Client-Side Development Topics

Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Based on MDN Web Docs, developer.mozilla.org

Outline

- Client-Side Frameworks
- Vue.js Overview
- Client-Side Tools

Client-Side Frameworks

Client-Side Frameworks

- Software frameworks are designed to speed-up software development by providing pre-configured selection of libraries and configurations on how to use specific technologies.
- Frameworks exist for HTML, CSS, and JavaScript.
  - **HTML frameworks** include https://html5boilerplate.com.
  - **CSS frameworks** include getbootstrap.com, get.foundation, tailwindcss.com.
  - **JavaScript frameworks** include angularjs.org, reactjs.org, svelte.dev, vuejs.org.

JavaScript Frameworks

- Modern web applications make extensive use of JavaScript (used on ~95% of all web sites).
- The central problem that JavaScript frameworks address is
  - managing application state and synchronizing the user interface.
- JavaScript frameworks typically support:
  - Developer tools and pipelines;
  - Client-side routing;
  - Client-side templating, e.g. Handlebars;
  - Domain-specific languages, e.g. TypeScript;
  - Components, e.g. JSX (JavaScript and XML);
  - Dependency management.

Vue.js Example

Vue.js

- Vue.js is a client-side JavaScript framework.
- The initial release was in 2014 and is published under a MIT License.
- Vue.js is written in TypeScript and provides first-class TypeScript support.
- Compared to other JavaScript frameworks, Vue.js can be used to enhance existing HTML — providing facilities for progressive enhancement.

Installation

- Vue.js can be installed from a CDN.
  - From: unpkg, jsdelivr, cdnjs
  - Latest version
    - https://unpkg.com/vue
  - Specific versions
    - https://unpkg.com/vue@3
  - Production version
    - https://unpkg.com/vue@3/dist/vue.global.prod.js

- Or locally, using `npm` and `create-vue`, Vue's scaffolding tool.

Without Build Step (plug-in library)

Hello World

```
<!DOCTYPE html>
<html>
    <head>
        <title>Vue.js Hello World</title>
        <script src="https://unpkg.com/vue"></script>
    </head>
    <body>
        <div id="app">{{ message }}</div>
    </body>

    <script>
        const { createApp } = Vue

        createApp({
          data() {
            return {
              message: 'Hello Vue!'
            }
          }
        }).mount('#app')
    </script>
</html>
```

Binding User Data and Forms

- `v-model` is a directive that creates a two-way data binding between a value in the template and the data properties — when data changes, the input will change; if input changes, the data will change.

```
<body>
  <div id="app">
    <label for="name">Name:</label>
    <input id="name" type="text" v-model="name" />
    <p>Name variable: {{ name }}</p>
    <p>Again the name var: {{ name }}</p>
  </div>
</body>
```

```
<script>
  const { createApp } = Vue

  createApp({
    data() {
      return {
        name: ''
      }
    }
  }).mount('#app')

</script>
```

Binding Properties Manipulation

- Element properties can changed with the `v-bind` directive.
- The class property can be changed with `v-bind:class` or the shorthand `:class`.
- Class `red` will be present if the `active` property is `true`.

```
<style>
  .red {
    color: red;
  }
</style>


<body>
  <div id="app">
    <button @click="active = !active">Toggle me</button>
    <p :class="{ red: active }">Paragraph with toggling styles.</p>
  </div>
</body>
```

```
<script>
  const { createApp } = Vue

  createApp({
    data() {
      return {
        active: false
      }
    }
  }).mount('#app')
</script>
```

Event Handling

- `v-on`, or the shortcut `@`, is used to capture DOM events and run JavaScript code.
  - `v-on:click="handler"`
  - `@click="handler"`

```
<body>
  <div id="app">
    <button @click="count++">Add +1</button>
    <p>Count is currently {{ count }}</p>
  </div>
</body>
```

```
<script>
  const { createApp } = Vue

  createApp({
    data() {
      return {
        count: 0
      }
    }
  }).mount('#app')
</script>
```

Other Directives

- Conditional rendering:
  - `v-if` `v-else` `v-else-if`

```
<button @click="awesome = !awesome">Toggle</button>

<h1 v-if="awesome">Vue is awesome!</h1>
<h1 v-else>Oh no!</h1>
```

- List rendering:
  - `v-for`

```
data() {
    return {
        items: [{ message: 'Foo' }, { message: 'Bar' }]
    }
}
```

```
<li v-for="item in items">
    {{ item.message }}
</li>
```

Components

- Vue.js components can be used to build modular interfaces.

```
<script type="module">
  import SpecialParagraph from "./SpecialParagraph.js";

  const app = Vue.createApp({
    components: {SpecialParagraph},
    template: `
    <h1>Components</h1>
    <SpecialParagraph></SpecialParagraph>
    `
  }).mount("#app");
</script>
```

```
export default {
  data() {
    return {
      msg: "Hello world from component"
    }
  },
  template: `
<p>
  <strong>{{msg}}</strong>
</p>
`
}
```

SpecialParagraph.js

Client-Side Tools

Developer Tools

- Developer tools play an essential role in modern development environments.
- Tools can be structured in three broad categories:
  - Tools design to support **code development**.
  - Tools to **transform code** between different representations (CSS, JavaScript).
  - Tools for **testing or deployment**, after code is written.

Code Development

- **Source code control**, for backup and team work (e.g. Git).
- **Browser developer tools**, for code inspection and debugging.
- **Linters** check through the code to highlight existing errors and guidelines violations.
  - ESlint, for JavaScript, https://eslint.org
  - csslint, for CSS, http://csslint.net
- **Bundlers/Packages** prepare the code for production, e.g. remove non-used code, minify code for deployment.
  - Parcel, https://parceljs.org
  - Webpack, https://webpack.js.org

Code Transformation

- Code transformation tools allow developers to use
  - the latest language features, i.e. “future code”,
  - other development languages, e.g. TypeScript.
- Transformation tools will generate browser-compatible code to be used in production.
- There are code transformation tools for HTML, CS and JavaScript.

Use Latest Language Features

- Write code using the latest language features and have that code transformed into code that works in older devices.

- **Babel**, is a JavaScript compiler that transforms code using the latest JavaScript features to old JavaScript code, thus supported in more browsers.
  - https://babeljs.io
- **PostCSS**, is a similar tool for CSS, i.e. transforms modern CSS to older browser-supported CSS code.
  - https://postcss.org

Use Another Language

- Write code in a different language and transform it into a web standard language.

- **Sass/SCSS**, a stylesheet language that is compiled to CSS.
  - https://sass-lang.com
- **TypeScript**, a superset of JavaScript that offers additional features.
  - https://www.typescriptlang.org

Testing and Development

- **Testing tools** can automatically run tests against the code before moving further in the production pipeline (e.g. unit testing).
  - Popular tools: www.travis-ci.com, www.jenkins.io

- **Deployment tools** are used to automatically publish a web application.
  - Popular tools: pages.github.com

References

- MDN Web Docs, developer.mozilla.org
  - Introduction to client-side frameworks, MDN Web Docs
  - Getting started with Vue, MDN Web Docs
- Vue.js
  - vuejs.org
  - Vue.js Guide, vuejs.org/guide/introduction.html
  - vue-community.org