

Web Development Frameworks

PV219, spring 2024

Agenda

- What is a Web Development Framework (WDF)
- Benefits of using WDF
- Classification of WDF Architectures
- Types of WDF
- Difference between Front-End and Back-End WDF
- Front-End Frameworks
- Back-End Frameworks
- How to choose WDF

What is a WDF

Framework is a pre-established collection of software **tools, libraries, and guidelines** created to expedite and streamline the process of web application development.

Using a framework can assist in ensuring that the finished product is **scalable, reliable, and maintainable** due to the standardized approach to development it offers.

Benefits of using Frameworks

- **Speed and Efficiency:** It allows developers to build applications faster and with less code.
- **Consistency and Modularity:** Easy to maintain and scale applications over time.
- **Security:** Includes built-in security features that help to protect against threats.
- **Community Support:** Supports large communities of developers who provide support and resources.
- **Scalability:** Allows scaling of applications as they grow in size and complexity.
- **Testing and Debugging:** Offers various tools and features to make testing and debugging easier

Classification of Architectures

Architecture defines the structure and organization of the application and how different components interact with each other.

- Model-View-Controller (MVC)
- Model-View-ViewModel (MVVM)
- Three-Tier Architecture (PAD)

Model-View-Controller (MVC)

The **model** represents the data, the **view** represents the user interface, and the **controller** handles user input and business logic.

Usage: Ruby on Rails, Laravel, Spring, Django, ASP.NET

Model-View-ViewModel (MVVM)

Model-View-Controller (MVC) architecture but with some key differences. The **view-model** manages the communication between the model and the view.

Usage: AngularJS, Vue.js, Knockout.js, React

Three-Tier Architecture (PDA)

Also known as Multi-Tier Architecture, separates the application into three main tiers.

The **presentation** tier handles the user interface, the **application** tier handles the business logic, and the **database** tier handles data storage and retrieval.

Usage: CakePHP, Express.js, Ember.js, Backbone.js

Types of WDF (1/2)

Front-end frameworks are focused on the presentation layer of a web application and are responsible for **rendering the user interface**.

Some popular front-end frameworks include AngularJS, React, and Vue.js.

Types of WDF (2/2)

Back-end frameworks are focused on an application's **server-side functionality**. They provide the tools and structure necessary to build the application's backend, including database interaction, server-side rendering, and API development.

Some popular back-end frameworks include Ruby on Rails, Django, and Express.

Difference between FE and BE

Front-end provides pre-written tools and libraries for building interactive and responsive user interfaces like buttons, forms, and data visualization components.

Typically use a variety of **programming languages**, such as HTML, CSS, and JavaScript.

Handle tasks such as data binding, state management, and routing within the user's browser.

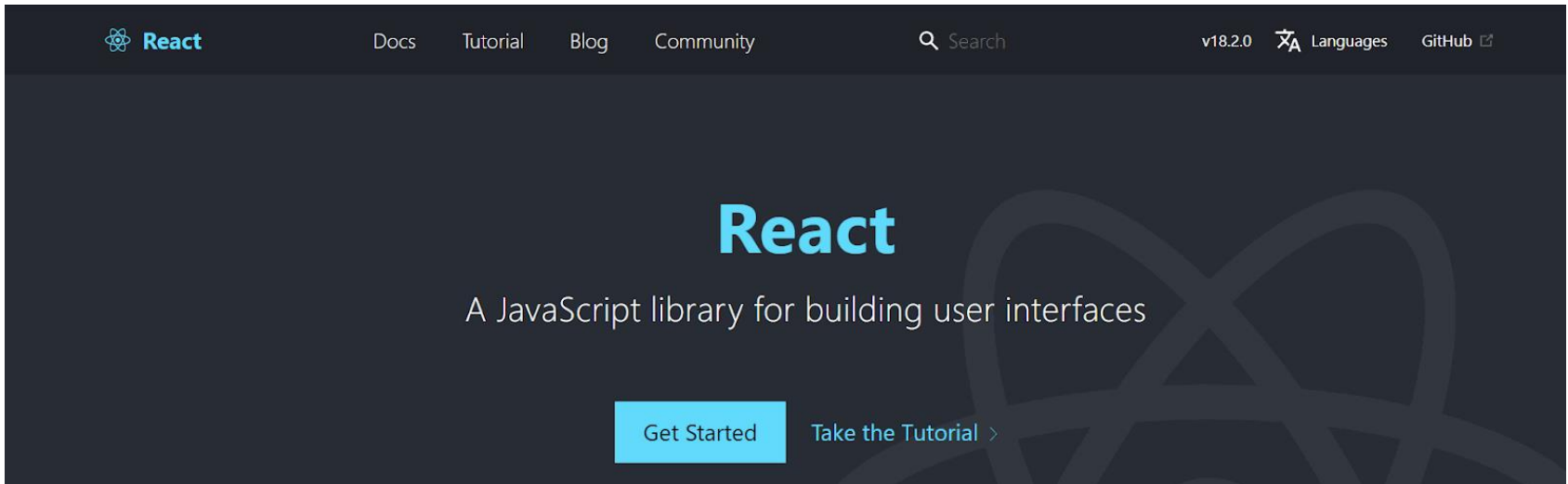
Difference between FE and BE

Back-end provides pre-written tools and libraries for building scalable and secure server-side applications, such as routing, authentication, and database integration.

Typically use a variety of **programming languages**, such as Ruby, PHP, Python, Java.

Handle tasks such as authentication, database interaction, and security within the server.

Front-End: React



Although it's more like a library, developers still consider [React](#) one of the best and first web development frameworks developed by Facebook to adopt a component-based architecture.

Front-End: React

Benefits:

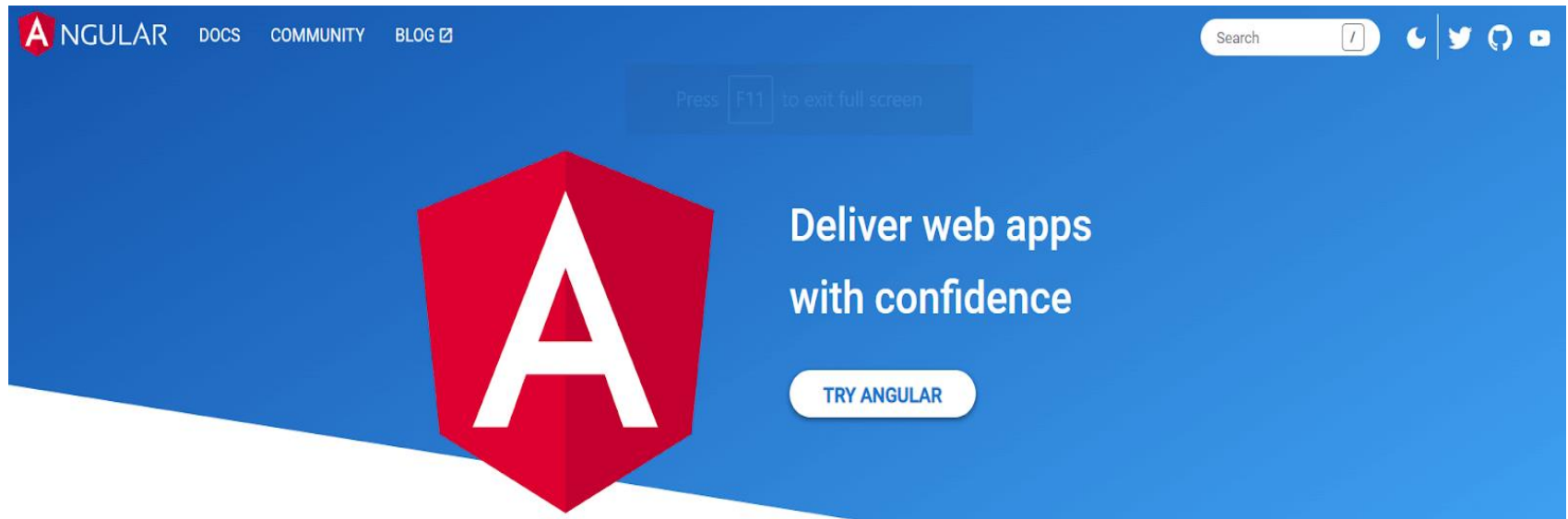
- Bringing **HTML into your JavaScript** helps developers write code quickly.
- Allows you to **break down complex UI** into smaller components and start developing the same.
- **Virtual DOM** helps React know when to re-render and ignore certain pieces of the DOM, thus increasing the page's performance.
- Due to the presence of many handy tools, when used as Chrome extensions, it allows developers to **inspect and debug the DOM** quickly.

Front-End: React

Drawbacks:

- Although it's rapidly growing, **community** space is smaller than others.
- Continuous upgradation in technology has become troublesome for developers in terms of making proper **documentation**.

Front-End: Angular



[Angular](#) is a web development framework for building dynamic single-page web applications (SPAs) for mobile, desktop, and web. It's led by Google's Angular Team and supported by a large community of developers.

Front-End: Angular

Benefits:

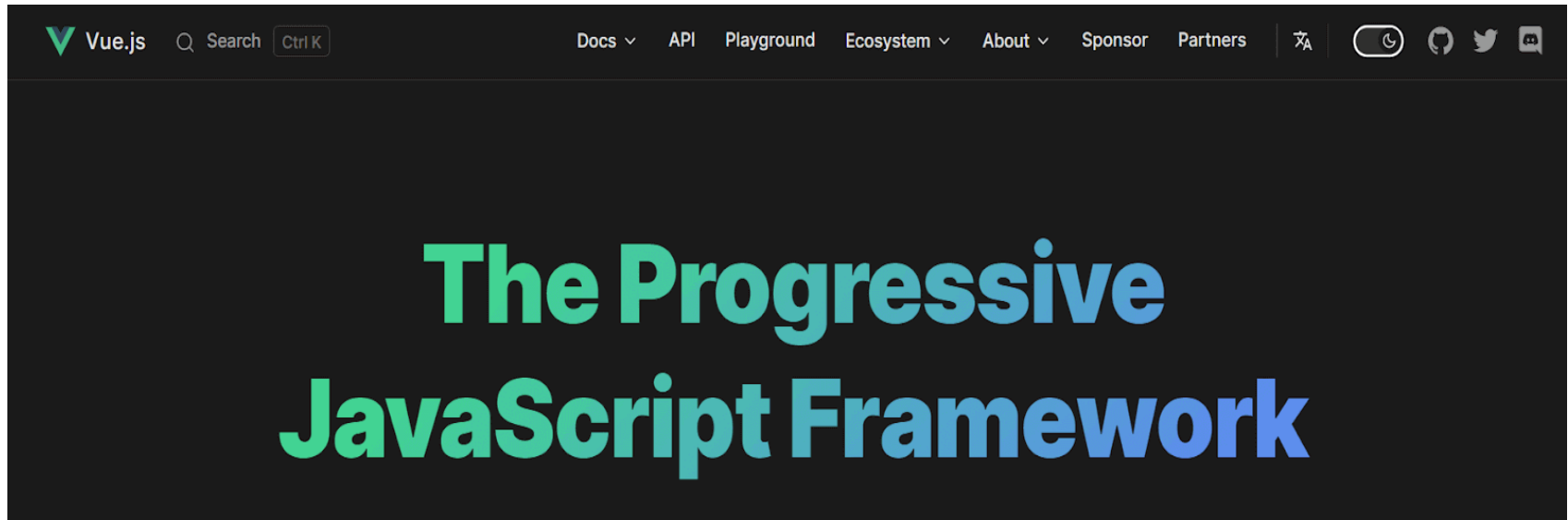
- Well-detailed **documentation** that keeps updated in a timely manner by Google.
- Allows various **third-party integrations** that improve the overall functionality of websites.
- Allow **dependency injection** to execute several functions at a time.
- Provides faster **loading time**.

Front-End: Angular

Drawbacks:

- Have a **steep learning curve**.
- Requires a lot of **boilerplate code** that makes it complex during installation.

Front-End: Vue.js



[Vue.js](https://vuejs.org) is one of the progressive JavaScript frameworks allowing developers to build desktop and mobile apps and web interfaces. Developed in 2013, but still ranked among the third web development frameworks.

Front-End: Vue.js

Benefits:

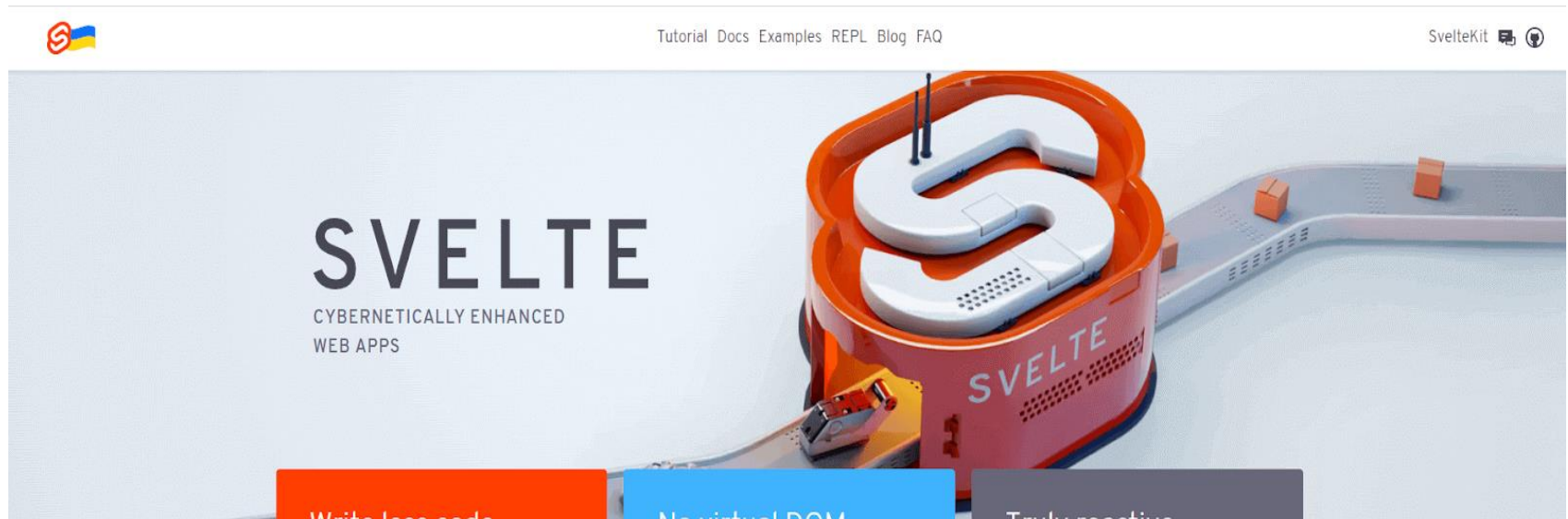
- With a minified jQuery library, the **compressed Vue.js** weighs around 29 KB.
- **Simple and easy** to grasp.
- Offers **extensive documentation**, which makes it easier to learn.
- Can easily handle **2-way data binding**, thus making the code reactive.

Front-End: Vue.js

Drawbacks:

- Being flexible, but while working on **large-scale projects** with multiple developers, flexibility can lead to complexity.
- **Smaller community** and lacks widespread support.

Front-End: Svelte



[Svelte](#) is a modern framework that allow developers to use its clear-to-read HTML templating. The approach to reactive programming is also unique, allowing for a more intuitive and efficient way of building reactive UI components.

Front-End: Svelte

Benefits:

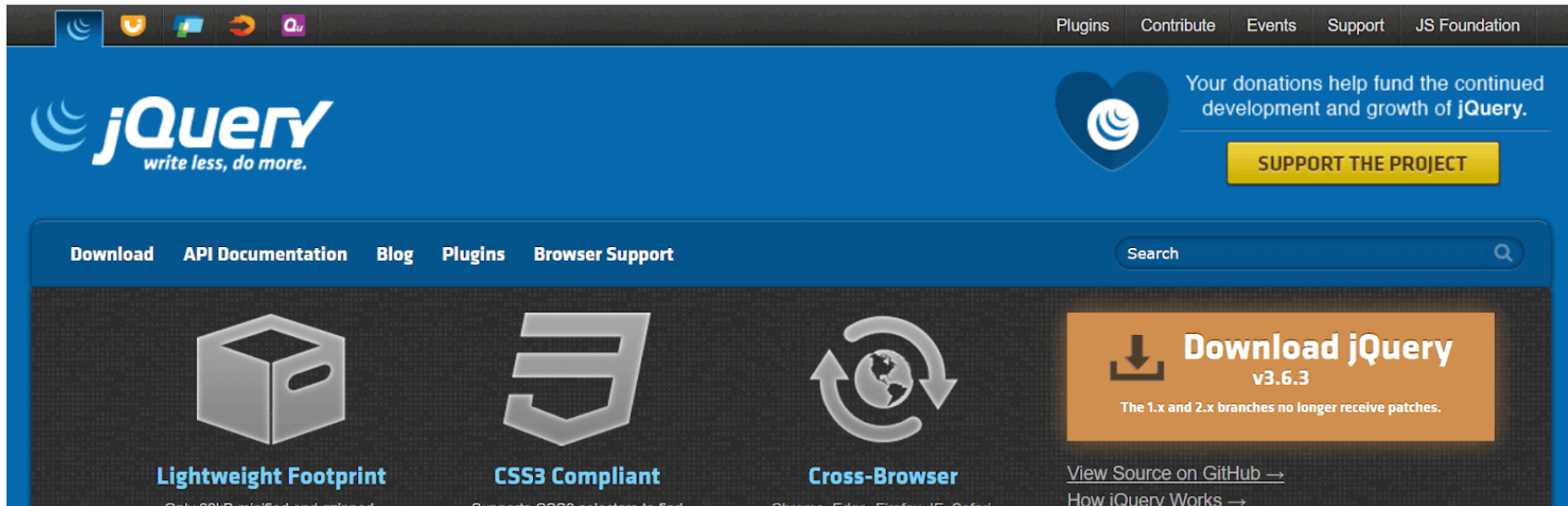
- **Learning curve** is relatively straightforward.
- Allows better performance due to the **server-side rendering** mode.
- Offers robust and **good-looking APIs**.
- **Optimizes code** during compilation; thus, overhead runtime reduces.

Front-End: Svelte

Drawbacks:

- The **architecture of compilers** moves complexity from source code and run time to tools and build time.
- Not suitable for **large projects**.

Front-End: jQuery



[jQuery](#) is a JavaScript library-based framework that makes adding dynamic and interactive elements to web pages more accessible. Released in 2006 and has become one of the most widely used JavaScript libraries on the web.

Front-End: jQuery

Benefits:

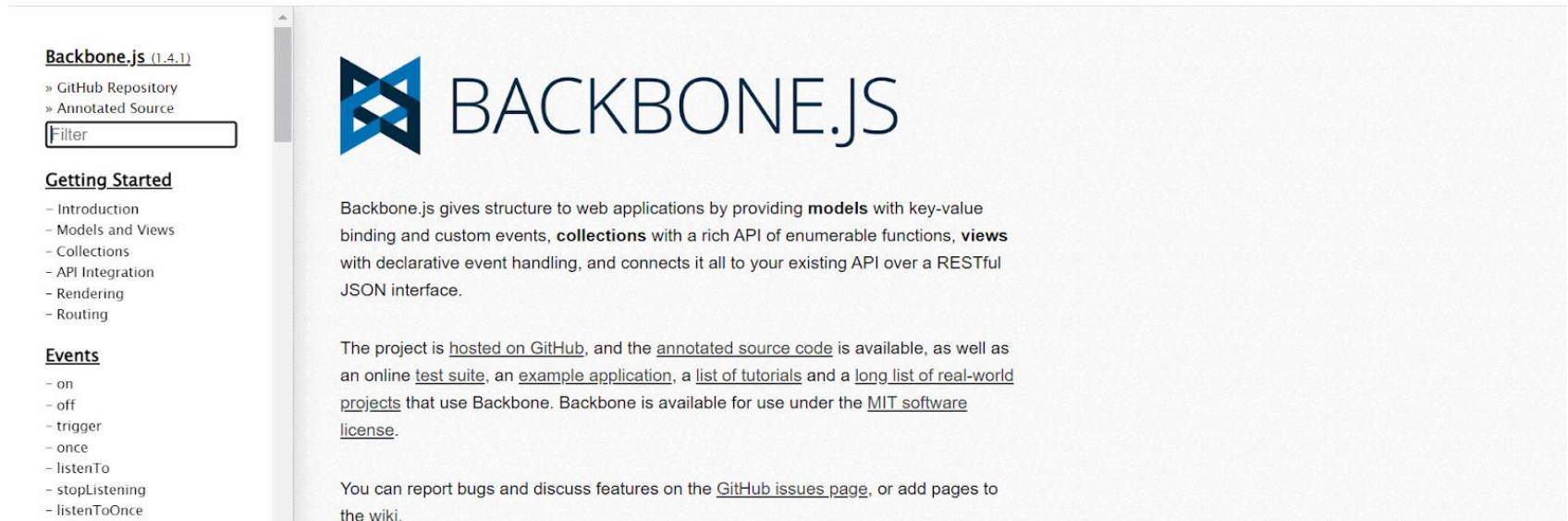
- With its **simple syntax** and a wide range of built-in methods makes it easy for developers to add dynamic behavior to web pages.
- The library's API is designed to be **intuitive and straightforward**, making it easier for developers to get started and make progress quickly.
- It has a wide range of **plugins and extensions** that can add additional functionality to web applications.

Front-End: jQuery

Drawbacks:

- jQuery is a JavaScript library, meaning its functionality **depends on JavaScript** being enabled in the user's browser. If JavaScript is disabled, the dynamic behavior provided by jQuery will not be available, and the user experience may be degraded.
- It **can be complex** in some cases, particularly for developers new to the library.

Front-End: Backbone.js



The screenshot shows the Backbone.js website. On the left is a navigation sidebar with the following items:

- Backbone.js (1.4.1)**
 - » GitHub Repository
 - » Annotated Source
- Filter
- Getting Started**
 - Introduction
 - Models and Views
 - Collections
 - API Integration
 - Rendering
 - Routing
- Events**
 - on
 - off
 - trigger
 - once
 - listenTo
 - stopListening
 - listenToOnce

The main content area features the Backbone.js logo (a blue geometric shape) and the text "BACKBONE.JS". Below the logo, the text reads: "Backbone.js gives structure to web applications by providing **models** with key-value binding and custom events, **collections** with a rich API of enumerable functions, **views** with declarative event handling, and connects it all to your existing API over a RESTful JSON interface."

Below this, it states: "The project is [hosted on GitHub](#), and the [annotated source code](#) is available, as well as an online [test suite](#), an [example application](#), a [list of tutorials](#) and a [long list of real-world projects](#) that use Backbone. Backbone is available for use under the [MIT software license](#)."

At the bottom, it says: "You can report bugs and discuss features on the [GitHub issues page](#), or add pages to the wiki."

[Backbone.js](#) is a lightweight front-end web development framework designed to provide structure to web applications. It is based on the Model-View-Controller (MVC) architecture pattern and allows developers to build dynamic and responsive user interfaces easily.

Front-End: Backbone.js

Benefits:

- Provides a structured way of **organizing code** and helps to separate concerns between data models, views, and controllers.
- It is a **lightweight** framework.
- **Flexible**, i.e., it allows developers to choose which components to use and how to use them.
- Compatible with different platforms, including **desktop and mobile**.
- Has a **large and active community** of developers who contribute to its development and provide support.

Front-End: Backbone.js

Drawbacks:

- While it is a lightweight framework, Backbone.js has a **steep learning curve** for beginners due to its complex architecture.
- Provides only basic functionalities; additional libraries may be required to add more features.
- It requires writing a lot of **boilerplate code**, which can be time-consuming.
- Lack of opinionated design can lead to **inconsistency and confusion** among developers.

Back-End: Node.js



Learn About Download Blog Docs ↗ Certification ↗

Search Start typing... Ctrl K



Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

```
Create an HTTP Server Write Tests Read and Hash a File Stre
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) => {
5   res.writeHead(200, { 'Content-Type': 'text/pl
6   res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally on port :
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
```

[Node.js](#) is an open-source, cross-platform JavaScript runtime environment that allows you to run JavaScript code outside of a web browser. In simpler terms, it lets you use JavaScript to build server-side applications and t

Back-End: Node.js

Benefits:

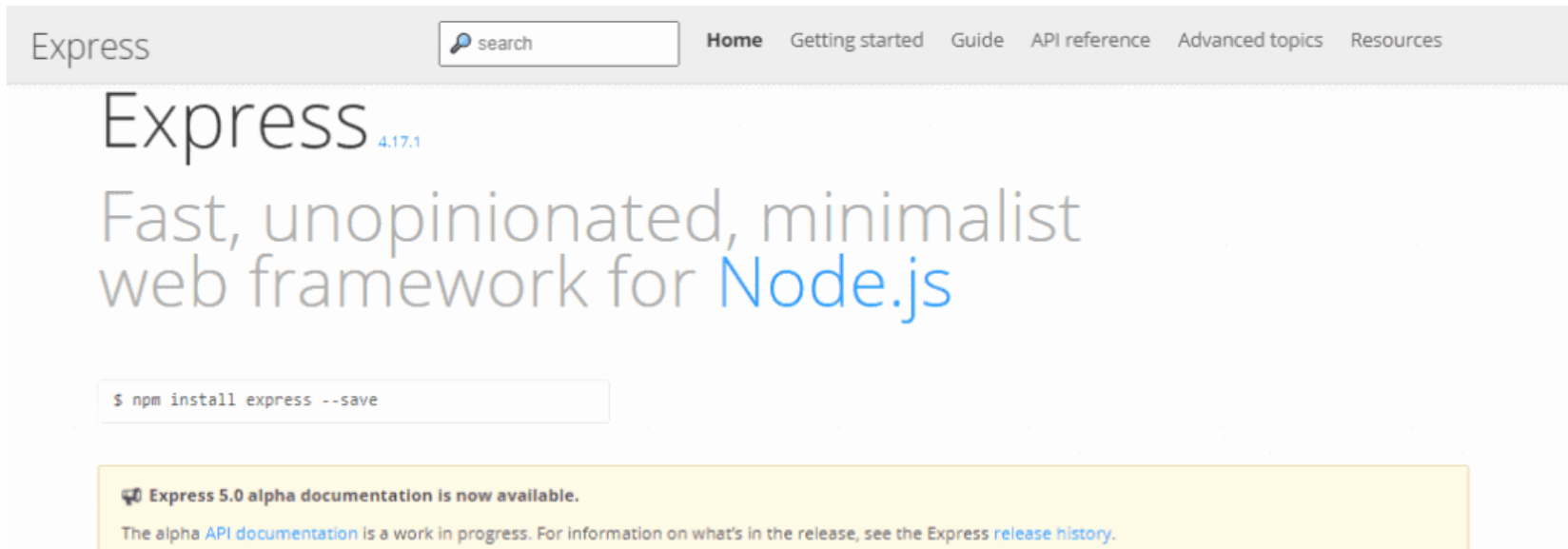
- JavaScript's familiarity and the availability of **pre-built packages** can speed up development.
- Excels at handling **many concurrent connections** due to its event-driven architecture. It's suitable for scalable web apps.
- Well-suited for building **real-time applications** that require constant two-way communication between server and client.
- With the rise of JavaScript frameworks like React or Angular, Node.js allows developers to use JavaScript for both front-end and back-end development, creating a more **unified development experience**.

Back-End: Node.js

Drawbacks:

- Not the best choice for applications that involve complex calculations or heavy data processing. It is **single-threaded**, slowing down the entire event loop and **impacting responsiveness**.
- Compared to some mature back-end languages, the built-in **standard library is relatively limited**. This is mitigated by the vast ecosystem of third-party packages available through npm (**Node Package Manager**).
- While Node.js is under active development, there have been some **changes to its API** in the past. These changes can require developers to adapt their code to newer versions.

Back-End: Express.js



[Express.js](https://expressjs.com/) is a framework for Node.js that includes a powerful routing system, which enables developers to create complex routes. It also includes a wide range of middleware allowing add authentication, logging, and error handling.

Back-End: Express.js

Benefits:

- One can **define the routes** of a preexisting app with the help of URL and HTTP.
- **Static files** and resources are easy to serve.
- A very advanced feature is creating a **REST API** server.
- Connected with **MongoDB, Redis** as well as **MySQL**.
- When used with Node.js, both frameworks can support **multiple concurrent actions**.
- With an open-sourced community, the code constantly gets **reviewed and improved**. And whenever you are stuck, you will get help from the other community users.

Back-End: Express.js

Drawbacks:

- When a developer is migrating from any different programming language, the **event-driven nature** of Express.js can be a little complex to comprehend.
- If you plan to develop a mobile app with Express.js, you must get familiar with the **concept of middleware**.
- The framework is perfect for medium and minor projects. If you aim to create a **large project**, a middleware framework like Express.js should not be made your choice.

Back-End: Django

django

The web framework for
perfectionists with deadlines.

[OVERVIEW](#) [DOWNLOAD](#) [DOCUMENTATION](#) [NEWS](#) [COMMUNITY](#) [CODE](#) [ISSUES](#) [ABOUT](#) [▼ DONATE](#)

Django makes it easier to build better Web
apps more quickly and with less code.

[Get started with Django](#)

[Django](#) is a Python-based Model-View-Template framework. It adheres to the Convention Over Configuration pattern and the DRY pattern. Main focus of the framework is on enablement of the secure websites creation.

Back-End: Django

Benefits:

- Offers the **best documentation**, available in multiple langs.
- There is **no need for prior experience** in the backend to build a fully functional website. It runs multiple files simultaneously without requiring the development of separate server files for database design.
- It provides **flexibility through pluggable apps**, through which third-party applications can be quickly plugged.
- Provides high **security by default**, closing loopholes previously left open for the backend developer to complete.

Back-End: Django

Drawbacks:

- Not supported for **highly scalable** applications.
- Django has a **steep learning curve**.
- **Monolithic** framework, i.e., has a limited number of dependencies, which makes it difficult to use.

Back-End: Ruby on Rails

[Blog](#)

[Guides](#)

[API](#)

[Forum](#)



[Foundation](#)

[Team](#)

[Contribute](#)

Compress the complexity of modern web apps.

Learn just what you need to get started, then keep leveling up as you go. **Ruby on Rails scales from HELLO WORLD to IPO.**

[Ruby on Rails](#), often simply called Rails, is an open-source framework. It emphasizes convention over configuration and follows the Model-View-Controller (MVC) architectural pattern. Also includes database migrations, RESTful routing, and scaffolding.

Back-End: Ruby on Rails

Benefits:

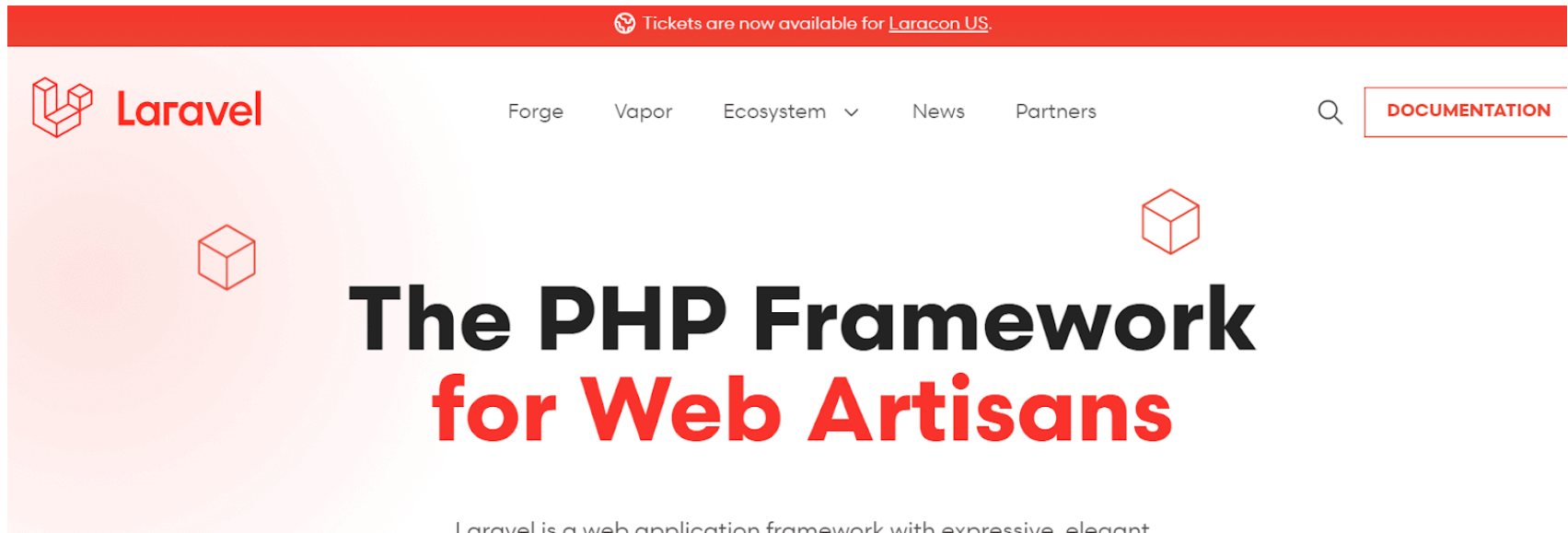
- Allows a **rapid development**, providing tools and abstractions, reducing effort required to build a functional application.
- Rails is designed to promote “***convention over configuration***”, meaning that it follows established conventions and best practices, without having to configure every aspect of the application from scratch.
- Offers a rich **ecosystem of gems** and plugins that can be used to add additional functionality to web applications.
- Highly **scalable**, making it a good choice for web applications that handle large amounts of traffic and data.

Back-End: Ruby on Rails

Drawbacks:

- The functionality **depends on Ruby** being installed on the server. It makes it more difficult to deploy Rails applications, particularly in environments where Ruby is not widely used or supported.
- Rails provides a full-stack framework, which includes **components for both the front-end and back-end** of the application. It can be a drawback for some developers, who may prefer to have more control over the different components of the application.

Back-End: Laravel



[Laravel](#) is a popular open-source PHP framework. It has a robust ecosystem of packages and tools, which makes it easy for developers to extend its features and functionality.

Back-End: Laravel

Benefits:

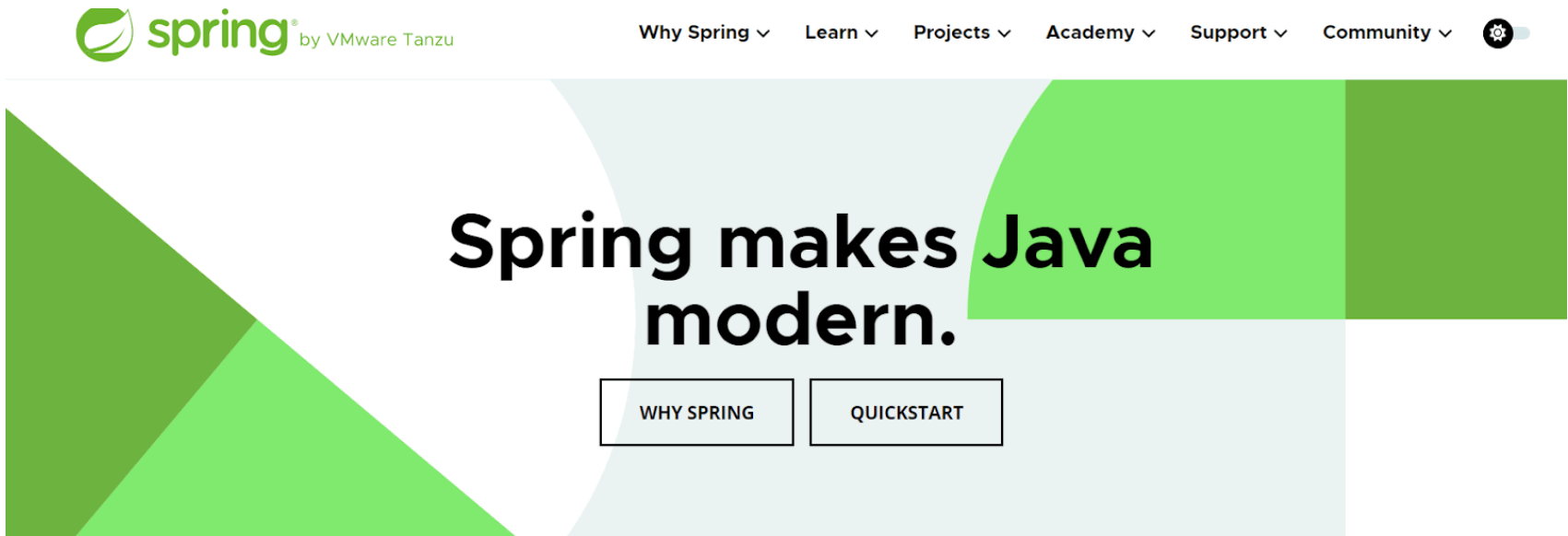
- Offers an easy-to-use **command-line interface** that makes it simple to perform common tasks, such as generating code and running database migrations.
- Offers a **templating engine** called Blade that makes creating and managing templates for your application easy.
- It includes an **Object-Relational Mapping (ORM)** tool called Eloquent, making it easy to interact with databases and perform database operations.
- It provides a simple and **flexible routing system** that makes it easy to define the URLs for your application.

Back-End: Laravel

Drawbacks:

- Relies heavily on **third-party packages** and extensions to provide additional functionality.
- Frequent releases of new versions make its **support limited**.
- Works on **complex architecture** difficult for developers to understand, especially those new to the framework.

Back-End: Spring



[Spring](#) is an open-source framework for building enterprise-level Java applications. It covers dependency injection, data access, and a modular design, making it an ideal choice for building scalable, robust, and maintainable applications.

Back-End: Spring

Benefits:

- An **IoC container** is responsible for managing the lifecycle of the application components. It makes it easy to build loosely coupled applications and reduces boilerplate code.
- Uses a **Model-View-Controller** (MVC) architecture.
- Spring provides a suite of **tools for testing** application, including support for unit testing, integration testing, and functional testing.
- It includes a **comprehensive security** framework that makes it easy to implement authentication, authorization, and other security features for your application.

Back-End: Spring

Drawbacks:

- Large and complex, and it can be **challenging for developers** to understand all its features and components.
- Relies heavily on **third-party libraries** and components, making it difficult to keep track of all the dependencies and ensure compatibility between different versions.
- It has a **large memory footprint**, which can be problematic for applications running on resource-constrained environments.

Check While Choosing a WDF

Choosing the right web development framework **can have a significant impact on the success** of your project. Consider:

- Ease of Installation and Maintenance
- Documentation and Support
- Licensing
- Scalability and Flexibility
- Cost and Budget
- Learning Curve
- Customization and Extensibility