# **Installation**

### **Compatibility Note**

Vue does ****not**** support IE8 and below, because it uses ECMAScript 5 features that are un-shimmable in IE8. However it supports all **[ECMAScript 5 compliant browsers](https://caniuse.com/" \l "feat=es5" \t "https://v2.vuejs.org/v2/guide/_blank)**.

### **Semantic Versioning**

Vue follows **[Semantic Versioning](https://semver.org/" \t "https://v2.vuejs.org/v2/guide/_blank)** in all its official projects for documented features and behavior. For undocumented behavior or exposed internals, changes are described in **[release notes](https://github.com/vuejs/vue/releases" \t "https://v2.vuejs.org/v2/guide/_blank)**.

### **Release Notes**

Latest stable version: 2.6.14

Detailed release notes for each version are available on **[GitHub](https://github.com/vuejs/vue/releases" \t "https://v2.vuejs.org/v2/guide/_blank)**.

## **[Vue Devtools](https://v2.vuejs.org/v2/guide/installation.html" \l "Vue-Devtools" \o "Vue Devtools)**

When using Vue, we recommend also installing the **[Vue Devtools](https://github.com/vuejs/vue-devtools" \l "vue-devtools" \t "https://v2.vuejs.org/v2/guide/_blank)** in your browser, allowing you to inspect and debug your Vue applications in a more user-friendly interface.

## **[Direct <script> Include](https://v2.vuejs.org/v2/guide/installation.html" \l "Direct-lt-script-gt-Include" \o "Direct <script> Include)**

Simply download and include with a script tag. Vue will be registered as a global variable.

Don’t use the minified version during development. You will miss out on all the nice warnings for common mistakes!

**[Development Version](https://v2.vuejs.org/js/vue.js)**With full warnings and debug mode  
  
**[Production Version](https://v2.vuejs.org/js/vue.min.js)**Warnings stripped, 33.46KB min+gzip

### **[CDN](https://v2.vuejs.org/v2/guide/installation.html" \l "CDN" \o "CDN)**

For prototyping or learning purposes, you can use the latest version with:

<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>

For production, we recommend linking to a specific version number and build to avoid unexpected breakage from newer versions:

<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14"></script>

If you are using native ES Modules, there is also an ES Modules compatible build:

<script type="module">

import Vue from 'https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.esm.browser.js'</script>

You can browse the source of the NPM package at **[cdn.jsdelivr.net/npm/vue](https://cdn.jsdelivr.net/npm/vue/" \t "https://v2.vuejs.org/v2/guide/_blank)**.

Vue is also available on **[unpkg](https://unpkg.com/vue@2.6.14/dist/vue.js" \t "https://v2.vuejs.org/v2/guide/_blank)** and **[cdnjs](https://cdnjs.cloudflare.com/ajax/libs/vue/2.6.14/vue.js" \t "https://v2.vuejs.org/v2/guide/_blank)** (cdnjs takes some time to sync so the latest release may not be available yet).

Make sure to read about **[the different builds of Vue](https://v2.vuejs.org/v2/guide/installation.html" \l "Explanation-of-Different-Builds)** and use the ****production  
version**** in your published site, replacing vue.js with vue.min.js. This is a smaller build optimized for speed instead of development experience.

## **[NPM](https://v2.vuejs.org/v2/guide/installation.html" \l "NPM" \o "NPM)**

NPM is the recommended installation method when building large scale applications with Vue. It pairs nicely with module bundlers such as **[Webpack](https://webpack.js.org/" \t "https://v2.vuejs.org/v2/guide/_blank)** or **[Browserify](http://browserify.org/" \t "https://v2.vuejs.org/v2/guide/_blank)**. Vue also provides accompanying tools for authoring **[Single File Components](https://v2.vuejs.org/v2/guide/single-file-components.html)**.

# latest stable

$ npm install vue

## **[CLI](https://v2.vuejs.org/v2/guide/installation.html" \l "CLI" \o "CLI)**

Vue provides an **[official CLI](https://github.com/vuejs/vue-cli" \t "https://v2.vuejs.org/v2/guide/_blank)** for quickly scaffolding ambitious Single Page Applications. It provides batteries-included build setups for a modern frontend workflow. It takes only a few minutes to get up and running with hot-reload, lint-on-save, and production-ready builds. See **[the Vue CLI docs](https://cli.vuejs.org/" \t "https://v2.vuejs.org/v2/guide/_blank)** for more details.

The CLI assumes prior knowledge of Node.js and the associated build tools. If you are new to Vue or front-end build tools, we strongly suggest going through **[the guide](https://v2.vuejs.org/v2/guide/)** without any build tools before using the CLI.

**[Watch a video explanation on Vue Mastery](https://www.vuemastery.com/courses/real-world-vue-js/vue-cli" \o "Vue CLI" \t "https://v2.vuejs.org/v2/guide/_blank)**

## **[Explanation of Different Builds](https://v2.vuejs.org/v2/guide/installation.html" \l "Explanation-of-Different-Builds" \o "Explanation of Different Builds)**

In the **[dist/ directory of the NPM package](https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/" \t "https://v2.vuejs.org/v2/guide/_blank)** you will find many different builds of Vue.js. Here’s an overview of the difference between them:

|  | **UMD** | **CommonJS** | **ES Module (for bundlers)** | **ES Module (for browsers)** |
| --- | --- | --- | --- | --- |
| ****Full**** | vue.js | vue.common.js | vue.esm.js | vue.esm.browser.js |
| ****Runtime-only**** | vue.runtime.js | vue.runtime.common.js | vue.runtime.esm.js | - |
| ****Full (production)**** | vue.min.js | - | - | vue.esm.browser.min.js |
| ****Runtime-only (production)**** | vue.runtime.min.js | - | - | - |

### **[Terms](https://v2.vuejs.org/v2/guide/installation.html" \l "Terms" \o "Terms)**

****Full****: builds that contain both the compiler and the runtime.

****Compiler****: code that is responsible for compiling template strings into JavaScript render functions.

****Runtime****: code that is responsible for creating Vue instances, rendering and patching virtual DOM, etc. Basically everything minus the compiler.

**[UMD](https://github.com/umdjs/umd" \t "https://v2.vuejs.org/v2/guide/_blank)**: UMD builds can be used directly in the browser via a <script> tag. The default file from jsDelivr CDN at **[https://cdn.jsdelivr.net/npm/vue@2.6.14](https://cdn.jsdelivr.net/npm/vue@2.6.14" \t "https://v2.vuejs.org/v2/guide/_blank)** is the Runtime + Compiler UMD build (vue.js).

**[CommonJS](http://wiki.commonjs.org/wiki/Modules/1.1" \t "https://v2.vuejs.org/v2/guide/_blank)**: CommonJS builds are intended for use with older bundlers like **[browserify](http://browserify.org/" \t "https://v2.vuejs.org/v2/guide/_blank)** or **[webpack 1](https://webpack.github.io/" \t "https://v2.vuejs.org/v2/guide/_blank)**. The default file for these bundlers (pkg.main) is the Runtime only CommonJS build (vue.runtime.common.js).

**[ES Module](http://exploringjs.com/es6/ch_modules.html" \t "https://v2.vuejs.org/v2/guide/_blank)**: starting in 2.6 Vue provides two ES Modules (ESM) builds:

ESM for bundlers: intended for use with modern bundlers like **[webpack 2](https://webpack.js.org/" \t "https://v2.vuejs.org/v2/guide/_blank)** or **[Rollup](https://rollupjs.org/" \t "https://v2.vuejs.org/v2/guide/_blank)**. ESM format is designed to be statically analyzable so the bundlers can take advantage of that to perform “tree-shaking” and eliminate unused code from your final bundle. The default file for these bundlers (pkg.module) is the Runtime only ES Module build (vue.runtime.esm.js).

ESM for browsers (2.6+ only): intended for direct imports in modern browsers via <script type="module">.

### **[Runtime + Compiler vs. Runtime-only](https://v2.vuejs.org/v2/guide/installation.html" \l "Runtime-Compiler-vs-Runtime-only" \o "Runtime + Compiler vs. Runtime-only)**

If you need to compile templates on the client (e.g. passing a string to the template option, or mounting to an element using its in-DOM HTML as the template), you will need the compiler and thus the full build:

// this requires the compilernew Vue({

template: '<div>{{ hi }}</div>'

})

// this does notnew Vue({

render (h) {

return h('div', this.hi)

}

})

When using vue-loader or vueify, templates inside \*.vue files are pre-compiled into JavaScript at build time. You don’t really need the compiler in the final bundle, and can therefore use the runtime-only build.

Since the runtime-only builds are roughly 30% lighter-weight than their full-build counterparts, you should use it whenever you can. If you still wish to use the full build instead, you need to configure an alias in your bundler:

#### **Webpack**

module.exports = {

// ...

resolve: {

alias: {

'vue$': 'vue/dist/vue.esm.js' // 'vue/dist/vue.common.js' for webpack 1

}

}

}

#### **Rollup**

const alias = require('rollup-plugin-alias')

rollup({

// ...

plugins: [

alias({

'vue': require.resolve('vue/dist/vue.esm.js')

})

]

})

#### **Browserify**

Add to your project’s package.json:

{

// ...

"browser": {

"vue": "vue/dist/vue.common.js"

}

}

#### **Parcel**

Add to your project’s package.json:

{

// ...

"alias": {

"vue" : "./node\_modules/vue/dist/vue.common.js"

}

}

### **[Development vs. Production Mode](https://v2.vuejs.org/v2/guide/installation.html" \l "Development-vs-Production-Mode" \o "Development vs. Production Mode)**

Development/production modes are hard-coded for the UMD builds: the un-minified files are for development, and the minified files are for production.

CommonJS and ES Module builds are intended for bundlers, therefore we don’t provide minified versions for them. You will be responsible for minifying the final bundle yourself.

CommonJS and ES Module builds also preserve raw checks for process.env.NODE\_ENV to determine the mode they should run in. You should use appropriate bundler configurations to replace these environment variables in order to control which mode Vue will run in. Replacing process.env.NODE\_ENV with string literals also allows minifiers like UglifyJS to completely drop the development-only code blocks, reducing final file size.

#### **Webpack**

In Webpack 4+, you can use the mode option:

module.exports = {

mode: 'production'

}

But in Webpack 3 and earlier, you’ll need to use **[DefinePlugin](https://webpack.js.org/plugins/define-plugin/" \t "https://v2.vuejs.org/v2/guide/_blank)**:

var webpack = require('webpack')

module.exports = {

// ...

plugins: [

// ...

new webpack.DefinePlugin({

'process.env': {

NODE\_ENV: JSON.stringify('production')

}

})

]

}

#### **Rollup**

Use **[rollup-plugin-replace](https://github.com/rollup/rollup-plugin-replace" \t "https://v2.vuejs.org/v2/guide/_blank)**:

const replace = require('rollup-plugin-replace')

rollup({

// ...

plugins: [

replace({

'process.env.NODE\_ENV': JSON.stringify('production')

})

]

}).then(...)

#### **Browserify**

Apply a global **[envify](https://github.com/hughsk/envify" \t "https://v2.vuejs.org/v2/guide/_blank)** transform to your bundle.

NODE\_ENV=production browserify -g envify -e main.js | uglifyjs -c -m > build.js

Also see **[Production Deployment Tips](https://v2.vuejs.org/v2/guide/deployment.html)**.

### **[CSP environments](https://v2.vuejs.org/v2/guide/installation.html" \l "CSP-environments" \o "CSP environments)**

Some environments, such as Google Chrome Apps, enforce Content Security Policy (CSP), which prohibits the use of new Function() for evaluating expressions. The full build depends on this feature to compile templates, so is unusable in these environments.

On the other hand, the runtime-only build is fully CSP-compliant. When using the runtime-only build with **[Webpack + vue-loader](https://github.com/vuejs-templates/webpack-simple" \t "https://v2.vuejs.org/v2/guide/_blank)** or **[Browserify + vueify](https://github.com/vuejs-templates/browserify-simple" \t "https://v2.vuejs.org/v2/guide/_blank)**, your templates will be precompiled into render functions which work perfectly in CSP environments.

## **[Dev Build](https://v2.vuejs.org/v2/guide/installation.html" \l "Dev-Build" \o "Dev Build)**

****Important****: the built files in GitHub’s /dist folder are only checked-in during releases. To use Vue from the latest source code on GitHub, you will have to build it yourself!

git clone https://github.com/vuejs/vue.git node\_modules/vuecd node\_modules/vue

npm install

npm run build

## **[Bower](https://v2.vuejs.org/v2/guide/installation.html" \l "Bower" \o "Bower)**

Only UMD builds are available from Bower.

# latest stable

$ bower install vue

## **[AMD Module Loaders](https://v2.vuejs.org/v2/guide/installation.html" \l "AMD-Module-Loaders" \o "AMD Module Loaders)**

All UMD builds can be used directly as an AMD module.

**[Introduction](https://v2.vuejs.org/v2/guide/index.html)** →

Caught a mistake or want to contribute to the documentation? **[Edit this on GitHub!](https://github.com/vuejs/v2.vuejs.org/blob/master/src/v2/guide/installation.md" \t "https://v2.vuejs.org/v2/guide/_blank)**Deployed on **[Netlify](https://url.netlify.com/HJ8X2mxP8" \t "https://v2.vuejs.org/v2/guide/_blank)**.

# **Introduction**

## **What is Vue.js?**

Vue (pronounced /vjuː/, like ****view****) is a ****progressive framework**** for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with **[modern tooling](https://v2.vuejs.org/v2/guide/single-file-components.html)** and **[supporting libraries](https://github.com/vuejs/awesome-vue" \l "components--libraries" \t "https://v2.vuejs.org/v2/guide/_blank)**.

If you’d like to learn more about Vue before diving in, we **[created a video](https://v2.vuejs.org/v2/guide/index.html)** walking through the core principles and a sample project.

If you are an experienced frontend developer and want to know how Vue compares to other libraries/frameworks, check out the **[Comparison with Other Frameworks](https://v2.vuejs.org/v2/guide/comparison.html)**.

**[Watch a free video course on Vue Mastery](https://www.vuemastery.com/courses/intro-to-vue-js/vue-instance/" \o "Free Vue.js Course" \t "https://v2.vuejs.org/v2/guide/_blank)**

# **Template Syntax**

Vue.js uses an HTML-based template syntax that allows you to declaratively bind the rendered DOM to the underlying Vue instance’s data. All Vue.js templates are valid HTML that can be parsed by spec-compliant browsers and HTML parsers.

Under the hood, Vue compiles the templates into Virtual DOM render functions. Combined with the reactivity system, Vue is able to intelligently figure out the minimal number of components to re-render and apply the minimal amount of DOM manipulations when the app state changes.

If you are familiar with Virtual DOM concepts and prefer the raw power of JavaScript, you can also **[directly write render functions](https://v2.vuejs.org/v2/guide/render-function.html)** instead of templates, with optional JSX support.

## **[Interpolations](https://v2.vuejs.org/v2/guide/syntax.html" \l "Interpolations" \o "Interpolations)**