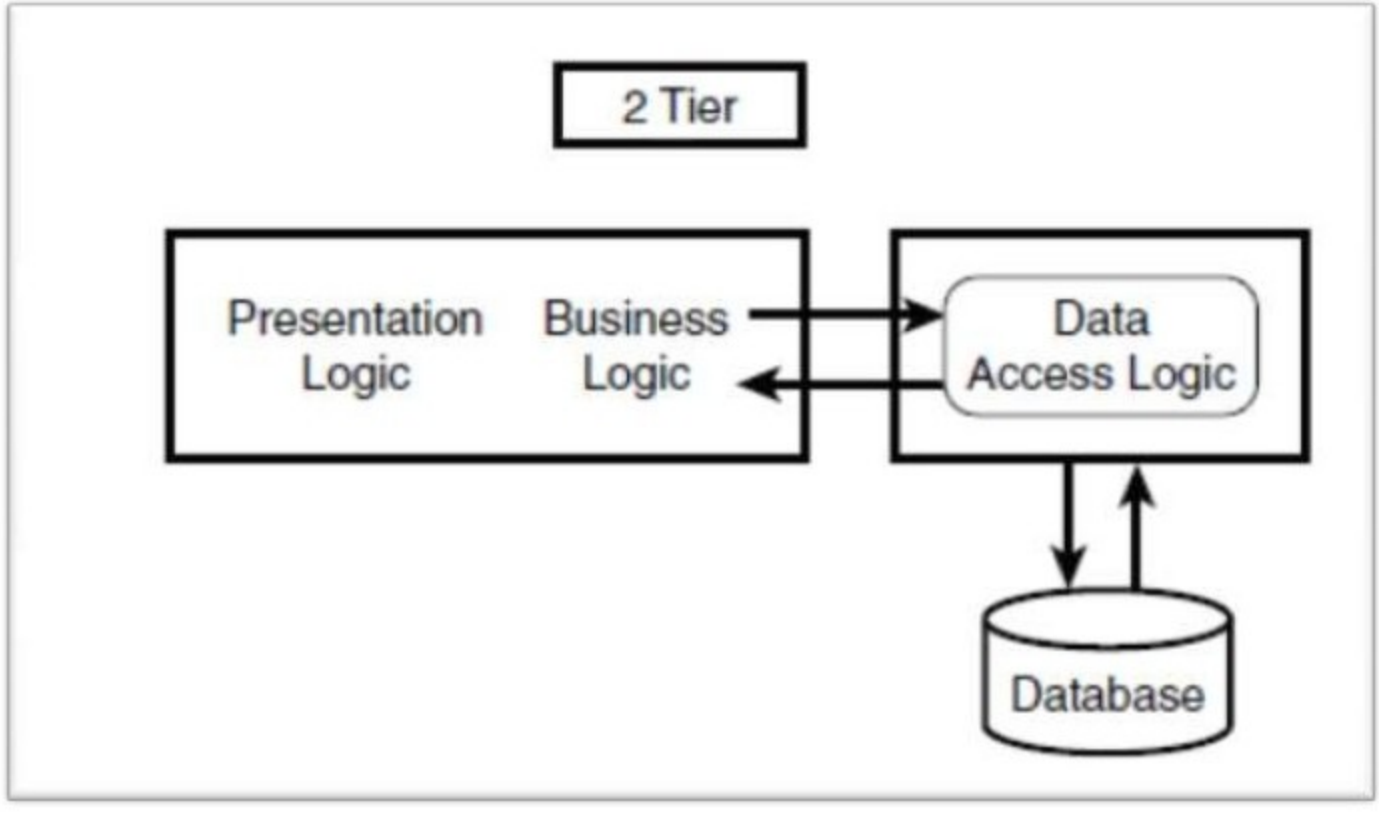
Type 1: server-side HTML web application



Type 1: server-side HTML web application

- The server generates HTML content and sends it to the client as a full-fledged HTML-page.
- Sometimes this architecture is called "Web 1.0", since it was the first to appear and currently dominates the sphere of web development.
- A huge amount of data is transferred between the server and the client.
- The user has to wait until the whole page reloads, responding to trivial actions, for example, when only a part of the page needs to be reloaded.
- UI templates on the client depend directly on the frameworks applied on the server. Due to the limitations of mobile Internet and huge amounts of transferred data, this architecture is hardly applicable in the mobile segment.
- There are no means of sending instant data updates or changes in real time.

2-tier


2 Tier
Presentation Logic
Business Logic
Data Access Logic
Database

2-Tier

- In this type of architectures the presentation layer, the business logic layer are separated from the data access layer.

- The advantages of this layer is that the code of the data access layer can be changed any time without affecting the code of the other layer i.e. the whole database and the layer can be changed anytime.

- The database(i.e. the data access layer) can be present anywhere around but the other two layers should be together(tightly connected).

2-Tier

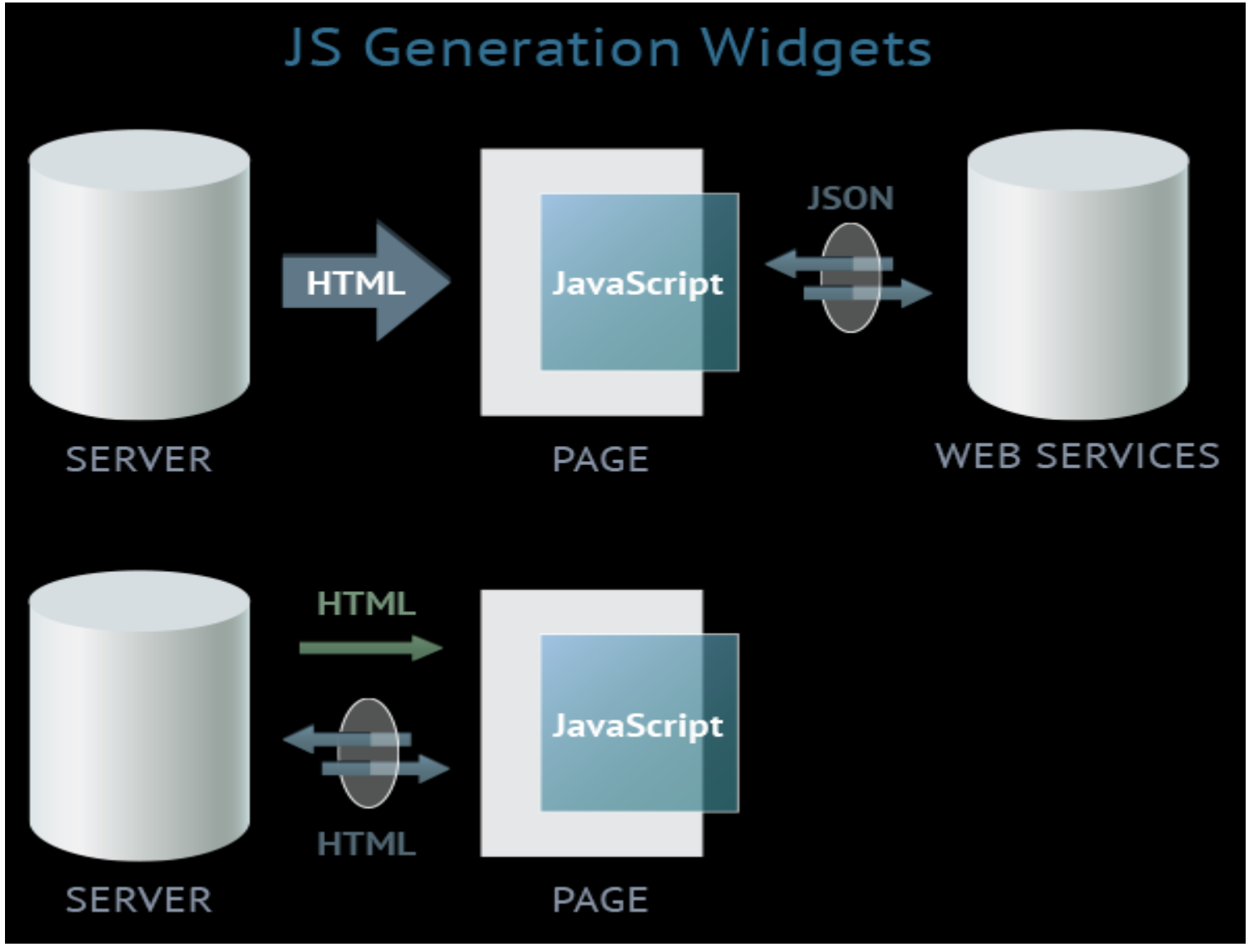
- As the presentation and the business logic are still connected they should be present at the client side to work together; due to the concentration of the client this type of client is called thick client.

- Problems faced by this type of architecture is the client should always get the updated copies of the application if there is a change in the application codes or application developer modifies the application.

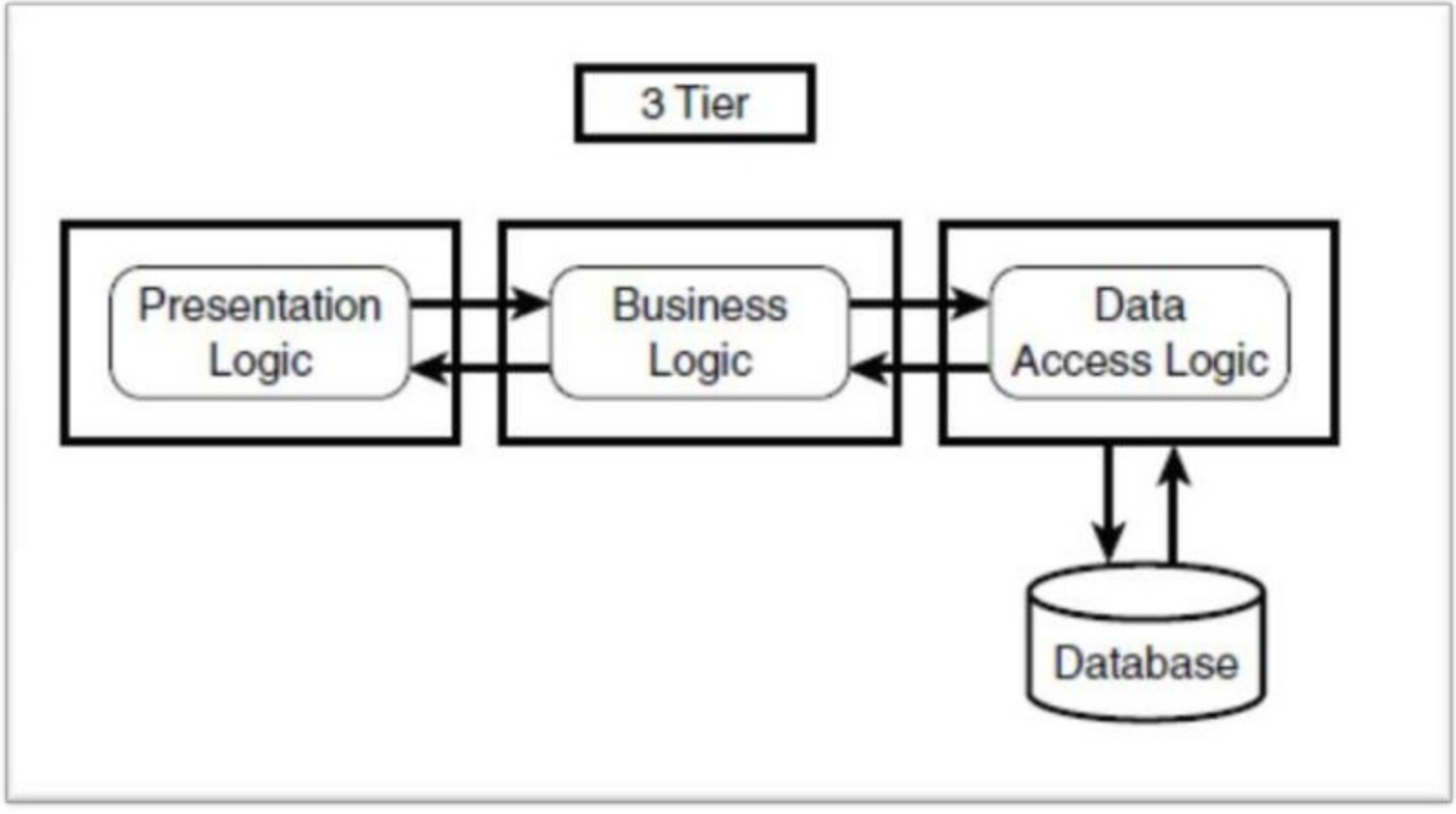
- The application developer may not want to give the code to the relatively third parities even if the code is pre compiled.

- Another problem is that the client may not want to install the code into his/her machine for using a particular application.

Type 2: JS generation widgets (AJAX)



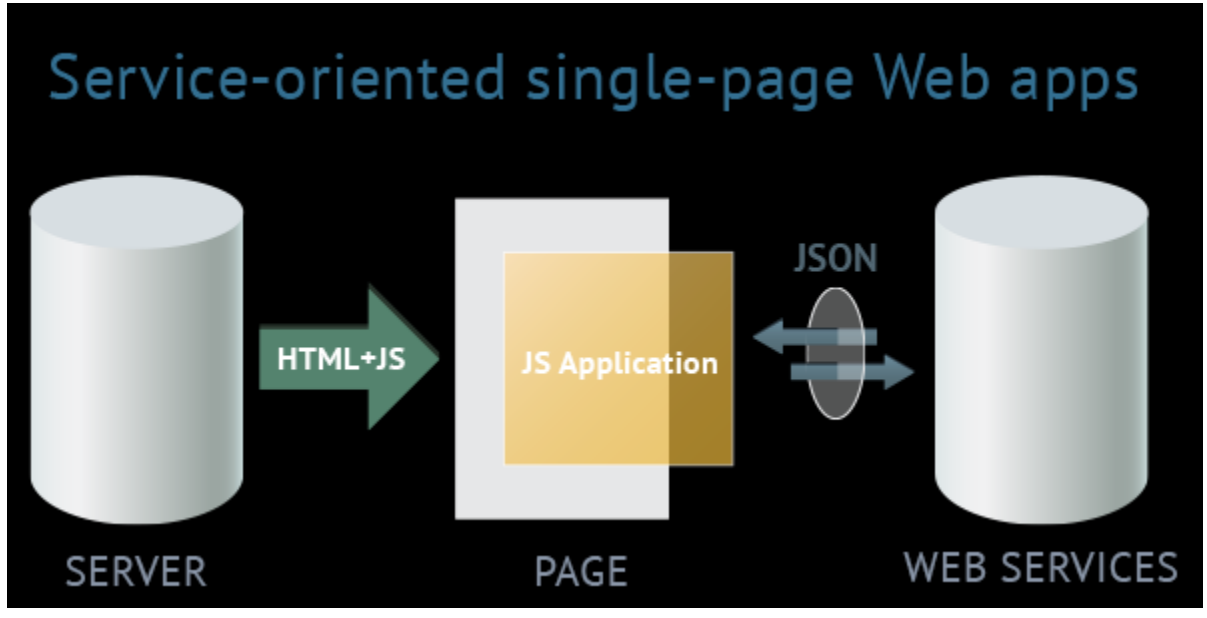
3-tier


3 Tier
Presentation Logic
Business Logic
Data Access Logic
Database

3-tier

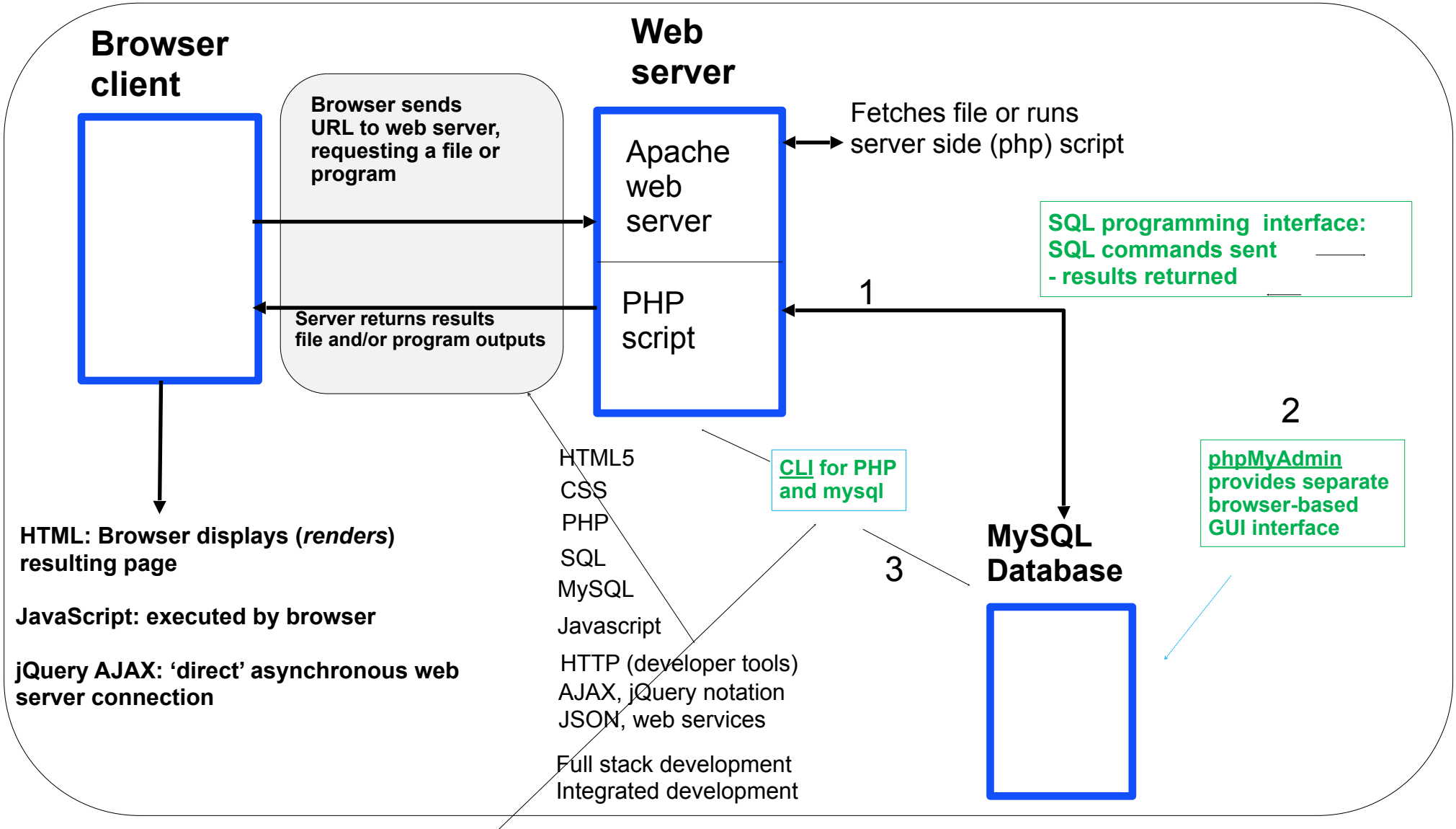
- In this type of architecture the presentation layer, the business logic layer and the data access layer are separated from each other and are present on three different tiers therefore they are loosely connected.
- The main advantages is that any change in the code in one layer will not affect the other layers and the platform can also be changed independently.
- Now the web designer can concentrate on the design of the user interface i.e. the presentation logic, the application developer concentrate on developing the application i.e. the business logic and the database manager can handle the database independently.
- Today's application are based on 3-tier architecture which are scalable, easy to maintain, components are reusable and faster development.

Type 3: service-oriented single-page web apps (Web 2.0, HTML5 apps)



Run as slide show to see animated display.

3-Tier Architecture



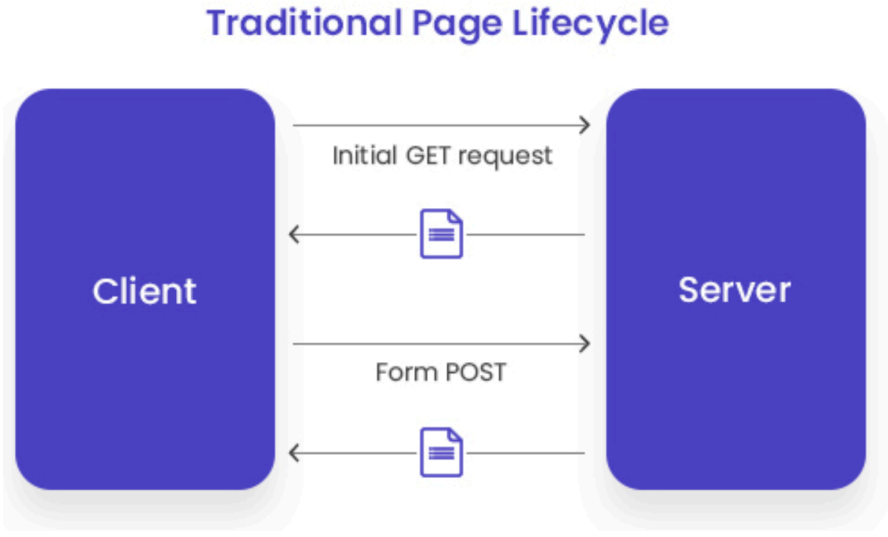
Command Line Interface (CLI) (eg in MobaXterm) provides alternative DB interface

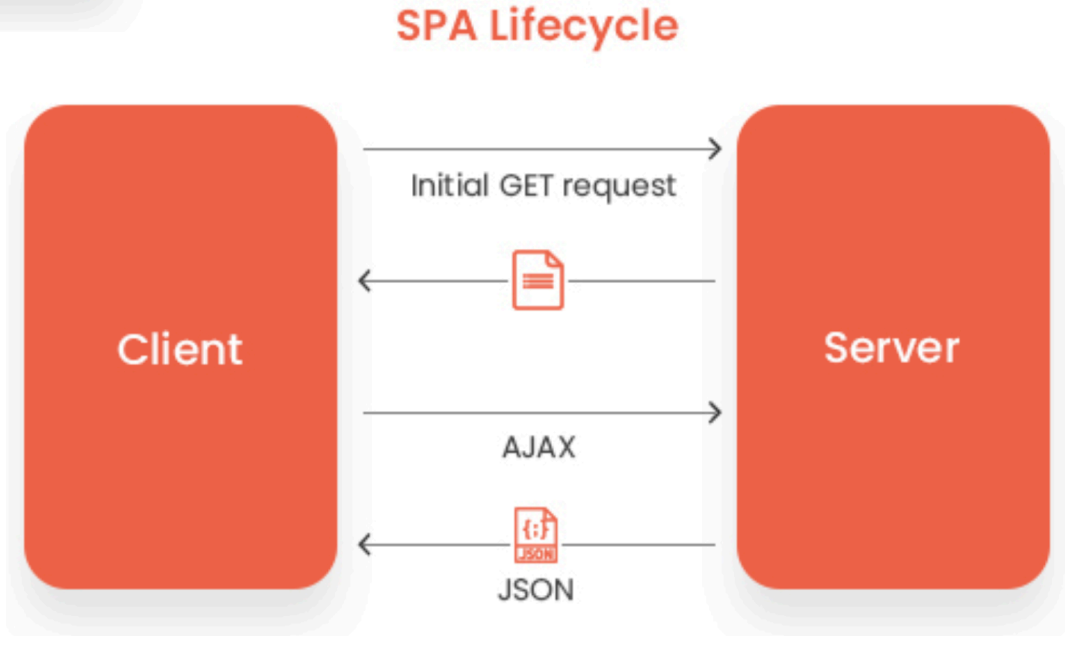
mysql commands → performance
php in CLI →error reporting

Single-page web apps & Service-oriented

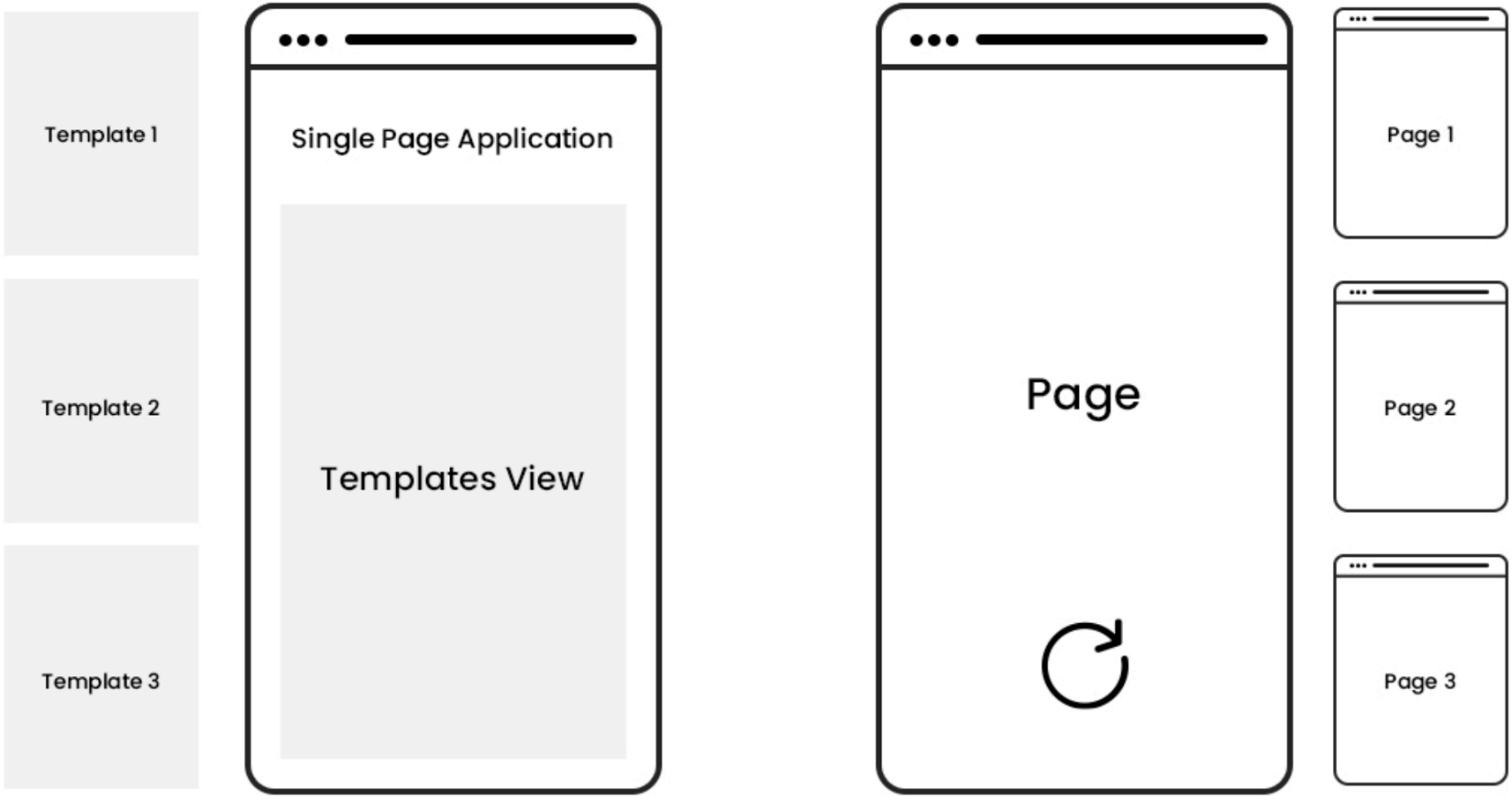
- Single Page Applications (aka SPA), and Service Oriented Architecture (aka SOA)
- SOA refers to the architecture of the back-end business layer, architecting for service reusability.
- This approach avoids interruption of the user experience between successive pages, making the application behave more like a desktop application.
- In an SPA, either all necessary code — HTML, JavaScript, and CSS is retrieved with a single page load
- Appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions.
- Thick and Thin architectures for client and server- Stateful and Stateless Versions

Single-Page web-apps



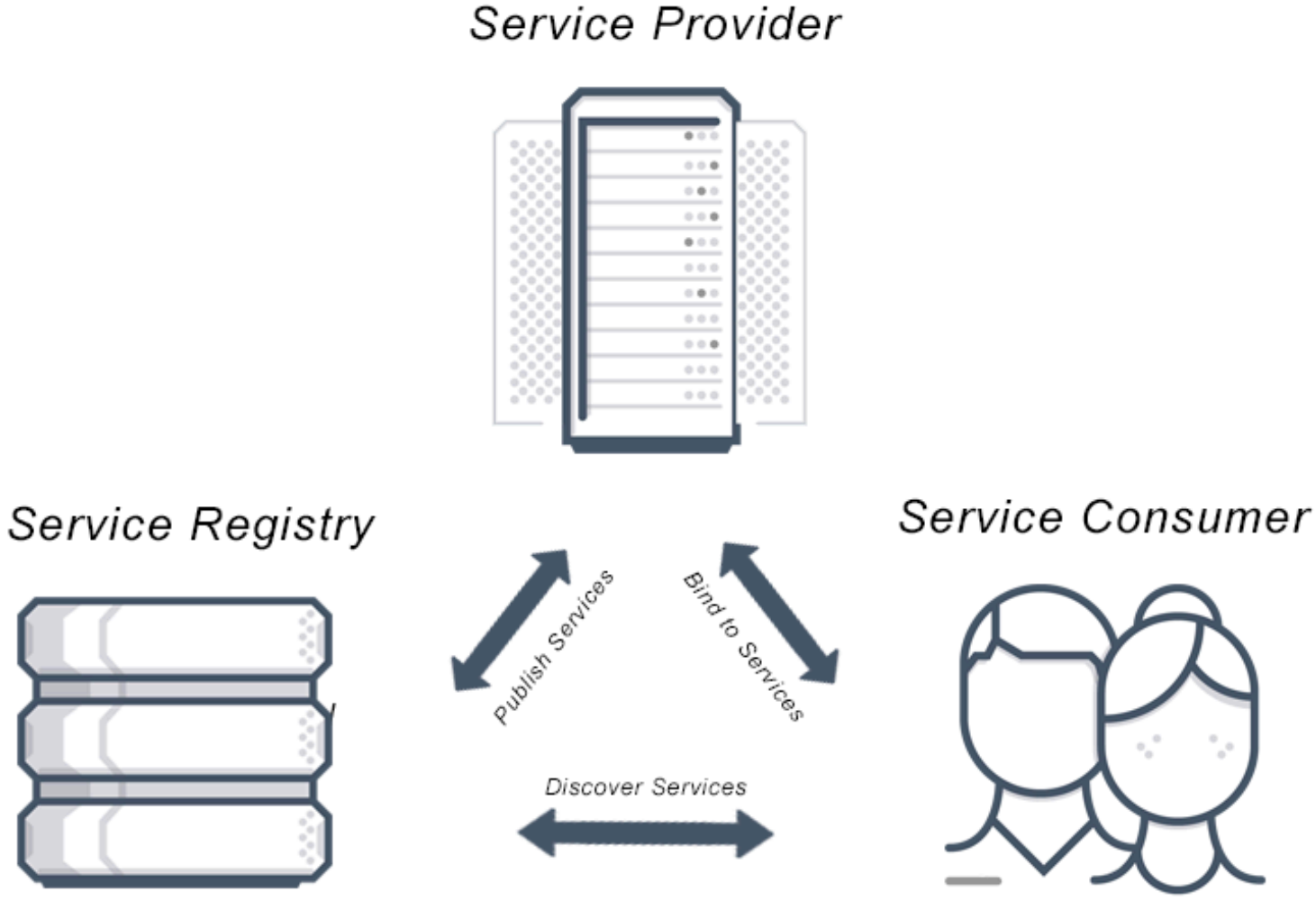


Single-Page web-apps



Single-Page web-apps

The Service Oriented Architecture Triangle



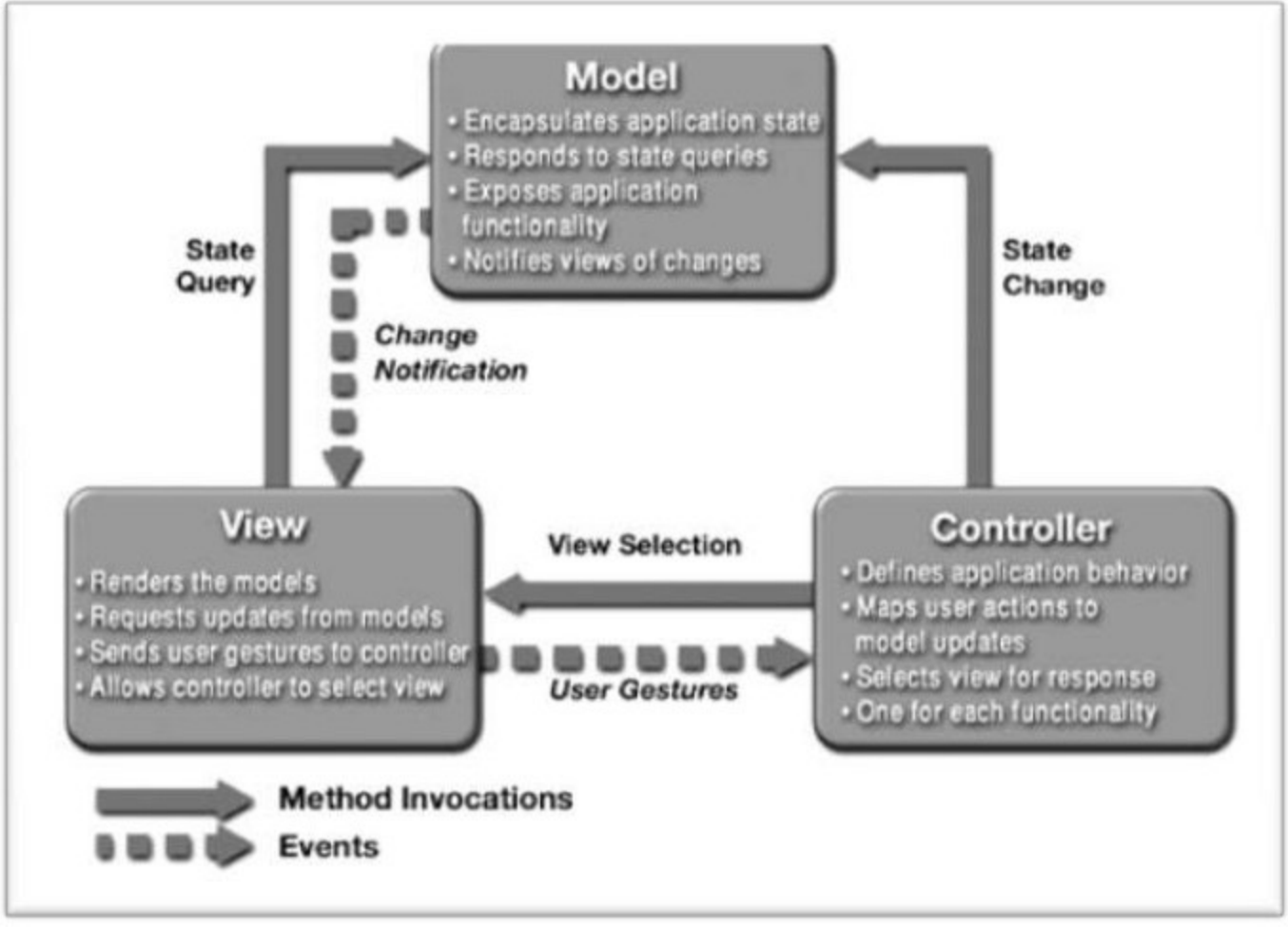
Need of MVC Architecture

- Need to access the data from different context such as mobiles, touch screen, desktop, etc.
- Need to show the data in different context such as mobiles, touch screen, desktop, etc.
- Need to show multiple views of the same data such as Thumbnails, List or details.
- Need to change the designs without changing the core business logic.

MVC Solutions

- Separate the core business logic form the presentation logic.
- Separate views for the same data.

MVC Architecture



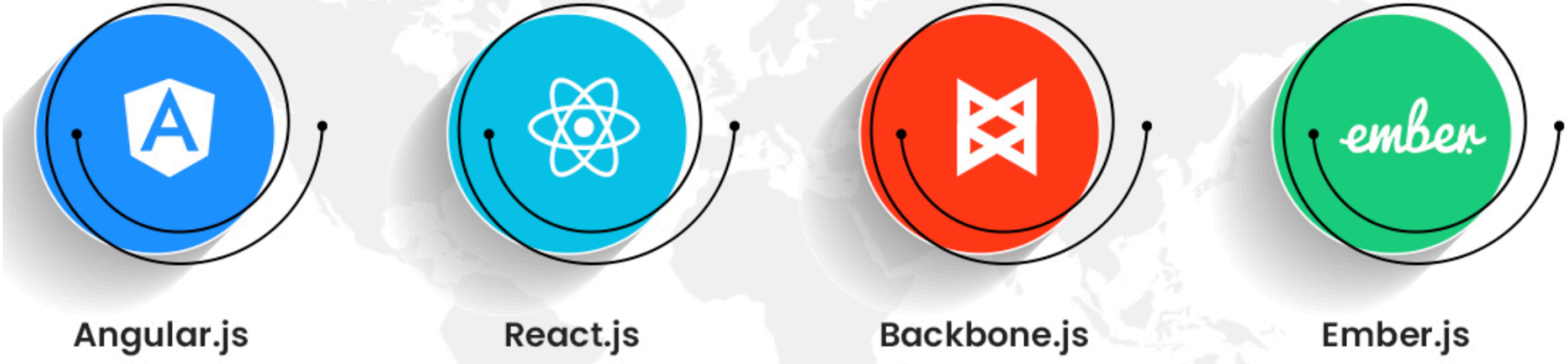
MVC Components

- *Model*: It contains the core functionalities and the business logic of the application. It accepts the state query from the model and controller and it provides the updated information to the view component.
- *View*: This component is responsible for the presentation logic and the user interaction between the application. The model provides different information to different user which can be represented in different ways. The main work of the view component is to provide the suitable information to the user.

MVC Components

- *Controller*: It accepts the user input through the view component and process them and if any changes are required then it perform the changes after that it response to the client.

MVC web-apps



Thank You