SCSJ 3483: Web Technology

Introduction To VueJS

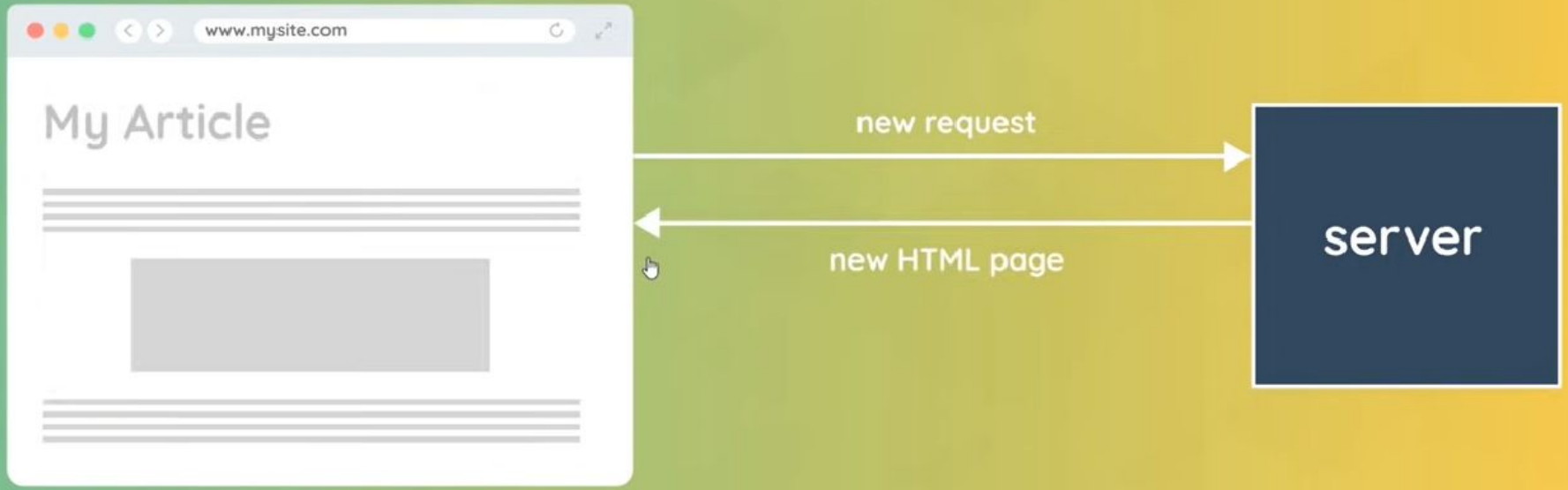
Intro to Vue.js

- A progressive JavaScript **front-end framework** used for building user interface.
- Known for its simplicity and flexibility
- Used to create dynamic and data-driven website (SPA). All routing in browser
- Can also be used to create standalone widget

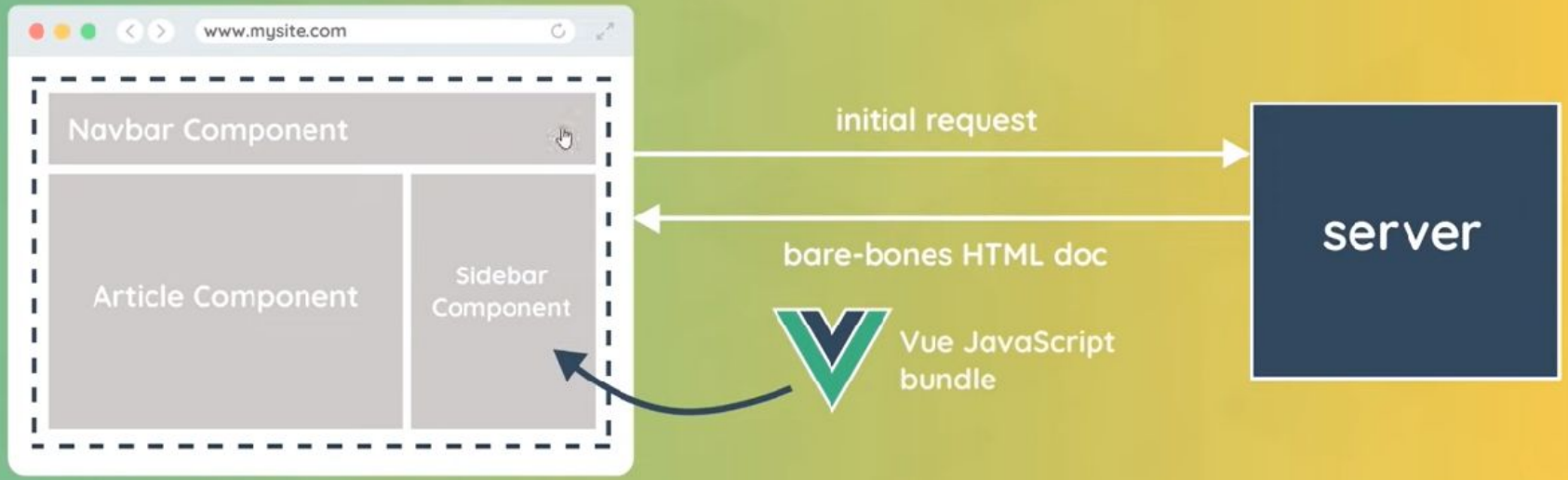
Vue Widgets


Vue Component
Signup Form
Become a member..
Get our free newsletter..

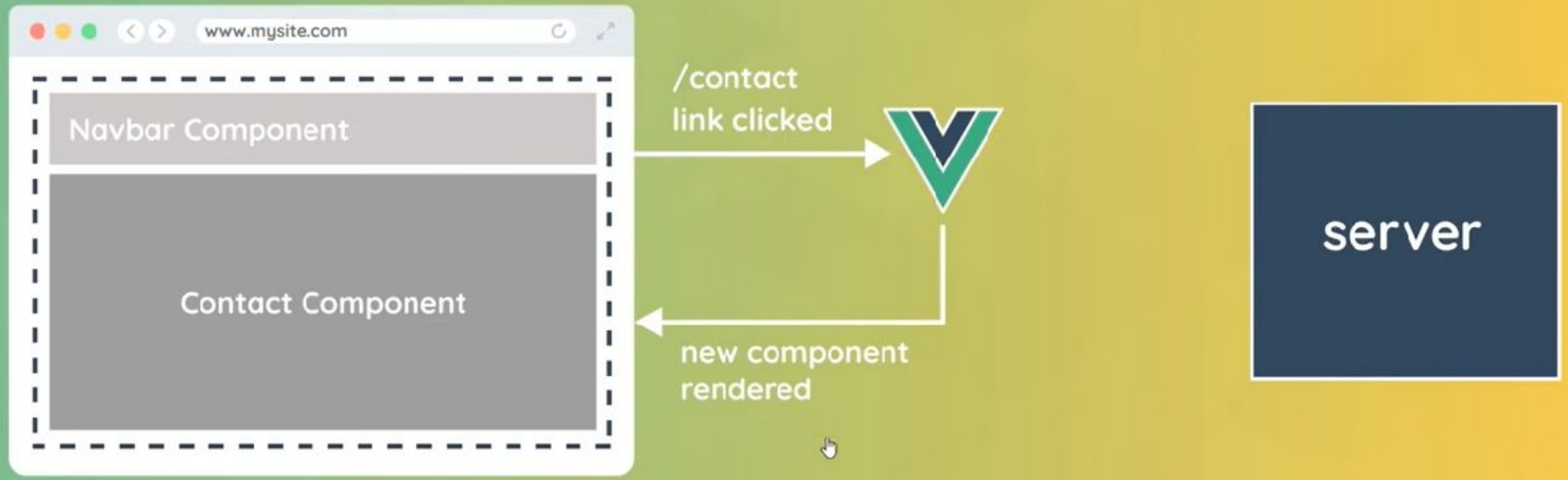
Typical non-Vue Websites



Vue Websites



Vue Websites



Single Page Application (SPA)

- Only a single HTML page sent (initially) to the browser
- Vue intercepts subsequent requests and handles "page" changes in the browser by swapping what components are shown on the page
- Results in a much faster and smoother website experience

Vue Basic/Core Concepts:

- **Directives, data binding, methods, and computed properties**
- **Vue Components**: a reusable Vue instances, composed together to build complex UIs.
- **Vue Router:** for handling navigation. Allow app to have multiple routes.
- **API Integration:** Learn how to make HTTP requests and handle responses

Directives

Vue.js uses directives to bind HTML elements to data in the Vue instance

Directives are prefixed with **v-**, and they are special attributes added to HTML elements

Examples : **v-bind, v-model, v-for, v-if, v-else**

Vue Directive

Directive	Description	Example Usage
v-bind	Binds an attribute to an expression.	<a v-bind:href="url">Link
v-model	Creates a two-way binding on a form input element or a component.	<input v-model="message">
v-if	Conditionally renders an element based on the truthiness of the expression.	Visible
v-else	Renders if the preceding v-if expression evaluates to false.	<div v-else>Not Visible
v-else-if	Similar to v-else, but used for additional conditions.	Other

Vue Directive (cont...)

Directive	Description	Example Usage
v-for	Renders a list of items based on an array.	<li v-for="item in items">{{ item }}
v-on	Listens to DOM events and triggers methods or expressions.	<button v-on:click="handleClick">Click</button>
v-show	Toggles the visibility of an element based on the truthiness of the expression.	Visible

Data Binding

- Data binding allows us to bind data from the Vue instance to the DOM
- Two types of data binding: **Interpolation** and **Property Binding**
- Interpolation : using {{ }}. example, **{{ message }}**
- Property binding : using v-bind. example, **v-bind:href="url"**

Event Handling

Vue Data Binding

Data Binding Technique	Description	Example Usage
Interpolation	Renders data from the Vue instance in the DOM using double curly braces {{ }}.	{{ message }}
v-bind	Dynamically binds an attribute to an expression in the Vue instance.	<a v-bind:href="url">Link
v-model	Creates a two-way binding on form input elements or components.	<input v-model="message">
v-on	Listens to DOM events and triggers methods or expressions in response.	<button v-on:click="handleClick">Click </button>
Computed Properties	Reactive properties that are calculated based on other reactive data in the Vue	computed: { fullName() { return this.firstName + ' ' +

v-bind

```
    <!-- Dynamically bind href attribute using v-bind -->
    <a :href="url">Visit Google</a>

    <!-- Dynamically bind style attribute using v-bind -->
    <div :style="{ color: textColor, backgroundColor: bgColor }">
        This div has dynamic styles.
    </div>

    <!-- Dynamically bind class attribute using v-bind -->
    <button :class="{ active: isActive, disabled: isDisabled }">Click me</button>
</div>

<script>
    const app = Vue.createApp({
        data() {
            return {
                url: 'https://www.google.com',
                textColor: 'blue',
                bgColor: 'lightgray',
                isActive: true,
                isDisabled: false
            };
        }
    });
```

v-model

```
<div id="app">
    <!-- Two-way binding with v-model -->
    <input type="text" v-model="message">
    <p>You typed: {{ message }}</p>

    <!-- Two-way binding with v-model on a textarea -->
    <textarea v-model="textareaContent"></textarea>
    <p>You wrote: {{ textareaContent }}</p>

    <!-- Two-way binding with v-model on a checkbox -->
    <input type="checkbox" v-model="isChecked">
    <p>Checkbox is checked: {{ isChecked }}</p>
</div>

<script>
    const app = Vue.createApp({
        data() {
            return {
                message: '',
                textareaContent: '',
                isChecked: false
            };
        }
    });
```

Methods

- Vue.js allows for the definition of methods within the Vue instance, which can then be invoked from the template or other methods.
- Methods reside within the methods property of the Vue instance.
- Example:

```
methods: {
  greet() {
    alert('Hello!');
  }
}
```

- **greet()** can be invoked using v-on directive or event handlers.

Computed Properties

- resemble method, but with caching based on their dependencies.
- Defined within the computed property of the Vue instance.
- Example:

```
computed: {
  fullName() {
    return this.firstName + ' ' + this.lastName;
  }
}
```

- Thus, **fullName** recalculates only when firstName or lastName changes.

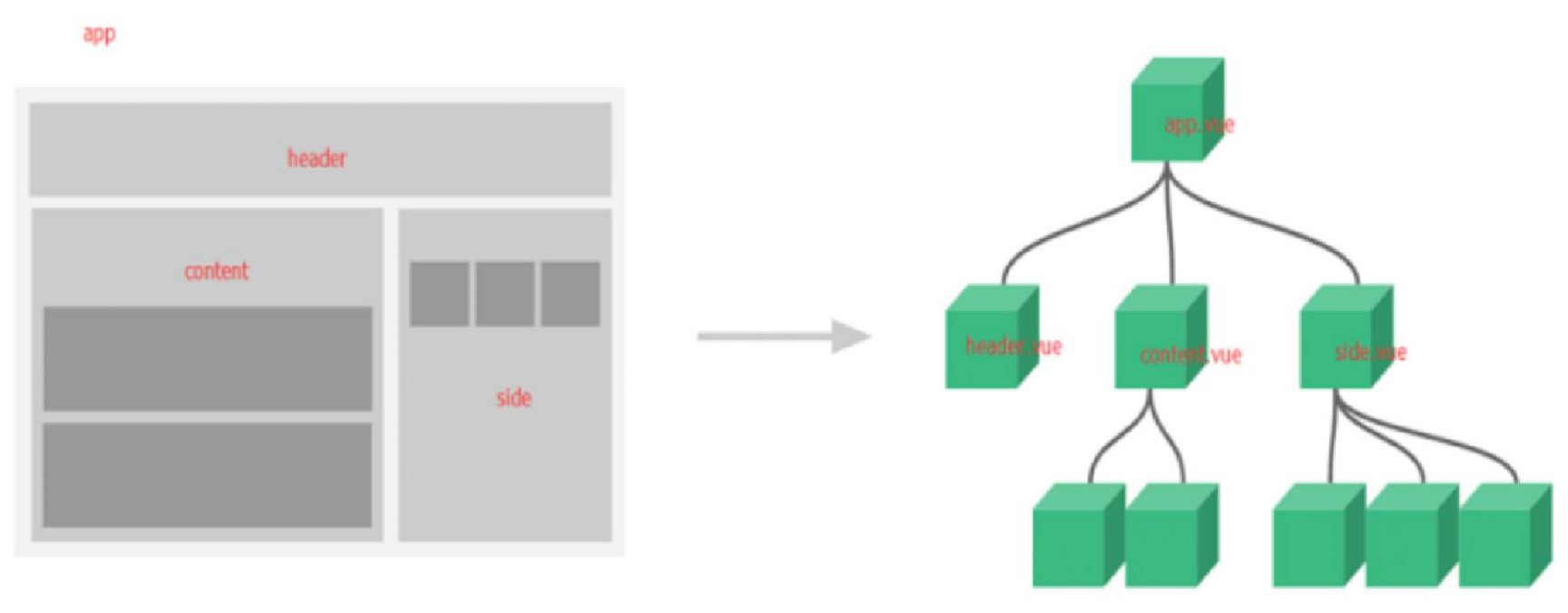
Vue Components

- Components : reusable, self-contained units of vue application, encapsulating HTML, CSS, and JavaScript logic.
- Allow developers to break down complex UIs into smaller, manageable pieces, promoting reusability and maintainability.
- In Vue, components typically defined using Vue.component method, or by creating a Vue instance with the Vue.extend method.
- Example:

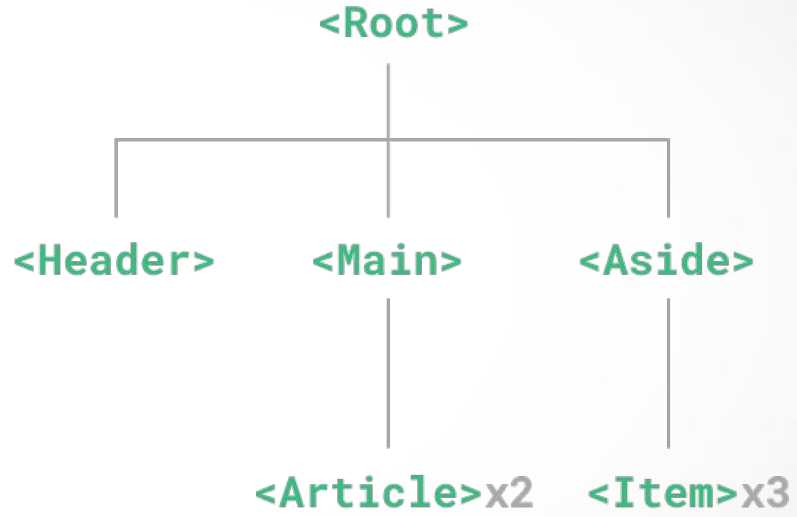
```
Vue.component('my-component', {
  // Component options (data, methods, etc.)
});
```

- Component Lifecycle: created, mounted, updated

Vue Components



Vue Components



Props

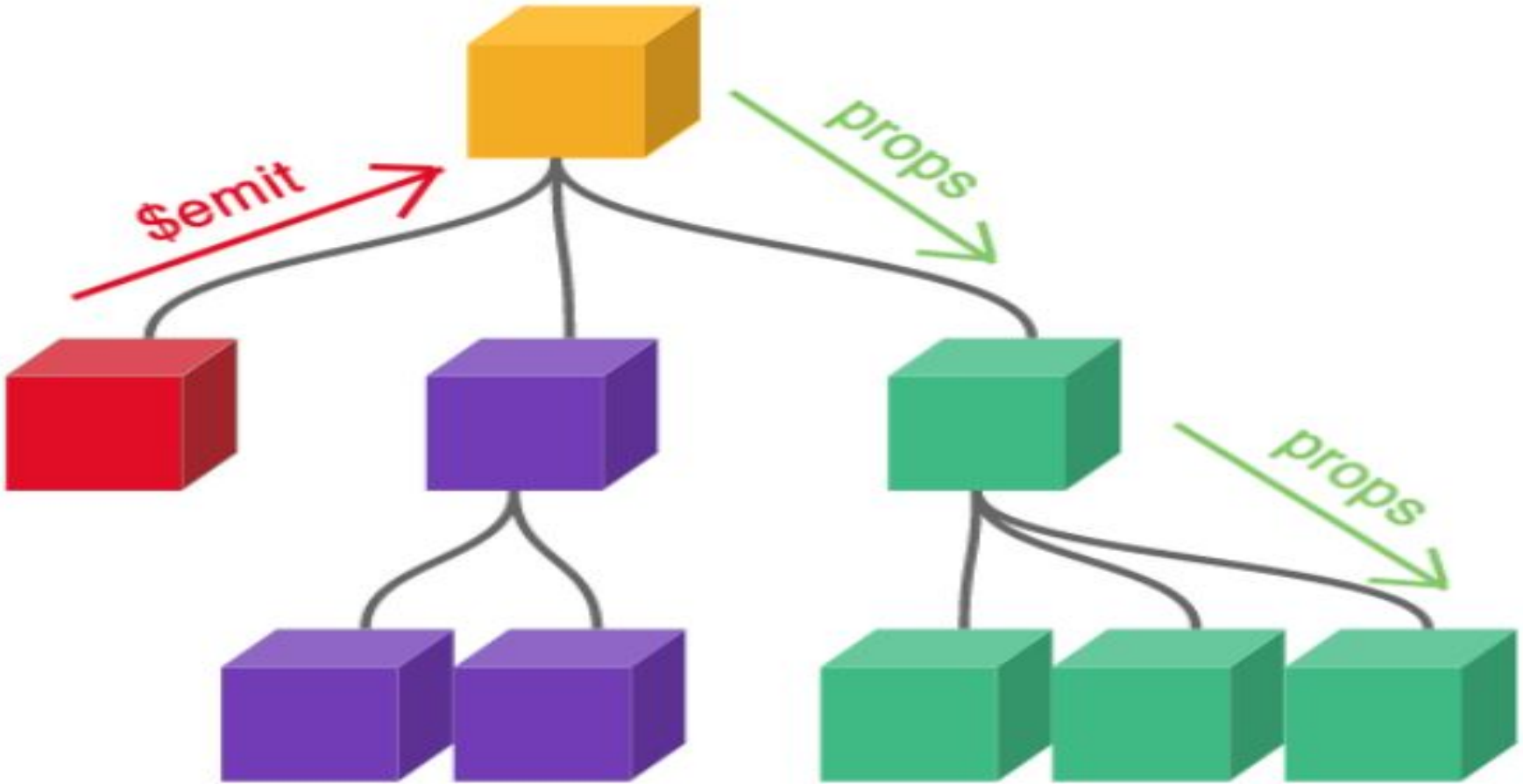
- A custom attributes that can be passed to a component, allowing data to be passed from the parent component to the child component.
- Defined in the parent component's template when using the component, and accessible within the child component.
- Example:

```
<!-- Parent component -->
<child-component :prop-name="data"></child-component>

// Child component
Vue.component('child-component', {
  props: ['propName'],
  // Component options
});
```

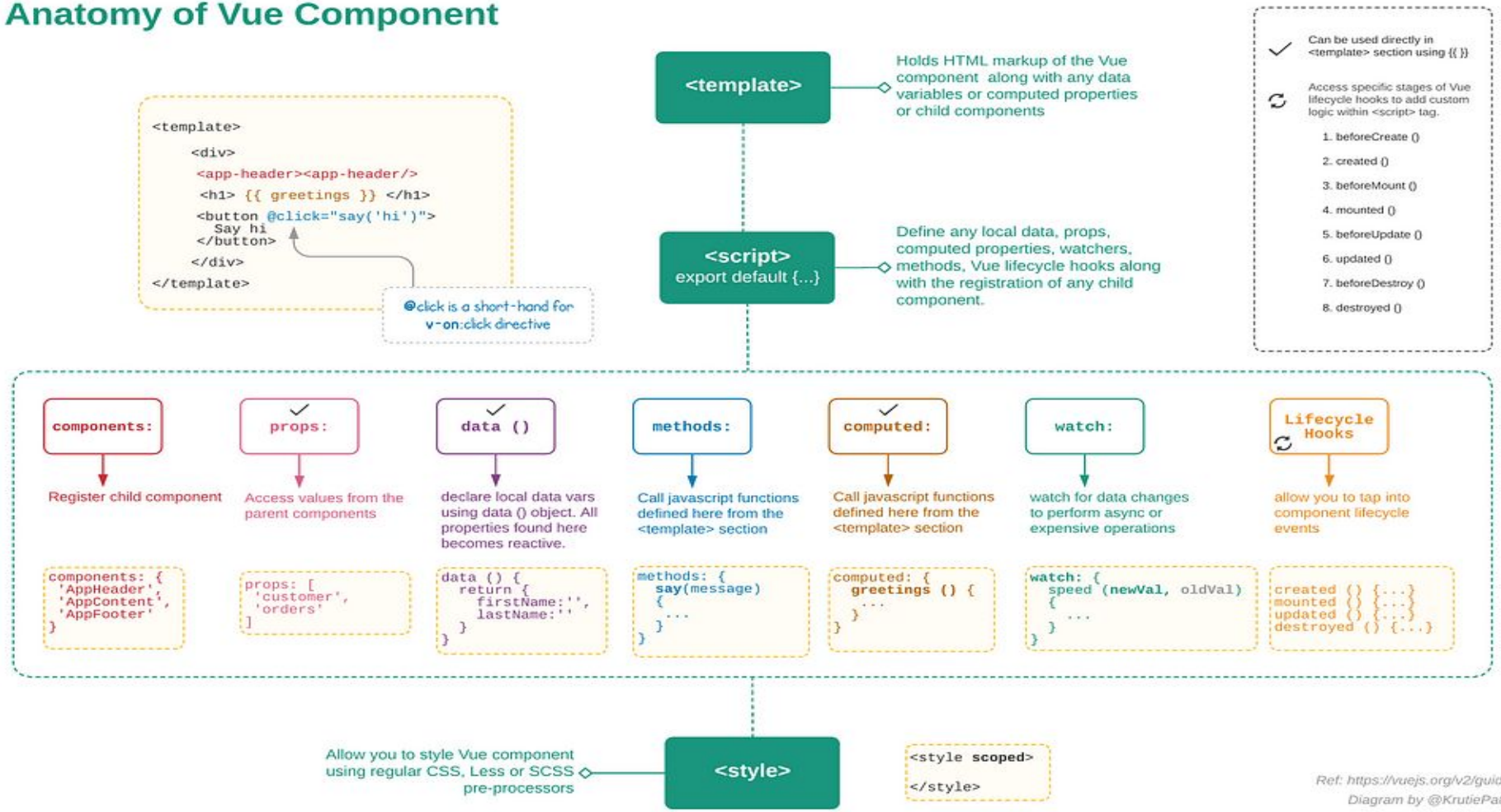
Events

- Child communicate with parent using custom events.
- Child emit events using **this.$emit**, and parent listen to these events using **v-on**.

```
// Child component emits an event
this.$emit('event-name', eventData);

<!-- Parent component listens to the event -->
<child-component v-on:event-name="handleEvent"></child-component>
```


$emit
props
props

Anatomy of Vue Component



Ref: https://vuejs.org/v2/guide/
Diagram by @KrutiePatel
As of Vue 2.6

Next...

- Vue Router:
  - enables navigation between different views in a Vue.js application
  - synchronizes application's state with URL, allowing for SPA experience without page reloads
- API Integration:
  - involves making HTTP requests to external APIs to fetch or send data
  - several ways to achieve API integration in Vue.js, but one of the most common approaches is to use libraries like Axios or Vue Resource