Vue.js Стартовий

Основи роботи з компонентами: methods, computed

Introduction

Кінаш Станіслав
Front-end dev

stanislav.kinash

stanislav.kinash

MCID: 9210561

Тема уроку

Основи роботи з компонентами: methods, computed

План уроку

1. Методи у Vue.js.
2. Що таке computed?
3. Різниця між computed і methods.
4. Приклад застосування.

Методи у Vue.js

Vue.js надає ряд методів, які допомагають управляти даними та елементами інтерфейсу вашої програми.

Методи у Vue.js

Ось кілька з найбільш популярних методів у Vue.js:

`data`: використовується для визначення даних, які застосовуються у вашому додатку. Ви можете використовувати
метод для визначення даних в об'єкті, який буде доступний у всьому додатку.

`watch`: використовується для відстежування змін властивостей даних та виконання відповідних дій, коли вони
змінюються. Наприклад, ви можете відстежувати зміни у полі введення і виконувати певні дії, коли значення змінюється.

`created`: викликається, коли Vue.js створює інстанс компонента. Ви можете використовувати цей метод для виконання певних дій при створенні компонента, наприклад, завантаження даних з сервера.

`mounted`: викликається, коли компонент Vue.js вставляється в DOM.

Методи у Vue.js

`computed`: метод дозволяє обчислювати дані на основі вхідних даних. Він буде автоматично оновлюватись, якщо змінюються вхідні дані. Це зручно використовувати, коли потрібно обчислити значення на основі даних, які змінюються.

`methods`: метод використовується для визначення функцій, які можуть викликатися з інтерфейсу. Ви можете використовувати цей метод для визначення функцій, які будуть виконуватися при натисканні кнопки або зміні значень.

Що таке computed?

У Vue.js стан програми представлений об'єктом JavaScript під назвою екземпляр Vue. Цей екземпляр містить усі дані та методи, необхідні для надання програми. Vue.js використовує цю інформацію для обчислення стану програми.

Наприклад, скажімо, у вас є програма, яка відображає список книг. Ви можете представити цей список як масив об'єктів, де кожен об’єкт містить назву, автора та дату публікації книги. Ви можете створити екземпляр Vue, який виглядає так:

```
const app = new Vue({
  el: '#app',
  data: {
    books: [
      { title: 'The Great Gatsby', author: 'F. Scott Fitzgerald', pubDate: '1925' },
      { title: 'To Kill a Mockingbird', author: 'Harper Lee', pubDate: '1960' },
      { title: '1984', author: 'George Orwell', pubDate: '1949' }
    ]
  }
})
```

Що таке computed?

У цьому прикладі властивість даних екземпляра Vue містить масив об'єктів, що представляють список книг. Vue.js використовуватиме ці дані для обчислення стану програми та відповідно оновлювати подання.

Щоб відобразити цей список книг у вікнах програми, ви можете використовувати V- для директиви в шаблоні:

```
<div id="app">
    <ul>
    <li v-for="book in books">
        {{ book.title }} by {{ book.author }} ({{ book.pubDate }})
    </li>
    </ul>
</div>
```

Що таке computed?

Ця директива v-for буде повторюватись над масивом книг у властивість даних екземпляра Vue та надасть елемент списку для кожної книги у масиві. Синтаксис {{ }} використовується для інтерполяції назви, автора та дати публікації книги в шаблон.

Коли дані екземпляра Vue змінюються, Vue.js автоматично оновлюватиме подання, щоб відобразити ці зміни. Наприклад, якщо додати нову книгу до списку:

```
app.books.push({ title: 'Pride and Prejudice',
                 author: 'Jane Austen',
                 pubDate: '1813' })
```

Що таке computed?

Vue.js автоматично оновлюватиме список книг у перегляді, щоб включити нову книгу, не вимагаючи додаткового коду. Це ядро системи реактивності Vue.js, де рамки автоматично оновлюють відображення, коли стан програми змінюється.

Різниця між computed і methods.

У Vue.js як **`computed`** властивості, так і **`methods`** використовуються для визначення функціональності екземпляра Vue, але між ними є деякі важливі відмінності.

Різниця між computed і methods.

Обчислені властивості (`**computed**`) – це функції, які використовуються для обчислення та повернення значення на основі інших властивостей або даних в екземплярі Vue. Обчислені властивості кешуються на основі їхніх залежностей, тому вони оновлюються лише тоді, коли змінюється одна з їхніх залежностей. Це робить їх корисними для дорогих або складних обчислень, які в іншому випадку потрібно було б виконувати щоразу, коли компонент повторно відтворюється.

З іншого боку, методи (`**methods**`) — це функції, які визначені в екземплярі Vue і можуть бути викликані безпосередньо з шаблону або з інших методів. Методи не кешуються та переоцінюються щоразу, коли компонент повторно відтворюється. Це робить їх корисними для більш динамічної або інтерактивної функціональності, яка потребує оновлення кожного разу, коли оновлюється компонент.

Різниця між computed і methods.

У цьому прикладі обчислена властивість **fullName** кешується і буде переоцінена лише тоді, коли зміниться **firstName** або **lastName**.

З іншого боку, метод **greet** переоцінюється щоразу, коли його викликають, тому, якщо ми викличемо його кілька разів, він щоразу перераховуватиме **fullName**.

```
const app = new Vue({
    el: '#app',
    data: {
      firstName: 'John',
      lastName: 'Doe',
    },
    computed: {
      fullName: function() {
        return this.firstName + ' ' + this.lastName;
      }
    },
    methods: {
      greet: function() {
        return 'Hello, ' + this.fullName + '!';
      }
    }
  })
```

Різниця між computed і methods.

Загалом, слід використовувати **computed** для обчислень, які є дорогими або складними, чи які потрібно кешувати з міркувань.

Ви повинні використовувати **methods** для більш динамічної або інтерактивної функціональності, яку потрібно оновлювати щоразу, коли компонент оновлюється.

Приклад застосування

Computed Example

Припустімо, у нас є програма Vue.js, яка відображає список товарів, і кожен товар має кількість і ціну. Ми хочемо відобразити загальну вартість усіх елементів у списку. Можемо використовувати **computed**, щоб обчислити загальну вартість на основі кількості та ціни кожного товару.

Приклад застосування

У цьому прикладі ми визначаємо обчислену властивість під назвою **totalCost**, яка обчислює загальну вартість усіх елементів у масиві items.

Ми використовуємо метод reduce, щоб переглянути масив елементів і обчислити загальну вартість на основі кількості та ціни кожного товару.

Обчислена властивість автоматично оновлюватиметься щоразу, коли змінюватиметься масив елементів.

```
const app = new Vue({
    el: '#app',
    data: {
      items: [
        { name: 'Item 1', quantity: 2, price: 10 },
        { name: 'Item 2', quantity: 3, price: 15 },
        { name: 'Item 3', quantity: 1, price: 5 }
      ]
    },
    computed: {
      totalCost: function() {
        return this.items.reduce((total, item) =>
            total + (item.quantity * item.price), 0);
      }
    }
  })
```

Приклад застосування

Відобразимо результат на екран:

```
<div id="app">
    <ul>
        <li v-for="item in items">
            {{ item.name }} - {{ item.quantity }} x ${{ item.price }} = ${{ item.quantity * item.price }}
        </li>
    </ul>
    <p>Total cost: ${{ totalCost }}</p>
</div>
```

Приклад застосування

Methods Example

Припустімо, у нас є форма, яка дозволяє користувачеві вводити своє ім’я та адресу електронної пошти, і ми хочемо перевірити адресу електронної пошти, щоб переконатися, що це дійсний формат. Ми можемо використати **methods** для виконання перевірки та повернення логічного значення, яке вказує, чи дійсний електронний лист.

Приклад застосування

```
const app = new Vue({
    el: '#app',
    data: {
      name: '',
      email: ''
    },
    methods: {
      validateEmail: function(email) {
        const regex = /\S+@\S+\.\S+/;
        return regex.test(email);
      }
    }
  })
```

У цьому прикладі ми визначаємо **method** під назвою **validateEmail**, який приймає адресу електронної пошти як аргумент і використовує регулярний вираз (Regex), щоб перевірити, чи електронний лист має дійсний формат. Метод повертає логічне значення, яке вказує, чи дійсний електронний лист.

Приклад застосування

Потім ми можемо використати метод у нашому шаблоні, щоб відобразити повідомлення про те, чи дійсний електронний лист:

```
<div id="app">
  <form>
    <label>
      Name:
      <input type="text" v-model="name">
    </label>
    <label>
      Email:
      <input type="text" v-model="email">
    </label>
    <p v-if="email && !validateEmail(email)">
        Please enter a valid email address.
    </p>
    <button type="submit">Submit</button>
  </form>
</div>
```

У цьому прикладі ми використовуємо директиву v-if для умовного відображення повідомлення, якщо електронний лист недійсний. Ми передаємо властивість даних електронної пошти методу validateEmail як аргумент, і метод повертає логічне значення, яке вказує, чи дійсна електронна адреса. Якщо електронна адреса недійсна, буде показано повідомлення; інакше нічого не відображатиметься.

Таким чином, ми можемо використовувати метод у Vue.js для виконання спеціальної перевірки та відображення повідомлення користувачеві на основі результату.


ITVDN
IT VIDEO DEVELOPERS NETWORK


TestProvider
Pearson
CyberBionic
systematics
Microsoft Partner
Windows Azure
EBA

Перевірка знань

TestProvider.com

TestProvider – це online-сервіс перевірки знань з інформаційних технологій. За його допомогою ви можете оцінити свій рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT-спеціаліста.

Перевірте, як ви засвоїли даний матеріал на TestProvider.com

Успішне проходження фінального тестування дозволить отримати відповідний Сертифікат.

Vue.js Стартовий

Дякую за увагу! До нових зустрічей!

Кінаш Станіслав
Front-end dev

MCID: 9210561