Introduction

You are reading the documentation for Vue 3!

- Vue 2 support will end on Dec 31, 2023. Learn more about Vue 2 Extended LTS.
- Vue 2 documentation has been moved to v2.vuejs.org.
- Upgrading from Vue 2? Check out the Migration Guide.

What is Vue?

Vue (pronounced /vjuː/, like **view**) is a JavaScript framework for building user interfaces. It builds on top of standard HTML, CSS, and JavaScript and provides a declarative and component-based programming model that helps you efficiently develop user interfaces, be they simple or complex.

Here is a minimal example:

```
import { createApp } from 'vue'

createApp({
  data() {
    return {
      count: 0
    }
  }
}).mount('#app')
```

```
<div id="app">
  <button @click="count++">
    Count is: {{ count }}
  </button>
</div>
```

The above example demonstrates the two core features of Vue:

- Declarative Rendering: Vue extends standard HTML with a template syntax that allows us to declaratively describe HTML output based on JavaScript state.
- Reactivity: Vue automatically tracks JavaScript state changes and efficiently updates the DOM when changes happen.

You may already have questions - don't worry. We will cover every little detail in the rest of the documentation. For now, please read along so you can have a high-level understanding of what Vue offers.

Prerequisites

The rest of the documentation assumes basic familiarity with HTML, CSS, and JavaScript. If you are totally new to frontend development, it might not be the best idea to jump right into a framework as your first step - grasp the basics and then come back! You

can check your knowledge level with this JavaScript overview. Prior experience with other frameworks helps, but is not required.

The Progressive Framework

Vue is a framework and ecosystem that covers most of the common features needed in frontend development. But the web is extremely diverse - the things we build on the web may vary drastically in form and scale. With that in mind, Vue is designed to be flexible and incrementally adoptable. Depending on your use case, Vue can be used in different ways:

- Enhancing static HTML without a build step
- Embedding as Web Components on any page
- Single-Page Application (SPA)
- Fullstack / Server-Side Rendering (SSR)
- Jamstack / Static Site Generation (SSG)
- Targeting desktop, mobile, WebGL, and even the terminal

If you find these concepts intimidating, don't worry! The tutorial and guide only require basic HTML and JavaScript knowledge, and you should be able to follow along without being an expert in any of these.

If you are an experienced developer interested in how to best integrate Vue into your stack, or you are curious about what these terms mean, we discuss them in more detail in Ways of Using Vue.

Despite the flexibility, the core knowledge about how Vue works is shared across all these use cases. Even if you are just a beginner now, the knowledge gained along the way will stay useful as you grow to tackle more ambitious goals in the future. If you are a veteran, you can pick the optimal way to leverage Vue based on the problems you are trying to solve, while retaining the same productivity. This is why we call Vue "The Progressive Framework": it's a framework that can grow with you and adapt to your needs.