

Components.

1. Create a New Component

Start by creating a new component file in your `src/components` directory. For example, create a file called `HelloWorld.vue`.

```
<!-- src/components/HelloWorld.vue -->
<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>

<style scoped>
1 reference
.hello {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

2. Register the Component

In the parent component where you want to use this new component, you need to import and register it. This is typically done in the `App.vue` file or any other parent component.

```

<!-- src/App.vue -->
<template>
  <div id="app">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>

<style>
/* Your styles here */
</style>

```

3. Use the Component in the Template

Once registered, you can use the custom component as a custom HTML tag in the parent component's template.

```

<!-- src/App.vue -->
<template>
  <div id="app">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

```

4. Passing Props

Props are a way to pass data from a parent component to a child component. In the example above, `msg` is a prop being passed to the `HelloWorld` component.

```

<!-- src/components/HelloWorld.vue -->
<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
  </div>
</template>



<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>

```

5. Scoped Styles

In the `HelloWorld.vue` component, the `<style scoped>` tag ensures that the styles apply only to this component and do not leak out to other parts of the application.

```

<style scoped>
1 reference
.hello {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  text-align: center;
  color:   #2c3e50;
  margin-top: 60px;
}
</style>

```

6. Reusing Components

Once a component is created, you can reuse it in multiple places by importing and registering it in any parent component.

Example with Multiple Components

If you have multiple components, such as `Header.vue` and `Footer.vue`, you can import and use them similarly.

```
vue Copy code

<!-- src/components/Header.vue -->
<template>
  <header>
    <h1>Header Component</h1>
  </header>
</template>

<script>
export default {
  name: 'Header'
}
</script>

<!-- src/components/Footer.vue -->
<template>
  <footer>
    <p>Footer Component</p>
  </footer>
</template>

<script>
export default {
  name: 'Footer'
}
</script>
```

Then, use them in `App.vue`:

```
vue Copy code

<!-- src/App.vue -->
<template>
  <div id="app">
    <Header />
    <HelloWorld msg="Welcome to Your Vue.js App"/>
    <Footer />
  </div>
</template>

<script>
import Header from './components/Header.vue'
import HelloWorld from './components/HelloWorld.vue'
import Footer from './components/Footer.vue'

export default {
  name: 'App',
  components: {
    Header,
    HelloWorld,
    Footer
  }
}
</script>
```

Summary

1. **Create a Component:** Write a `.vue` file for your component.
2. **Register the Component:** Import and register the component in a parent component.
3. **Use the Component:** Use the custom tag of your component in the parent component's template.
4. **Pass Props:** Send data to your components using props.
5. **Style Scoped:** Use scoped styles to keep component styles encapsulated.

By following these steps, you can effectively create and use components in Vue.js to build a modular and maintainable application.