MERN STACK

APRIL 2022

YONATAN BENEZRA

Web Development Frameworks

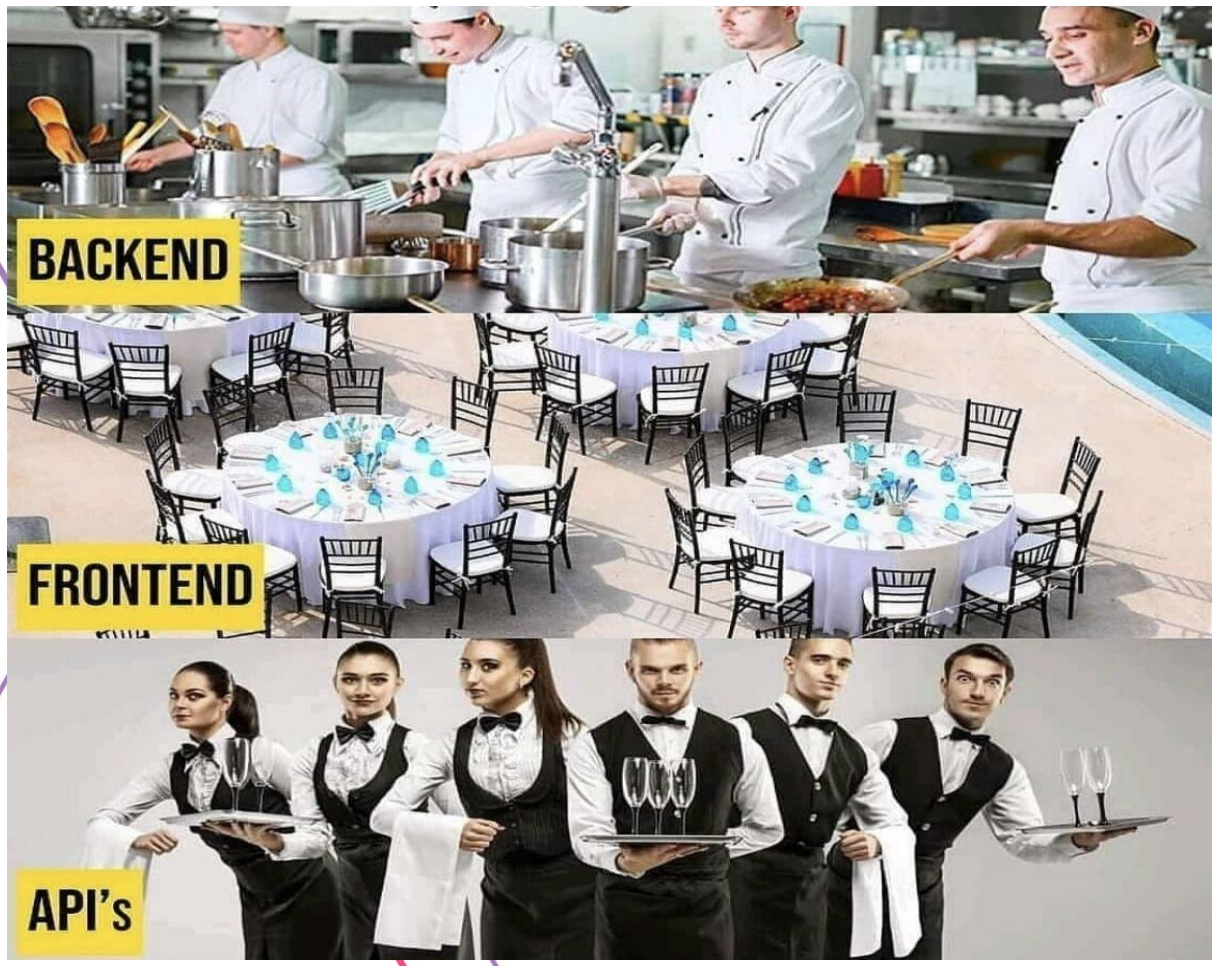
A web development framework is a software platform that simplifies the web development process and makes it easier to build a functional website, thus making our lives easier as full-stack developers.

You can think of a web application framework as a pre-built structure that manages the most common and repetitive processes associated with developing a website.

Types Of Web Frameworks: Frontend & Backend Web Frameworks

There are two major components to any website:

- The portion that is visible to the users is called frontend or client-side frameworks.
- The behind-the-scenes portion is responsible for the background functioning of the websites and is known as the backend or the server-side framework.

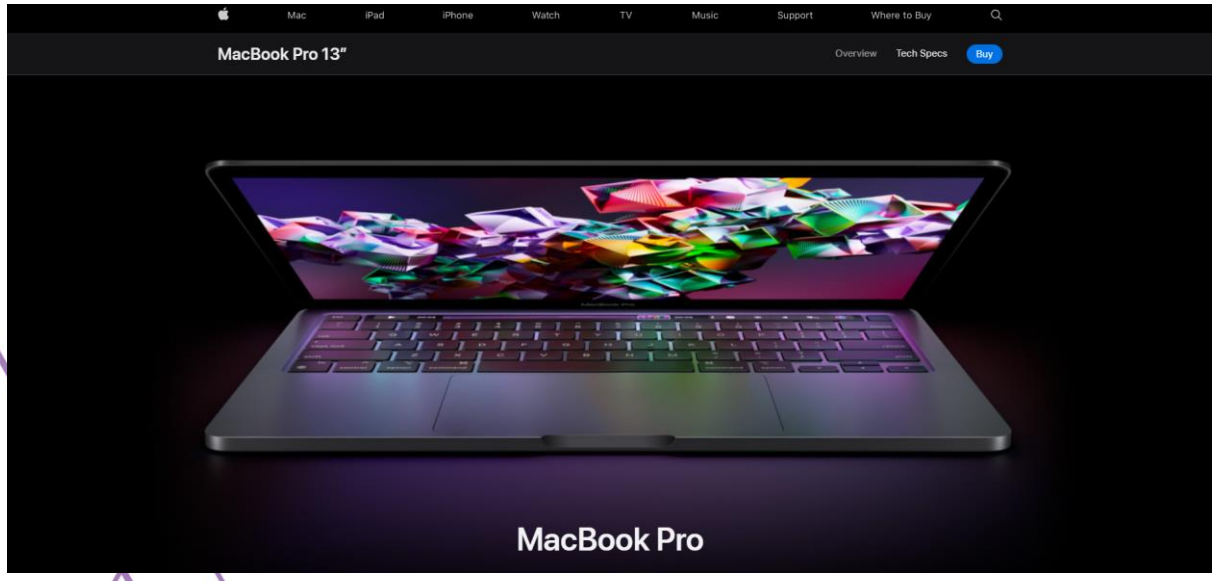


Frontend Web Development Frameworks

Frontend web frameworks usually offer tools for UI/UX designing, SEO optimization, performance optimization, and scalability. To develop the front end of any website, client-side markup (HTML) and scripting (CSS, JavaScript, and jQuery) languages are used.

Frontend web framework offers pre-written reusable design templates, code snippets, and widgets, and is responsible for managing user interactions.

HTML, CSS, JavaScript, and JQuery are examples of front-end languages.
Vue, Ember, Bootstrap, and Angular are examples of front-end frameworks.
React is a front-end library widely mistaken for a framework.



Backend Web Development Frameworks

A backend or server-side framework is a library of tools and modules that help in building the architecture of a website. These frameworks primarily focus on server-side scripting languages, such as:

- Ruby
- JavaScript
- Python
- PHP

They may also include compiled languages, such as Java and C#.

Easy database connectivity & manipulation, user authorization, encryption, third-party API integration, and reusable components are some of the benefits of using backend frameworks.

Examples of backend languages:

- Python
- JavaScript
- PHP
- Ruby
- .NET

Examples of backend frameworks:

- Django
- Ruby On Rails
- Spring
- Express
- ASP.NET Core

Why Should You Use Web Frameworks?

Today, web development frameworks have become an important part of building a website.

There are many explanations for their popularity, including:

- Web application frameworks help you build robust and interactive web applications.
- Integrated web frameworks support the development of both the frontend and the backend.
- With increased and better user interaction, the chances of the success of your website improve dramatically.
- They help in reducing and simplifying the web development and maintenance processes.
- As the code of these frameworks is thoroughly tested by millions of coders, you can easily avoid common pitfalls, errors, bugs, and security lapses.
- The overall programming process becomes ordered and systemized.
- A web framework effectively takes care of all the background details, such as data binding and configuration settings.

Advantages Of Using Frameworks

1. Reduces Code Length

A general task can keep you involved for hours or days with many lines of code. A great web development framework would eliminate the need to write a lot of repetitive code, allowing developers to:

- Focus on the unique needs of the client.
- Build websites and applications much quicker.
- Deliver more productivity in the same timeframe.

2. Reinforces Security

A web framework takes many months to be developed and released. Any updates and patches are regularly released to make them less vulnerable. Most popular frameworks also have a strong support team or developer community.

With such advanced support and security features, you can rest assured knowing that there are experts to take care of any issues that may arise.

3. Makes Debugging & Application Maintenance Easier

Most frameworks have the dedicated support of a strong community of developers. They are quick to respond to any problem that you may encounter while using the framework.

4. Cost-Effective

Most popular web application frameworks are open-source and available for free. Since most of the templates and website functionality are already present, the development is faster.

The per-unit cost of development is reduced dramatically due to these two reasons alone. It also helps in the quick delivery of projects.

Web framework architectures:

1. Model View Controller (MVC)

The MVC model is an architectural pattern that separates an application into three main logical components – Model, View, and Controller.

- **Model:** Includes all the data, its schema, and related logic.
- **View:** Presents data to the user and handles user interaction.
- **Controller:** An interface between the Model and View components, which also contains the business logic.

MVC separates the business logic and presentation layer from the backend data being managed by the Model. It enables fast development, easy testing, and full control over your HTML and URL structure.

Advantages

- Easy to learn, get support, and receive guidance, even for amateurs.
- Supports parallel development of multiple components or modules.
- SEO-friendly URLs and webpages.
- Uses the Front Controller design pattern to process web application requests through a single controller.

Disadvantages

- MVC is difficult to implement with a modern UI.
- Parallel development requires many programmers to be dedicated to the same project.
- The business logic on the controller requires extensive maintenance.

2. Model-View-View-Model (MVVM)

The MVVM, also known as the Model-View-Binder architecture model, is used by popular modern frameworks such as VueJS and KnockoutJS. It was primarily designed to develop software applications but was later adapted for the web.

The model view of the MVVM is similar to that of the MVC. The view model is defined as a bridge between the View and Model, and acts as the business logic.

MVVM manages the data from the underlying model to make data management very easy.

Advantages

- Enhances the reusability of code.
- All modules independently improve the testability of each layer.
- Makes project files maintainable and easy to make changes to.

Disadvantages

- The design pattern is not ideal for small projects.
- If the data binding logic is too complex, the application debugging becomes challenging.

3. Push-based vs. Pull-based Models

a) Push-Based Models

The work or user queries are “pushed” to the server, which has no choice in the matter. These architectures are also known as event-driven or action-based frameworks. The data (Model) is constructed and given to the View layer by the Controllers with the help of session or application scope global variables.

Advantages

- Incoming data is streamed to all users and integrated systems.
- The simpler systems use the push model to serve their core purpose of real-time alerts.

Disadvantages

- A hard-coded list of addresses is required to send requests.
- Load balancing is challenging.

Some examples are Ruby on Rails, Spring MVC, Django, Sails.js, and CodeIgniter.

b) Pull-Based Models

In this architecture, the server actively requests work, usually through an intermediary. They are commonly known as component-based systems.

Advantages

- These systems support work distribution.
- The pull-based systems use an intermediary and can keep the identity of the client and the server confidential.

Disadvantages

- Very difficult to receive the response at the right time.

4. Three-Tier Organization

The classical, three-tier architecture is used to organize applications into three logical and physical computing tiers: the presentation tier, the application tier, and the data tier.

The Three Tiers

- **Presentation Tier–** The UI or the communication layer of the application. It displays information and collects data from the users. HTML, CSS, and JavaScript are used to develop this tier.
- **Application Tier–** Where the data is processed. Commonly called the logic or the middle tier. It acts as the glue between the presentation and data tiers.
- **Data-Tier–** This is where the data associated with the application is stored and managed in databases and data files. It is also known as the data access tier or the backend.

Advantages

- Improved scalability and data integrity.
- Ability to update the technology stack of one tier, without affecting the others.

Disadvantages

- A more complex model with more interface points.
- A separate proxy server may be required.