

# Creating a Vue Application

## The Application Instance

Every Vue application starts by creating a new **application instance** with the `createApp` function:

```
import { createApp } from 'vue'  
  
const app = createApp({  
  /* root component options */  
})
```

js

## The Root Component

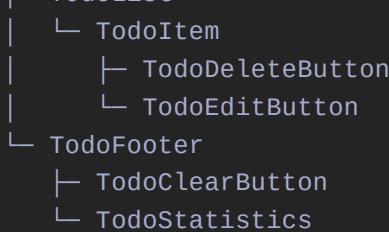
The object we are passing into `createApp` is in fact a component. Every app requires a "root component" that can contain other components as its children.

If you are using Single-File Components, we typically import the root component from another file:

```
import { createApp } from 'vue'  
// import the root component App from a single-file component.  
import App from './App.vue'  
  
const app = createApp(App)
```

js

While many examples in this guide only need a single component, most real applications are organized into a tree of nested, reusable components. For example, a Todo application's component tree might look like this:



In later sections of the guide, we will discuss how to define and compose multiple components together. Before that, we will focus on what happens inside a single component.

## Mounting the App

An application instance won't render anything until its `.mount()` method is called. It expects a "container" argument, which can either be an actual DOM element or a selector string:

```
<div id="app"></div>
```

html

```
app.mount('#app')
```

js

The content of the app's root component will be rendered inside the container element. The container element itself is not considered part of the app.

The `.mount()` method should always be called after all app configurations and asset registrations are done. Also note that its return value, unlike the asset registration methods, is the root component instance instead of the application instance.

## In-DOM Root Component Template

The template for the root component is usually part of the component itself, but it is also possible to provide the template separately by writing it directly inside the mount container:

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

html

```
import { createApp } from 'vue'

const app = createApp({
  data() {
    return {
      count: 0
    }
  }
})
```

js

```
app.mount('#app')
```

Vue will automatically use the container's `innerHTML` as the template if the root component does not already have a `template` option.

In-DOM templates are often used in applications that are [using Vue without a build step](#). They can also be used in conjunction with server-side frameworks, where the root template might be generated dynamically by the server.

## App Configurations

The application instance exposes a `.config` object that allows us to configure a few app-level options, for example, defining an app-level error handler that captures errors from all descendant components:

```
app.config.errorHandler = (err) => {  
  /* handle error */  
}
```

js

The application instance also provides a few methods for registering app-scoped assets. For example, registering a component:

```
app.component('TodoDeleteButton', TodoDeleteButton)
```

js

This makes the `TodoDeleteButton` available for use anywhere in our app. We will discuss registration for components and other types of assets in later sections of the guide. You can also browse the full list of application instance APIs in its [API reference](#).

Make sure to apply all app configurations before mounting the app!

## Multiple Application Instances

You are not limited to a single application instance on the same page. The `createApp` API allows multiple Vue applications to co-exist on the same page, each with its own scope for configuration and global assets:

```
)  
app1.mount('#container-1')  
  
const app2 = createApp({  
  /* ... */  
})  
app2.mount('#container-2')
```

If you are using Vue to enhance server-rendered HTML and only need Vue to control specific parts of a large page, avoid mounting a single Vue application instance on the entire page. Instead, create multiple small application instances and mount them on the elements they are responsible for.

 [Edit this page on GitHub](#)

[← Previous](#)

[Next →](#)

[Quick Start](#)

[Template Syntax](#)