

Vue.js Components

Topics : [Vue](#)

Written on [January 11, 2024](#)

In Vue.js, components are the building blocks of a Vue application. They allow you to create modular and reusable pieces of user interface and logic. Components encapsulate a specific functionality or feature, making it easier to manage and maintain complex applications. Here's a basic overview of Vue.js components:

1. Creating a Component:

To create a component in Vue.js, you define it using the `Vue.component` method or by creating a component object with a template, data, and other options.

Example using `Vue.component`:

```
<template>
<div>
<h1>{{ greeting }}</h1>
</div>
</template>

<script>
Vue.component('my-component', {
  data() {
    return {
      greeting: 'Hello, Vue!'
    };
  }
});
</script>
```

Example using a component object:

```
<template>
<div>
<h1>{{ greeting }}</h1>
</div>
</template>

<script>
export default {
  data() {
    return {
```

```
greeting: 'Hello, Vue!'
};
}
};
</script>
```

2. Using Components:

Once you've defined a component, you can use it within another component or the main application.

```
<template>
<div>
<my-component></my-component>
</div>
</template>

<script>
import MyComponent from './MyComponent.vue';

export default {
  components: {
    'my-component': MyComponent
  }
};
</script>
```

3. Component Options:

Components in Vue.js can have various options, such as:

- **data:** A function that returns the component's data.
- **computed:** Computed properties that are derived from the component's data.
- **methods:** Methods that can be called from the template or elsewhere.
- **props:** Properties passed from parent components.
- **watch:** Watchers that observe changes to specific data properties.

4. Communication Between Components:

Components in Vue.js can communicate with each other using props, events, and a central event bus. Props allow you to pass data from parent to child components, while events enable child components to emit events to notify parent components of changes.

Example using props:

```
<!-- Parent Component -->
<template>
<div>
<child-component :message="parentMessage"></child-component>
</div>
</template>

<script>
import ChildComponent from './ChildComponent.vue';
```

```

export default {
  components: {
    'child-component': ChildComponent
  },
  data() {
    return {
      parentMessage: 'Message from parent'
    };
  }
};
</script>

```

```

<!-- Child Component -->
<template>
  <div>
    <p>{{ message }}</p>
  </div>
</template>

```

```

<script>
export default {
  props: ['message']
};
</script>

```

5. Single File Components:

Vue.js supports Single File Components (SFC) that encapsulate the template, script, and style in a single .vue file. This makes the structure of a component more organized and maintainable.

Example of a Single File Component:

```

<template>
  <div>
    <h1>{{ greeting }}</h1>
  </div>
</template>

<script>
export default {
  data() {
    return {
      greeting: 'Hello, Vue!'
    };
  }
};
</script>

<style scoped>
/* Component-specific styles go here */
</style>

```

ARYATECHNO