University of Miskolc
Faculty of Mechanical Engineering and Informatics

Web Front -end Full Stack Development

N13020104

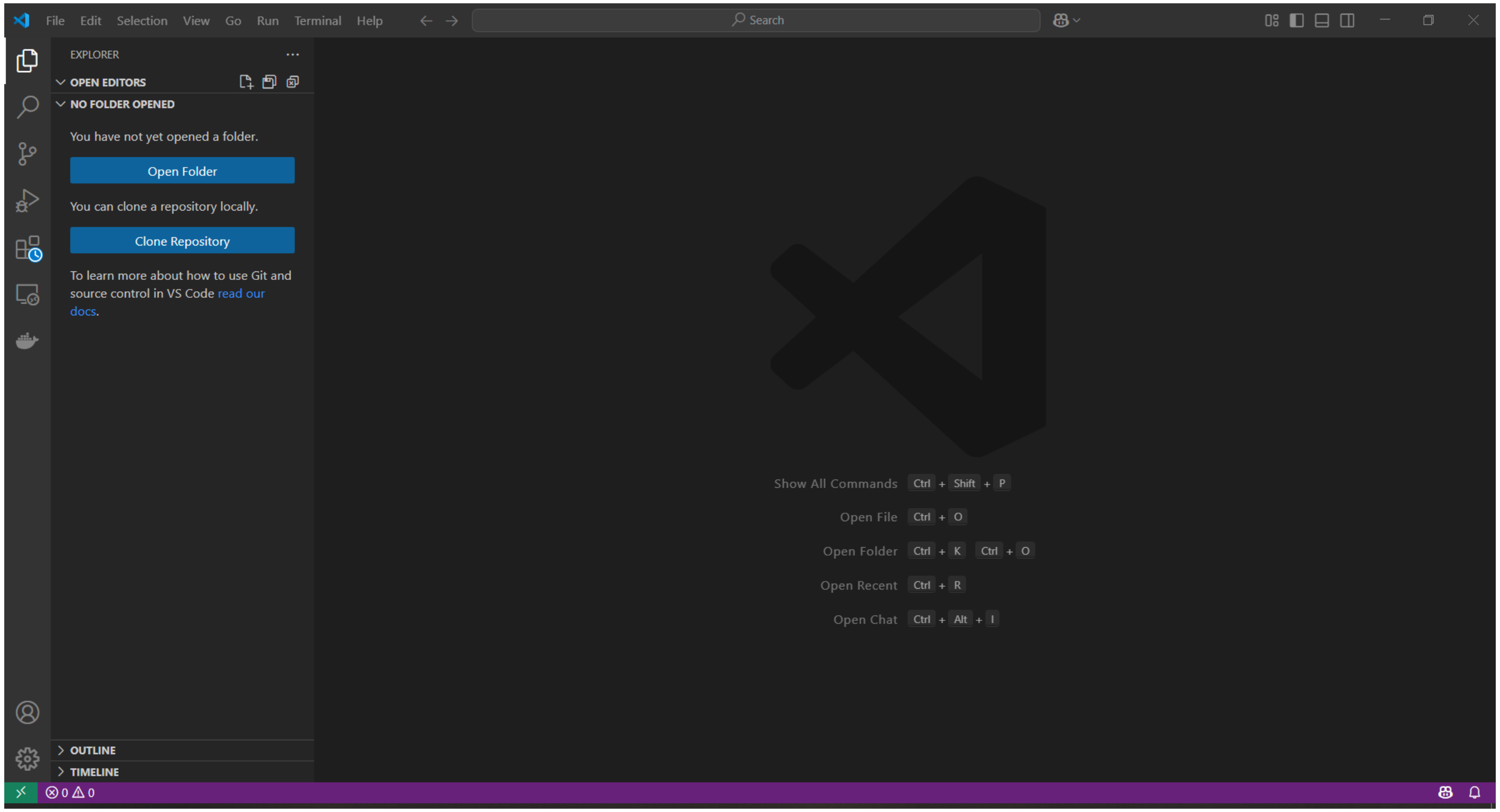
VueJS advanced

Tamás Tompa, PhD
assistant professor
Department of Information Technology
University of Miskolc

2025.

Setup

- VS Code Editor
  - download: https://code.visualstudio.com/download


File Edit Selection View Go Run Terminal Help

Search

EXPLORER

OPEN EDITORS

NO FOLDER OPENED

You have not yet opened a folder.

Open Folder

You can clone a repository locally.

Clone Repository

To learn more about how to use Git and source control in VS Code read our docs.

Show All Commands Ctrl + Shift + P

Open File Ctrl + O

Open Folder Ctrl + K Ctrl + O

Open Recent Ctrl + R

Open Chat Ctrl + Alt + I

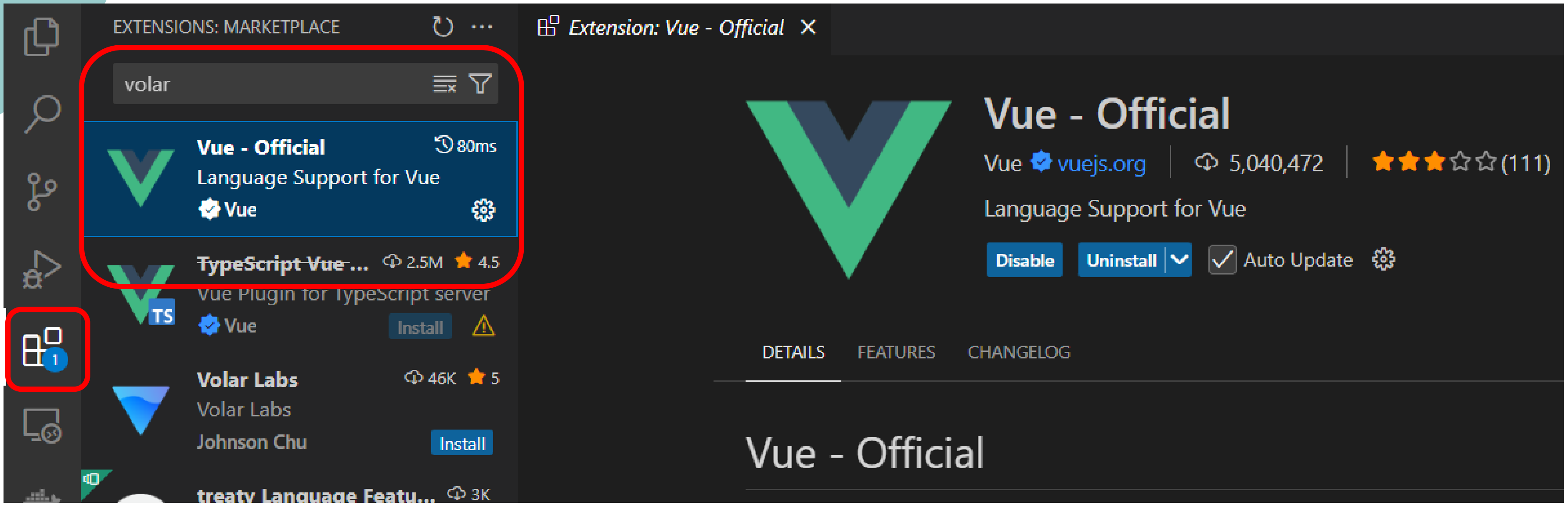
OUTLINE

TIMELINE

⊗ 0 ⚠ 0

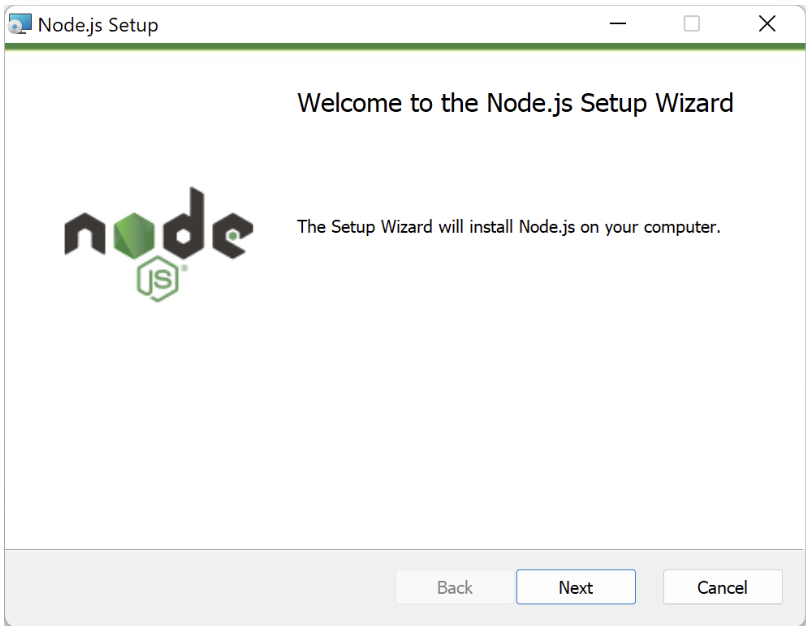
Setup

- VS Code Editor
  - VS Code "Volar" Extension



Install Node.js - NPM

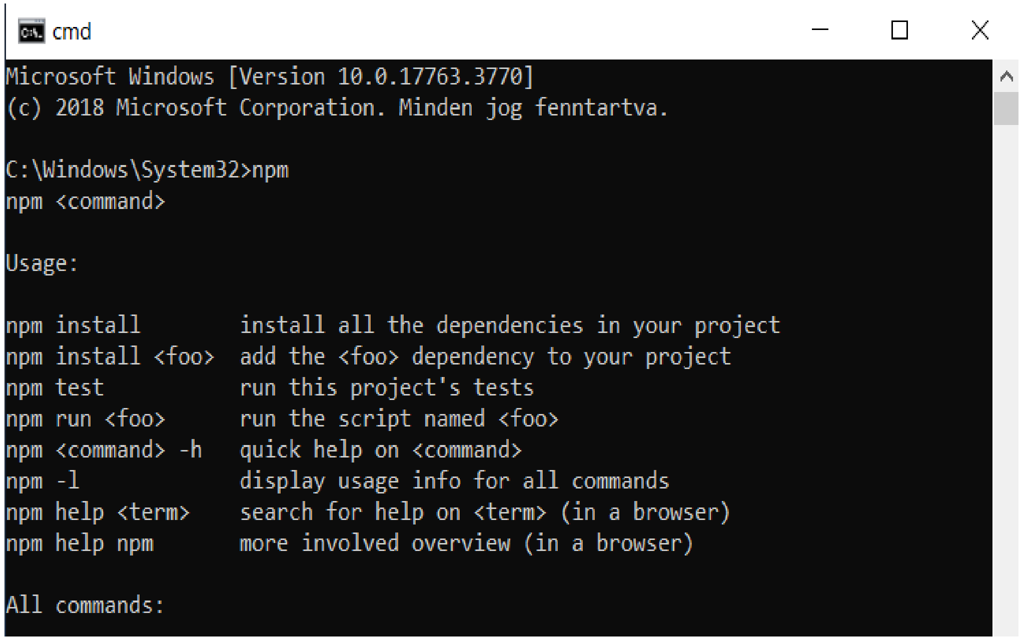
- **Install the Node.js to use the NPM**
  - https://nodejs.org/en/download (use the Windows installer)
  - **npm** is the **standard package manager** for Node.js
  - if a project has a package.json file can be use the `npm install` command to install/download all dependecies of the project



Node.js Setup

Welcome to the Node.js Setup Wizard

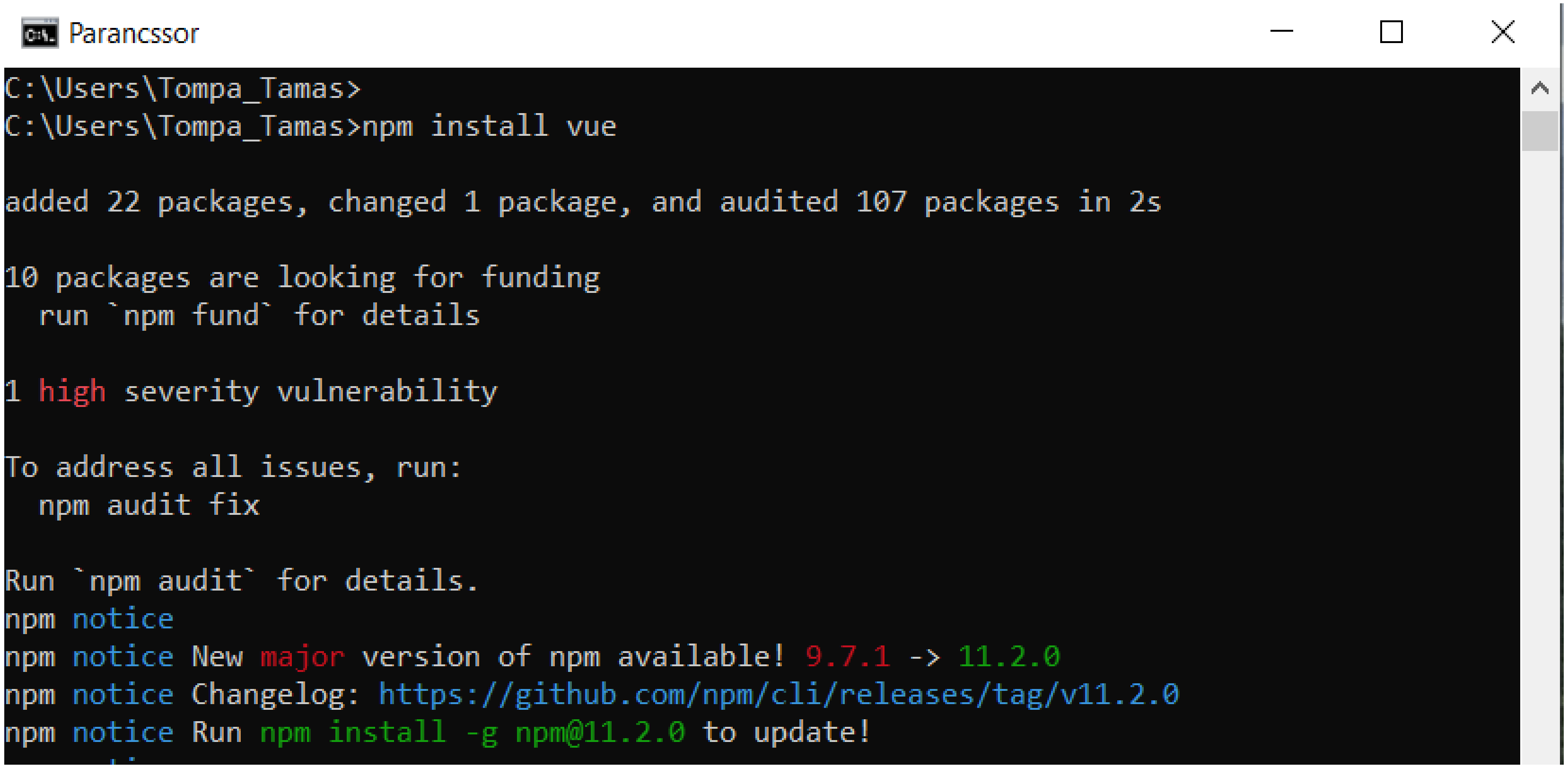
The Setup Wizard will install Node.js on your computer.



```
Microsoft Windows [Version 10.0.17763.3770]
(c) 2018 Microsoft Corporation. Minden jog fenntartva.

C:\Windows\System32>npm
npm <command>

Usage:

npm install        install all the dependencies in your project
npm install <foo>  add the <foo> dependency to your project
npm test           run this project's tests
npm run <foo>      run the script named <foo>
npm <command> -h   quick help on <command>
npm -l             display usage info for all commands
npm help <term>    search for help on <term> (in a browser)
npm help npm       more involved overview (in a browser)

All commands:
```

Install Vue.js - NPM

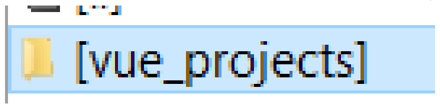
- There are many ways to install VueJS
  - **Using NPM:** `npm install vue`



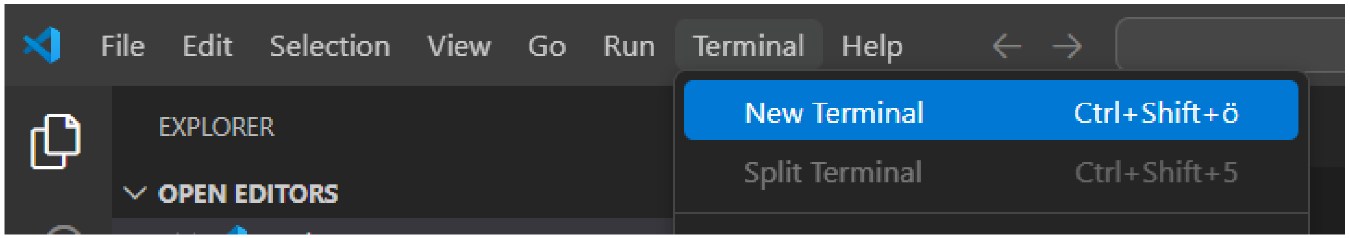
```
C:\Users\Tompa_Tamas>
C:\Users\Tompa_Tamas>npm install vue

added 22 packages, changed 1 package, and audited 107 packages in 2s

10 packages are looking for funding
  run `npm fund` for details

1 high severity vulnerability

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 9.7.1 -> 11.2.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.2.0
npm notice Run npm install -g npm@11.2.0 to update!
```

Create default example

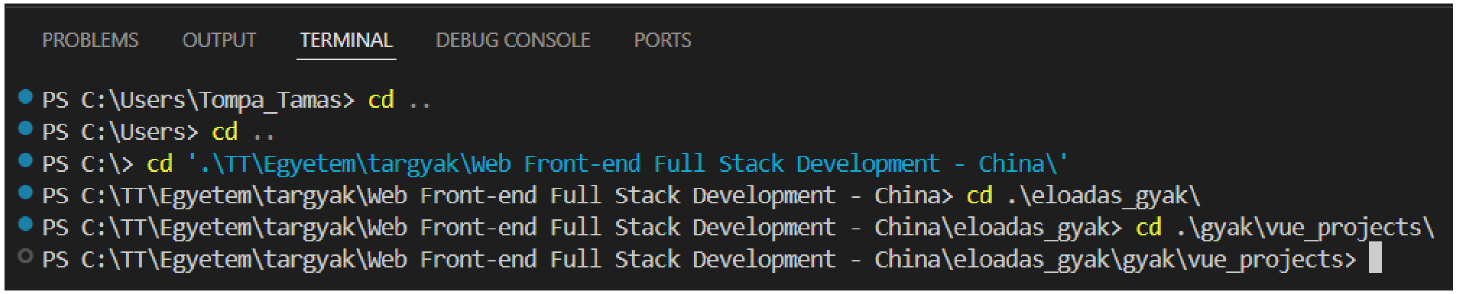
- Create a folder for your Vue projects on your computer:



- In VS Code, open a terminal by choosing Terminal → New Terminal from the menu:



- Use the terminal to navigate to the Vue folder (`cd` command):

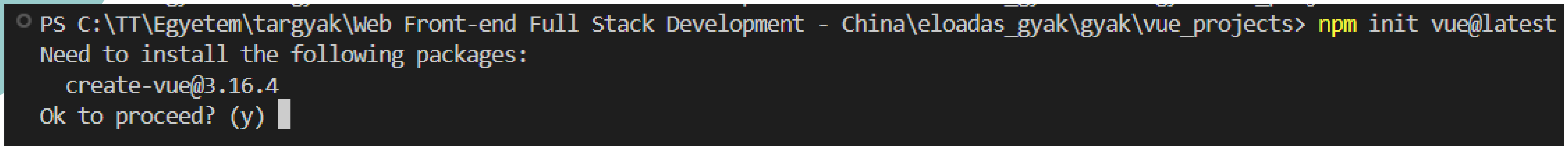


```
PS C:\Users\Tompa_Tamas> cd ..
PS C:\Users> cd ..
PS C:\> cd '.\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\'
PS C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China> cd .\eloadas_gyak\
PS C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak> cd .\gyak\vue_projects\
PS C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak\gyak\vue_projects>
```

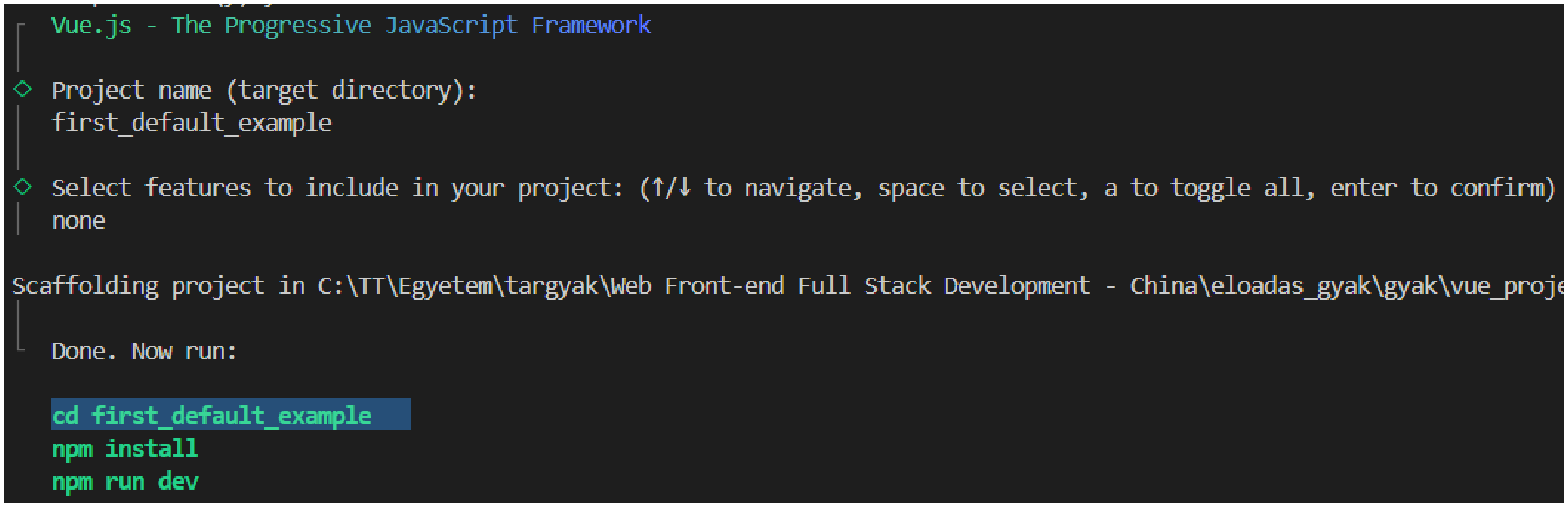
Create default example

- After you have navigated to your Vue folder in the terminal run the command:

```
npm init vue@latest
```

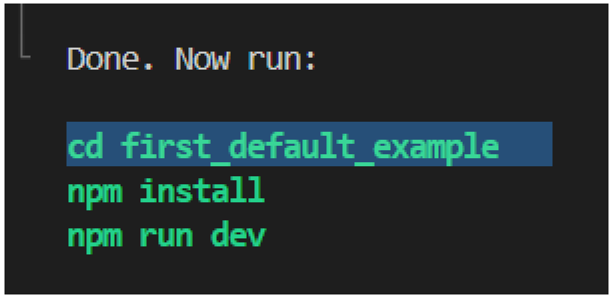


```
PS C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak\gyak\vue_projects> npm init vue@latest
Need to install the following packages:
  create-vue@3.16.4
Ok to proceed? (y)
```

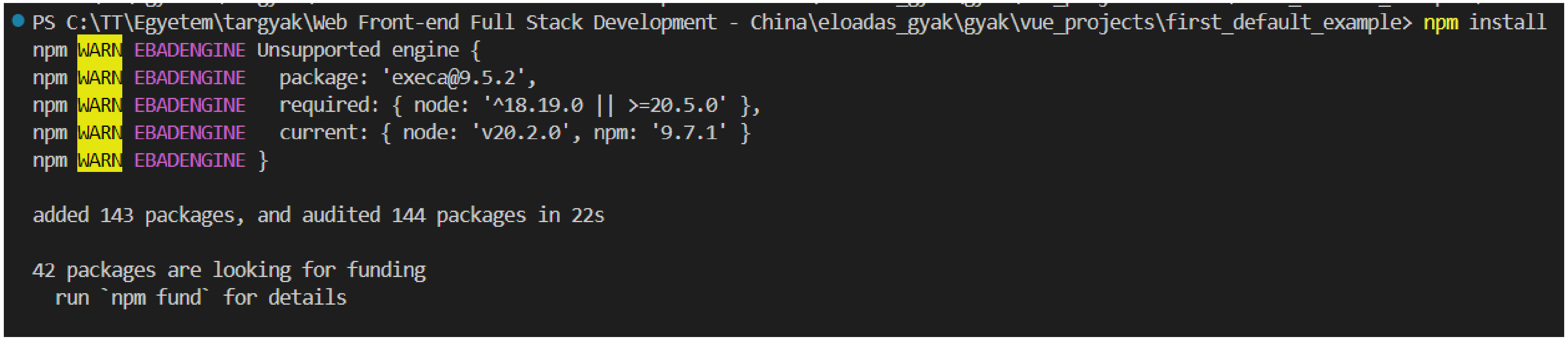


```
┌  Vue.js - The Progressive JavaScript Framework
│
◇  Project name (target directory):
│  first_default_example
│
◇  Select features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm)
│  none

Scaffolding project in C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak\gyak\vue_proje
│
└  Done. Now run:

   cd first_default_example
   npm install
   npm run dev
```

Create default example



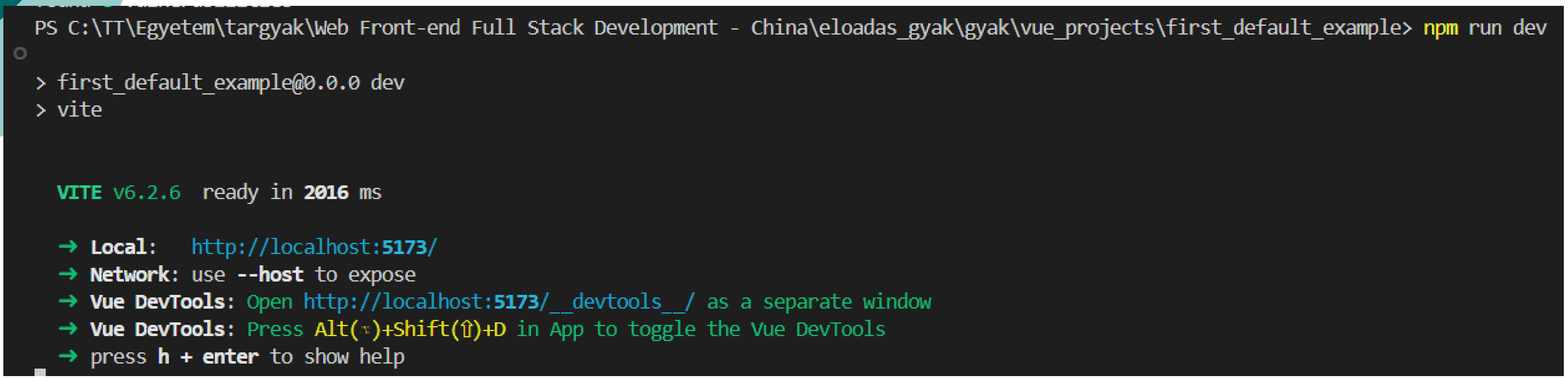
```
Done. Now run:

cd first_default_example
npm install
npm run dev
```



```
PS C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak\gyak\vue_projects\first_default_example> npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'execa@9.5.2',
npm WARN EBADENGINE   required: { node: '^18.19.0 || >=20.5.0' },
npm WARN EBADENGINE   current: { node: 'v20.2.0', npm: '9.7.1' }
npm WARN EBADENGINE }

added 143 packages, and audited 144 packages in 22s

42 packages are looking for funding
  run `npm fund` for details
```

Create default example

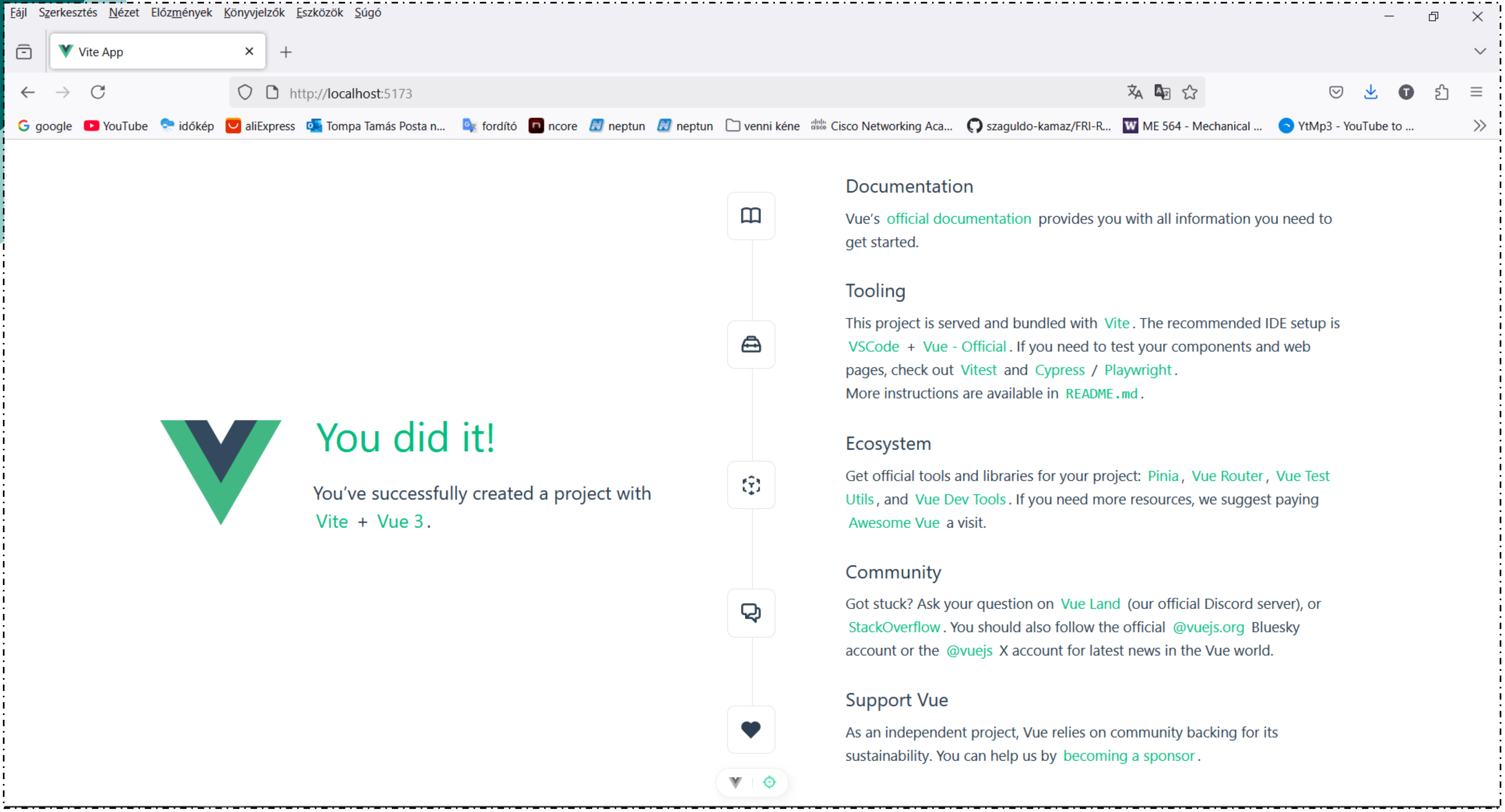
```
npm run dev
```



```
PS C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak\gyak\vue_projects\first_default_example> npm run dev

> first_default_example@0.0.0 dev
> vite


  VITE v6.2.6  ready in 2016 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  Vue DevTools: Open http://localhost:5173/__devtools__/ as a separate window
  ➜  Vue DevTools: Press Alt(⌥)+Shift(⇧)+D in App to toggle the Vue DevTools
  ➜  press h + enter to show help
```

Create default example

```
http://localhost:5173/
```


Fájl Szerkesztés Nézet Előzmények Könyvjelzők Eszközök Súgó
Vite App
http://localhost:5173
google YouTube időkép aliExpress Tompa Tamás Posta n... fordító ncore neptun neptun venni kéne Cisco Networking Aca... szaguldo-kamaz/FRI-R... ME 564 - Mechanical ... YtMp3 - YouTube to ...

You did it!

You've successfully created a project with Vite + Vue 3.

Documentation

Vue's official documentation provides you with all information you need to get started.

Tooling

This project is served and bundled with Vite. The recommended IDE setup is VSCode + Vue - Official. If you need to test your components and web pages, check out Vitest and Cypress / Playwright.
More instructions are available in `README.md`.

Ecosystem

Get official tools and libraries for your project: Pinia, Vue Router, Vue Test Utils, and Vue Dev Tools. If you need more resources, we suggest paying Awesome Vue a visit.

Community

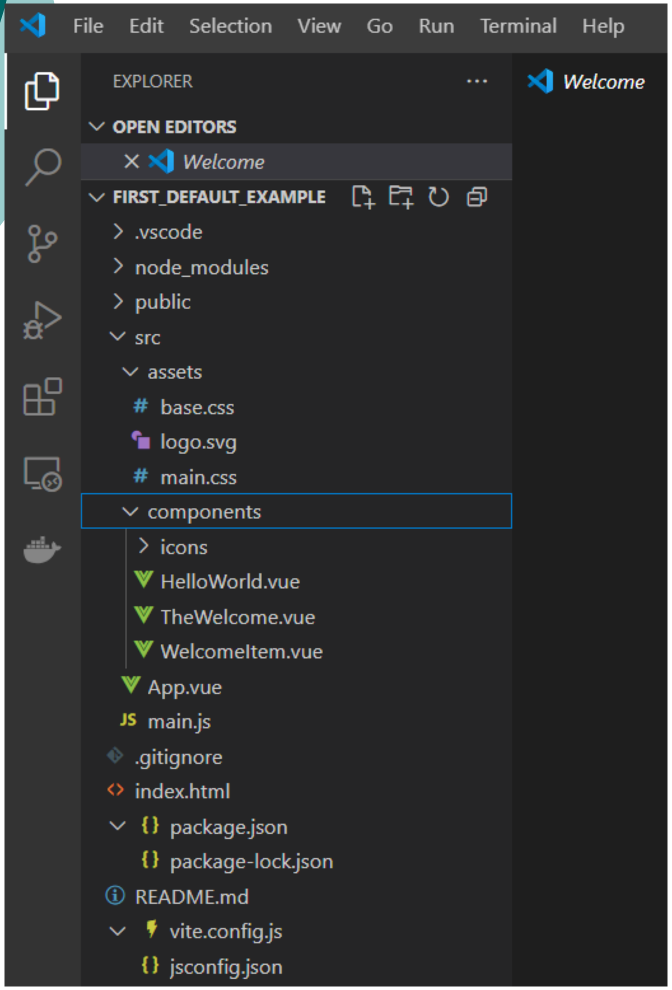
Got stuck? Ask your question on Vue Land (our official Discord server), or StackOverflow. You should also follow the official @vuejs.org Bluesky account or the @vuejs X account for latest news in the Vue world.

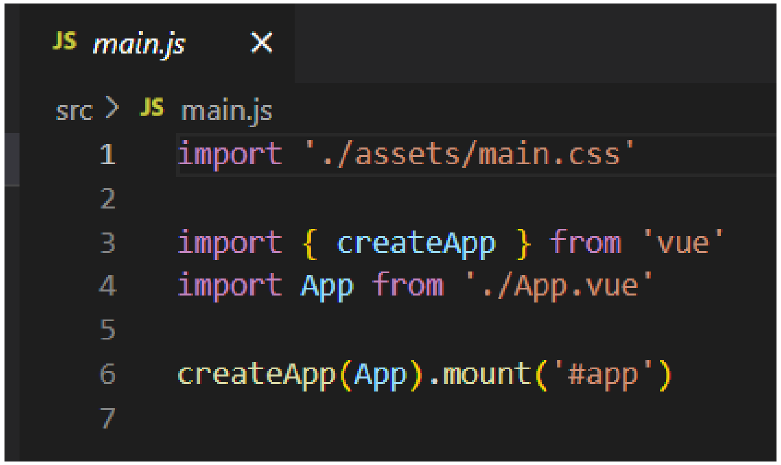
Support Vue

As an independent project, Vue relies on community backing for its sustainability. You can help us by becoming a sponsor.

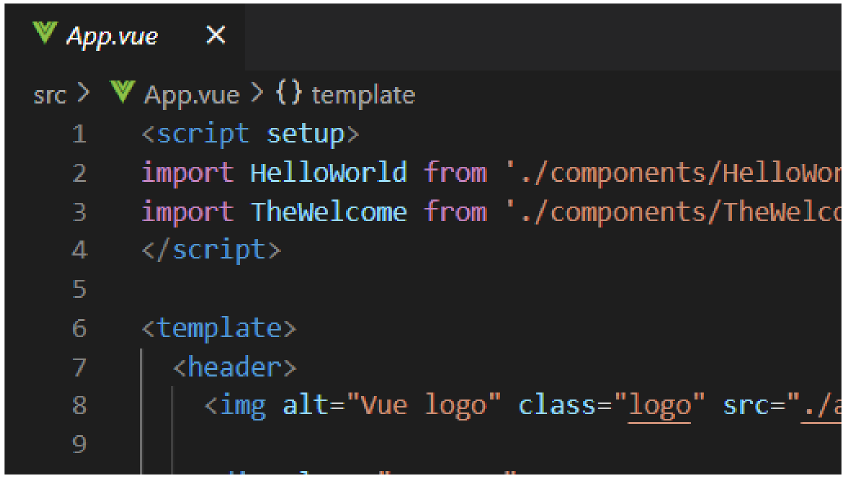
Create default example

- Open the created project by the VSCode:





```js
import './assets/main.css'

import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```



```vue
<script setup>
import HelloWorld from './components/HelloWor
import TheWelcome from './components/TheWelco
</script>

<template>
  <header>
    <img alt="Vue logo" class="logo" src="./a

```

First SFC Web Page

- **To create the first SFC web page from scratch follow the steps:**

  1. Create a new clean Vue project

  2. Write code in the 'App.vue' file

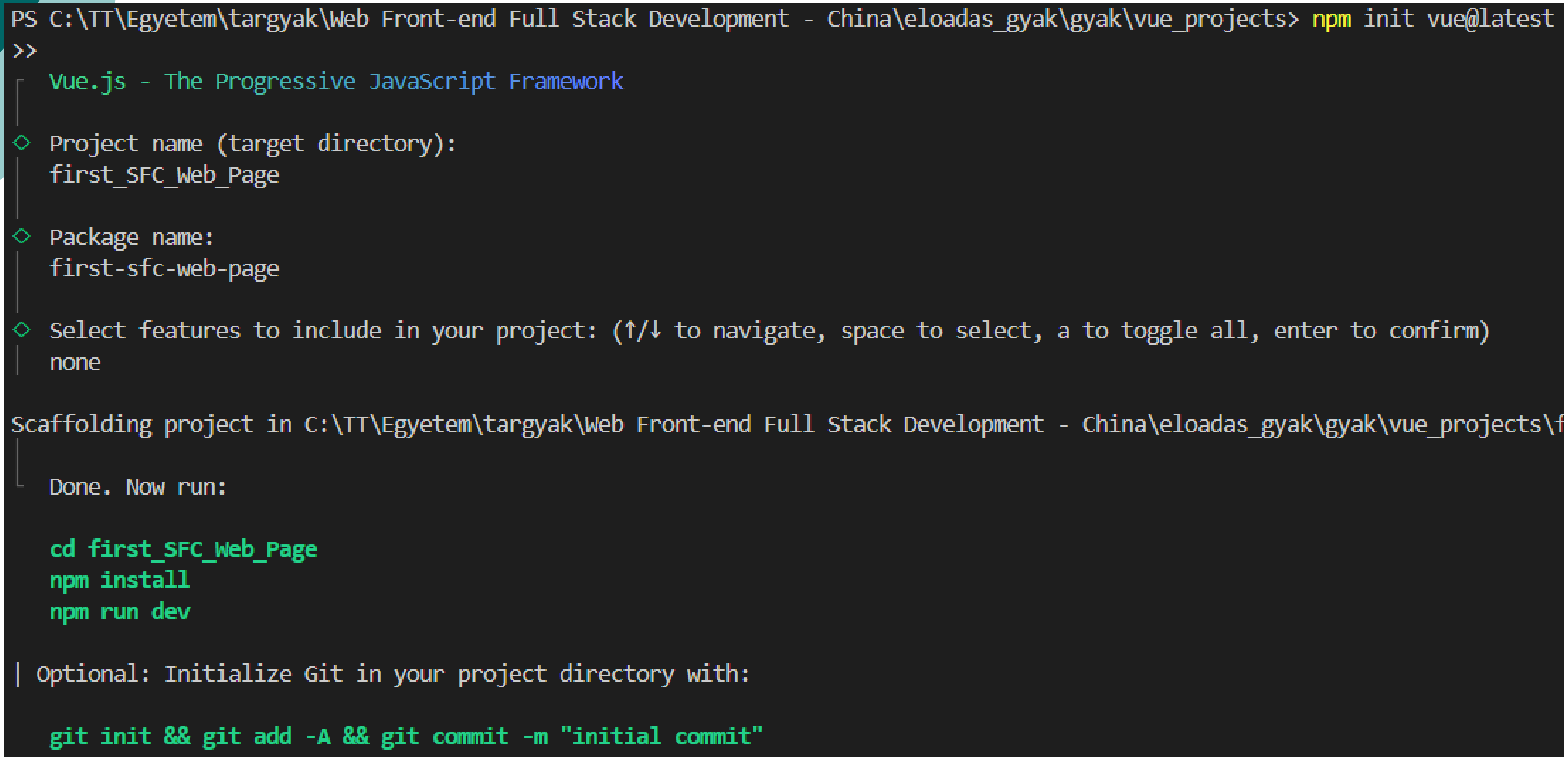
  3. See how the web page updates automatically during development

  4. Build the page for production

First SFC Web Page

1. Create a new clean Vue project

```
npm init vue@latest
```



```
PS C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak\gyak\vue_projects> npm init vue@latest
>>
┌  Vue.js - The Progressive JavaScript Framework
│
◇  Project name (target directory):
│  first_SFC_Web_Page
│
◇  Package name:
│  first-sfc-web-page
│
◇  Select features to include in your project: (↑/↓ to navigate, space to select, a to toggle all, enter to confirm)
│  none

Scaffolding project in C:\TT\Egyetem\targyak\Web Front-end Full Stack Development - China\eloadas_gyak\gyak\vue_projects\f
│
└  Done. Now run:

   cd first_SFC_Web_Page
   npm install
   npm run dev

| Optional: Initialize Git in your project directory with:

   git init && git add -A && git commit -m "initial commit"
```

First SFC Web Page

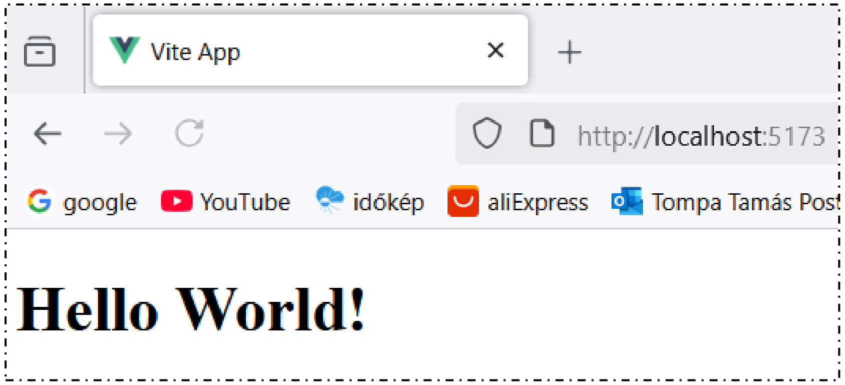
2. Write code in the „App.vue” and „main.js” files

```
<template>
  <h1>Hello World!</h1>
</template>

<script></script>
<style></style>
```

```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

3. See how the web page updates automatically during development

4. Build the page for production

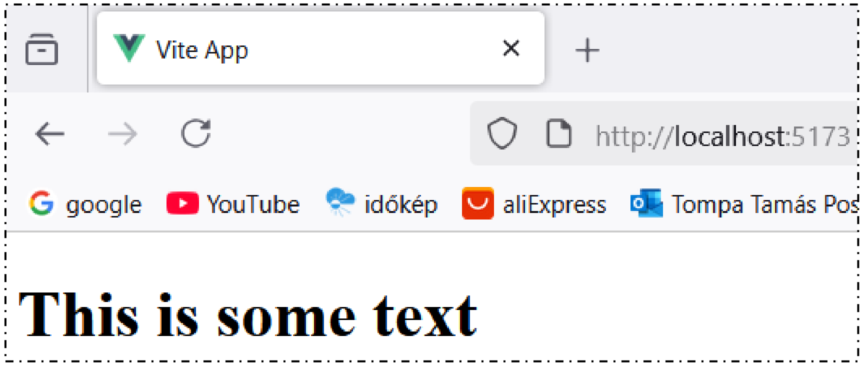


Hello World!

First SFC Web Page2

App.vue

```
<template>
   <h1>{{ message }}</h1>
</template>

<script>
export default {
   data() {
      return {
         message: 'This is some text'
      };
   }
};
</script>

<style></style>
```



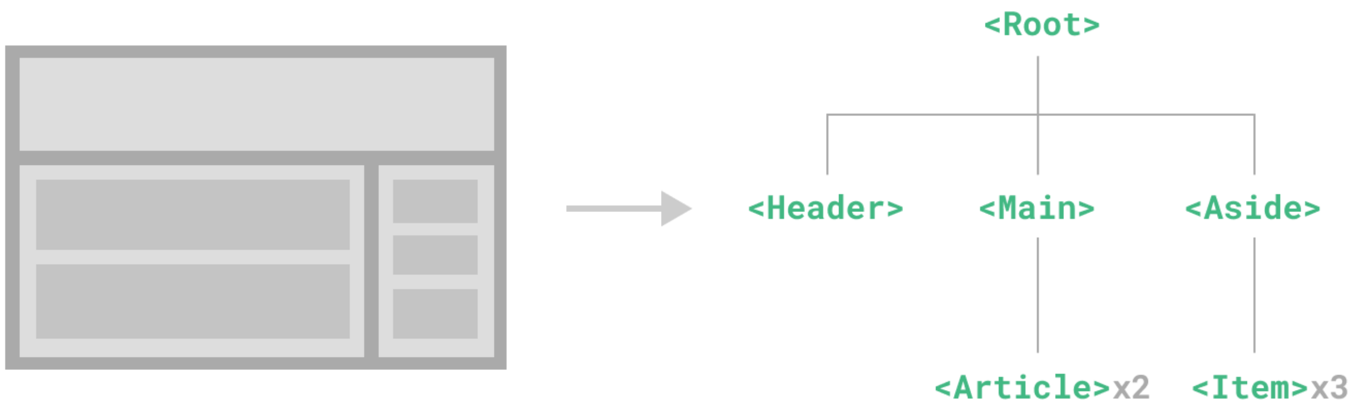
This is some text

main.js

```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

Components

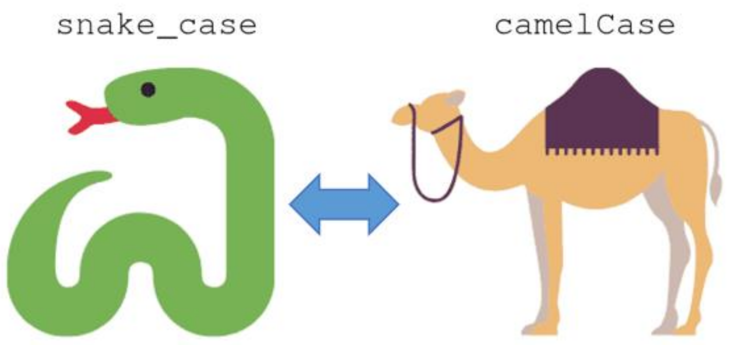
- **Components lets us decompose our web page into smaller pieces** that are easy to work with
- Components are **reusable and self-contained pieces of code** that encapsulates a specific part of the user interface
- A Vue component in isolation from the rest of the web page, with its **own content and logic**
- **Independent and reusable pieces**



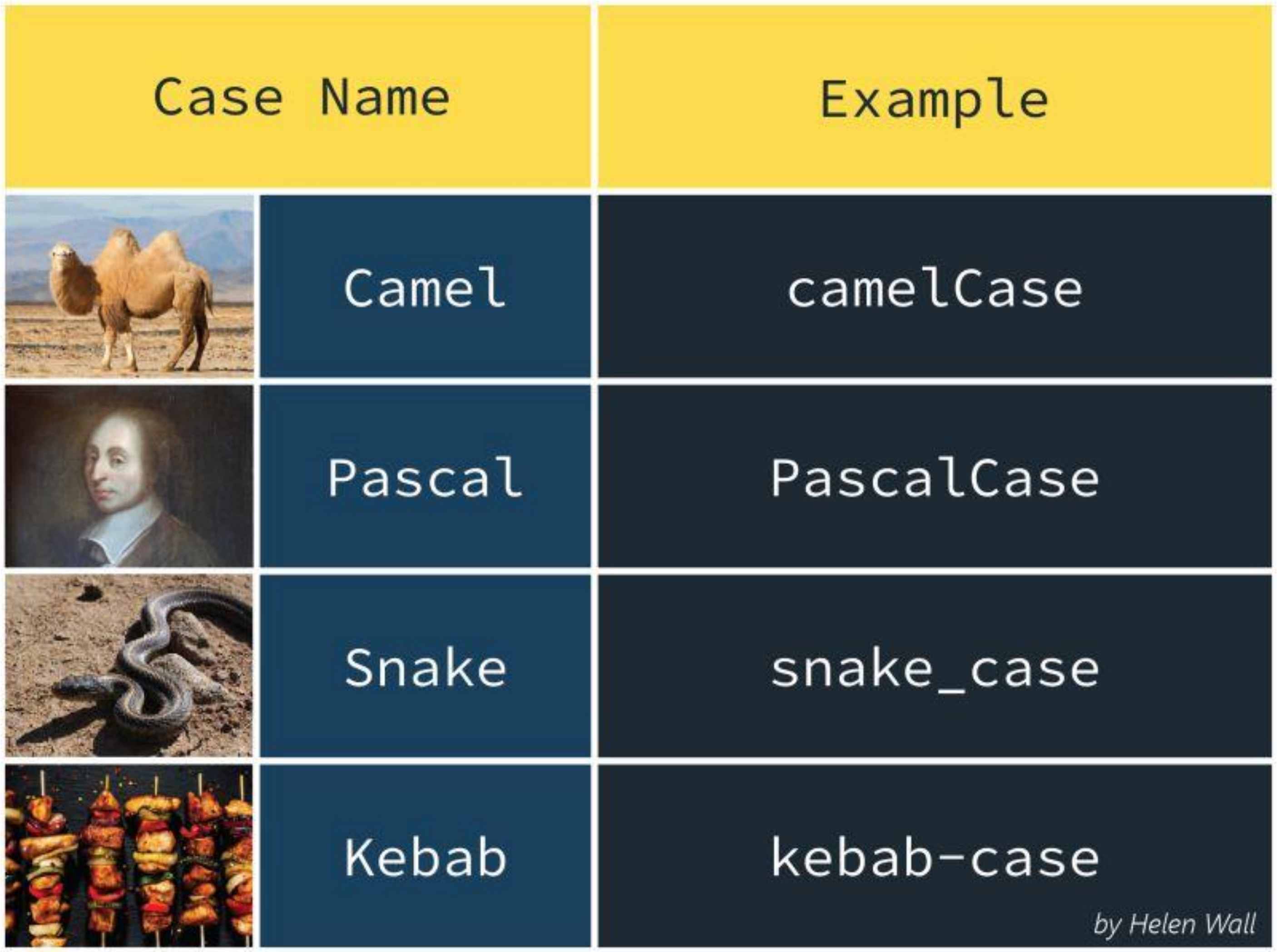
Components - Example

- Create a new folder `components` inside the `src` folder
- Inside the components folder, create a new file `FoodItem.vue`
  - it is common to name components with **CamelCase naming convention**, without spaces and where all new words starts with a capital letter, also the first word

PascalCase



Components - Example



Case Name	Example
Camel	camelCase
Pascal	PascalCase
Snake	snake_case
Kebab	kebab-case

by Helen Wall

Components - Example

FoodItem.vue

```
<template>
    <div>
      <h2>{{ name }}</h2>
      <p>{{ message }}</p>
    </div>
  </template>

  <script>
  export default {
    data() {
      return {
        name: 'Apples',
        message: 'I like apples'
      }
    }
  };
  </script>

  <style></style>
```

main.js

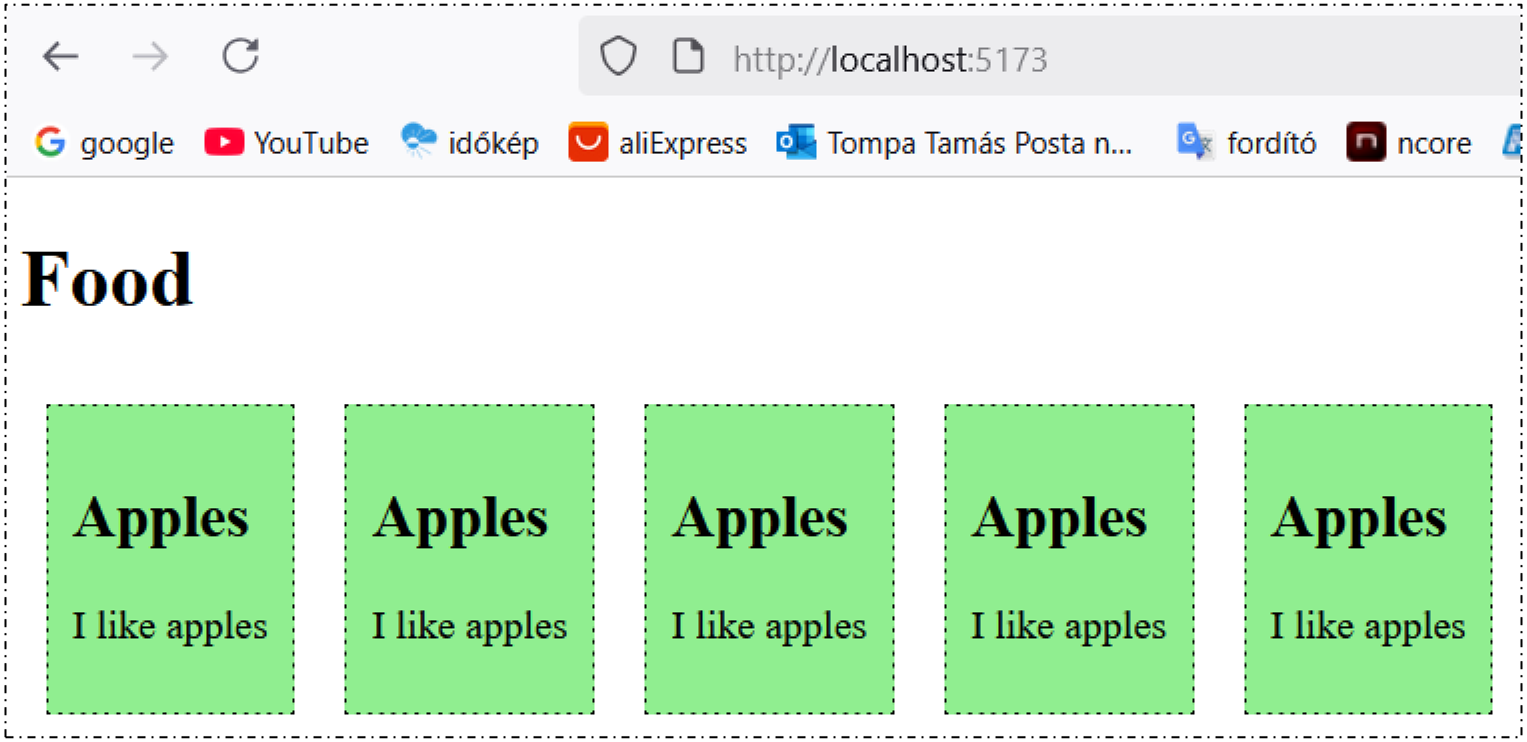
```
import { createApp } from 'vue'

import App from './App.vue'
import FoodItem from
'./components/FoodItem.vue'

const app = createApp(App)
app.component('food-item',
FoodItem)
app.mount('#app')
```

Components - Example

App.vue

```
<template>
  <h1>Food</h1>
  <food-item/>
  <food-item/>
  <food-item/>
</template>

<script></script>

<style>
  #app > div {
    border: dashed black 1px;
    display: inline-block;
    margin: 10px;
    padding: 10px;
    background-color: lightgreen;
  }
</style>
```

Components - Example



Food

Apples

I like apples

Apples

I like apples

Apples

I like apples

Apples

I like apples

Apples

I like apples

Components – Example2

- A very useful and powerful property when working with components is that can be make them behave individually, without having to mark elements with unique IDs
- Vue automatically takes care to treat each component individually
- CSS code added to the `<style>` tag in `App.vue`

```
<style>
  #app > div {
    border: dashed black 1px;
    display: inline-block;
    width: 120px;
    margin: 10px;
    padding: 10px;
    background-color: lightgreen;
  }
  #app > div:hover {
    cursor: pointer;
  }
</style>
```

Components – Example2

Vue.js

- FoodItem.vue

```
<template>
    <div v-on:click="countClicks">
        <h2>{{ name }}</h2>
        <p>{{ message }}</p>
        <p id="red">You have clicked me
{{ clicks }} times.</p>
    </div>
</template>

<script>
export default {
    data() {
        return {
            name: 'Apples',
            message: 'I like apples',
            clicks: 0
        }
    },
    methods: {
        countClicks() {
            this.clicks++;
        }
    }
};
</script>
…
```

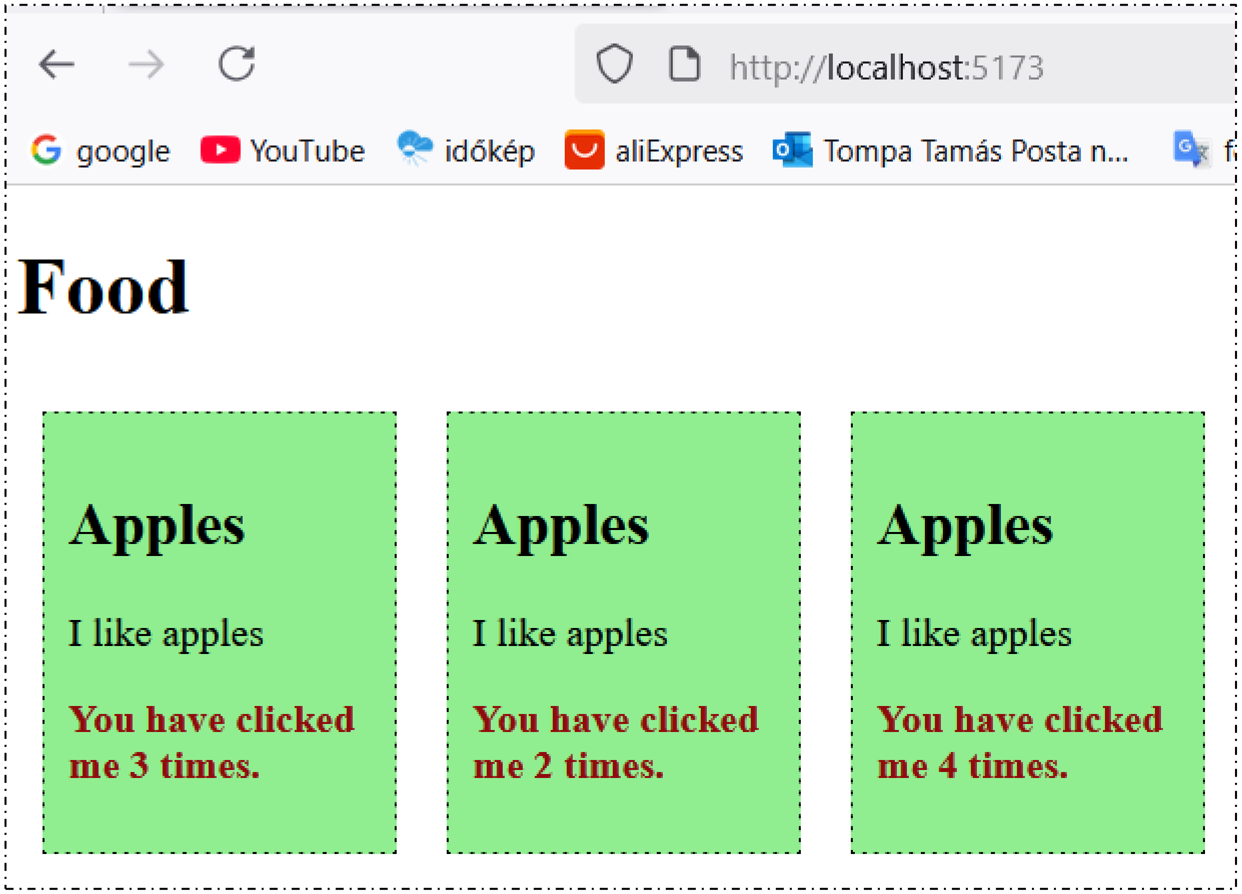
```
…
<style>
#red {
    font-weight: bold;
    color: rgb(144, 12, 12);
}
</style>
```

Components – Example2

- main.js

```
import { createApp } from 'vue'

import App from './App.vue'
import FoodItem from
'./components/FoodItem.vue'

const app = createApp(App)

app.component('food-item',
FoodItem)

app.mount('#app')
```

Components – Example2


http://localhost:5173

Food

Apples

I like apples

You have clicked me 3 times.

Apples

I like apples

You have clicked me 2 times.

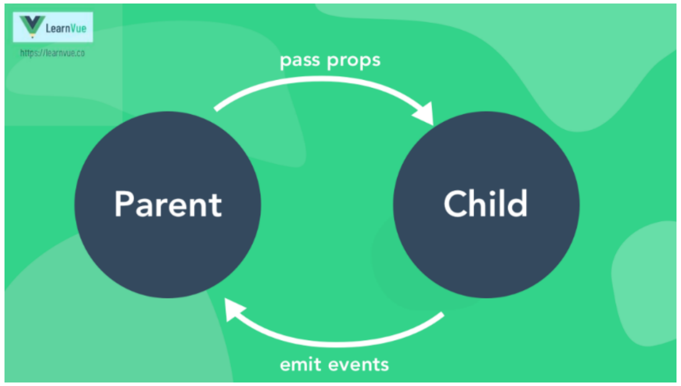
Apples

I like apples

You have clicked me 4 times.

Props

- **Props is a configuration option**
- With props **can be pass data to the components** via custom attributes to the component tag
  - pass data to a component
  - receive data inside a component
  - boolean props, object props, interface props, props validator
- Props attributes are written with a dash - to separate words (kebab-case)



Props – pass data to a component

Vue.js

App.vue

```
<template>
  <h1>Food</h1>
  <food-item food-name="Apples"/>
  <food-item food-name="Pizza"/>
  <food-item food-name="Rice"/>
</template>

<script></script>

<style>
  #app > div {
    border: dashed black 1px;
    display: inline-block;
    width: 120px;
    margin: 10px;
    padding: 10px;
    background-color: lightgreen;
  }
</style>
```

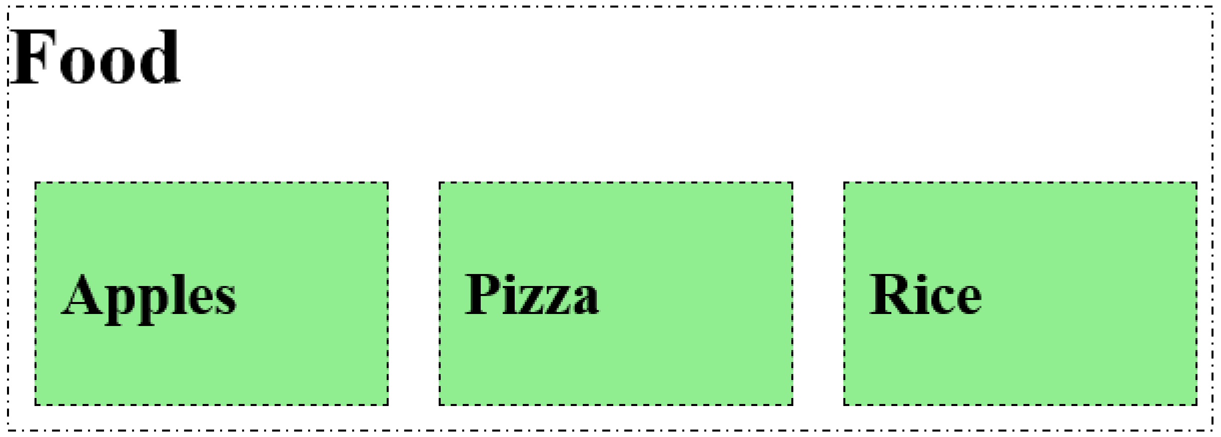
FoodItem.vue

```
<template>
    <div>
        <h2>{{ foodName }}</h2>
    </div>
</template>

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

<style></style>
```

main.js

```
import { createApp } from 'vue'

import App from './App.vue'
import FoodItem from
'./components/FoodItem.vue'

const app = createApp(App)

app.component('food-item', FoodItem)

app.mount('#app')
```

Props – pass data to a component



Food

Apples

Pizza

Rice

Props – boolean

App.vue

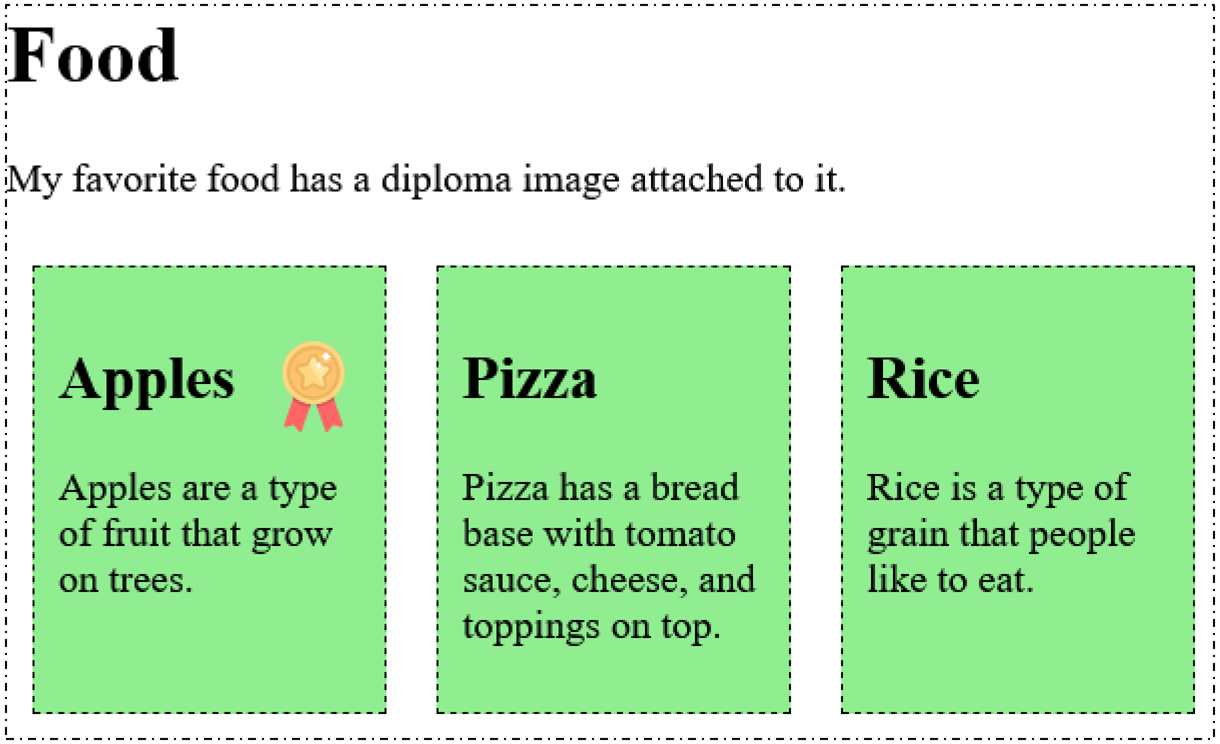
```
 <food-item
      food-name="Apples"
      food-desc="Apples are a type
of fruit that grow on trees."
      v-bind:is-favorite="true"/>
    <food-item
      food-name="Pizza"
      food-desc="Pizza has a bread
base with tomato sauce, cheese,
and toppings on top."
      v-bind:is-favorite="false"/>
    <food-item
      food-name="Rice"
      food-desc="Rice is a type of
grain that people like to eat."
      v-bind:is-favorite="false"/>
```

FoodItem.vue

```
 <h2>
    {{ foodName }}
        <img  src="/img_quality.svg"  v-
show="isFavorite">
 </h2>
 <p>{{ foodDesc }}</p>
```

```
<img src="/img_quality.svg" v-show="isFavorite">
```

Props – boolean



Food

My favorite food has a diploma image attached to it.

Apples

Apples are a type of fruit that grow on trees.

Pizza

Pizza has a bread base with tomato sauce, cheese, and toppings on top.

Rice

Rice is a type of grain that people like to eat.

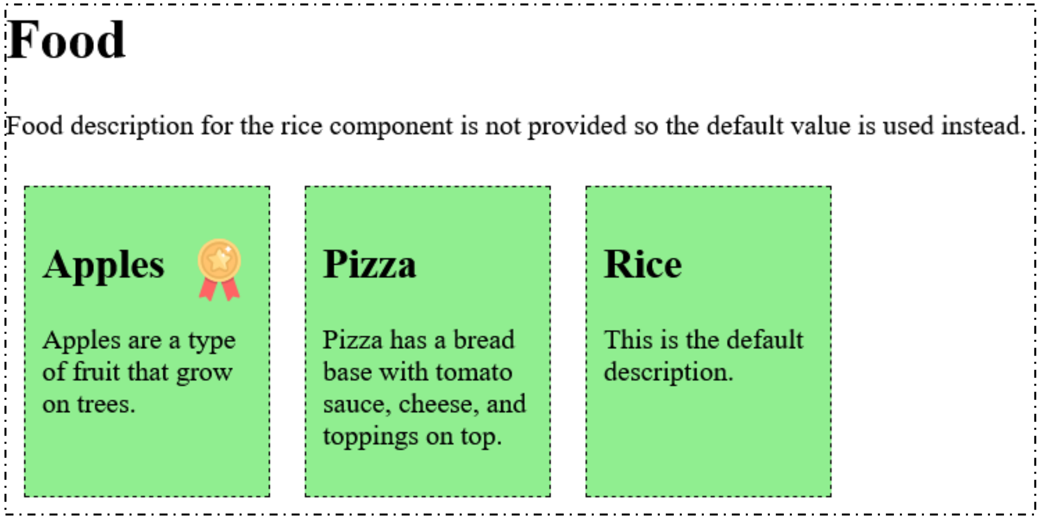
Props – object

App.vue

```
 <food-item
      food-name="Apples"
      food-desc="Apples are a type
of fruit that grow on trees."
      v-bind:is-favorite="true"/>
    <food-item
      food-name="Pizza"
      food-desc="Pizza has a bread
base with tomato sauce, cheese,
and toppings on top."
      v-bind:is-favorite="false"/>
    <food-item
      food-name="Rice"
      food-desc="Rice is a type of
grain that people like to eat."
      v-bind:is-favorite="false"/>
```

FoodItem.vue

```
<script>
  export default {
    // props:
['foodName','foodDesc','isFavorite']
    props: {
      foodName: String,
      foodDesc: String,
      isFavorite: Boolean
    }
  }
</script>
```



Food

Food description for the rice component is not provided so the default value is used instead.

Apples

Apples are a type of fruit that grow on trees.

Pizza

Pizza has a bread base with tomato sauce, cheese, and toppings on top.

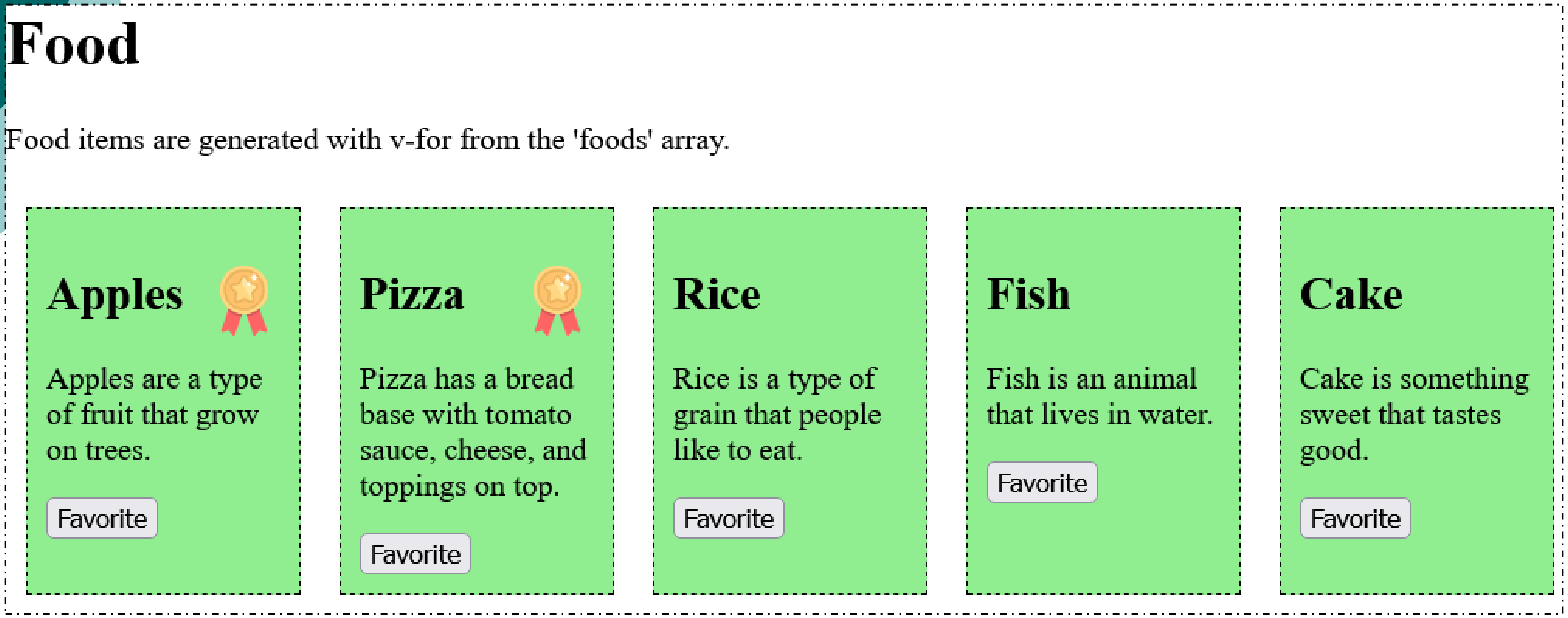
Rice

This is the default description.

V-for Components

- Components can be **reused with v-for to generate many elements of the same kind**
- When generating elements with v-for from a component, it is also very helpful that props **can be assigned dynamically based on values from an array**
- **The 'key' Attribute**
  - If we modify the array after the elements are created with v-for, errors can emerge because of the way Vue updates such elements created with v-for
  - The reason for elements being reused incorrectly is that elements do not have a unique identifier, and that is exactly what we use the key attribute for: to let Vue tell the elements apart

V-for Components



Food

Food items are generated with v-for from the 'foods' array.

Apples

Apples are a type of fruit that grow on trees.

Favorite

Pizza

Pizza has a bread base with tomato sauce, cheese, and toppings on top.

Favorite

Rice

Rice is a type of grain that people like to eat.

Favorite

Fish

Fish is an animal that lives in water.

Favorite

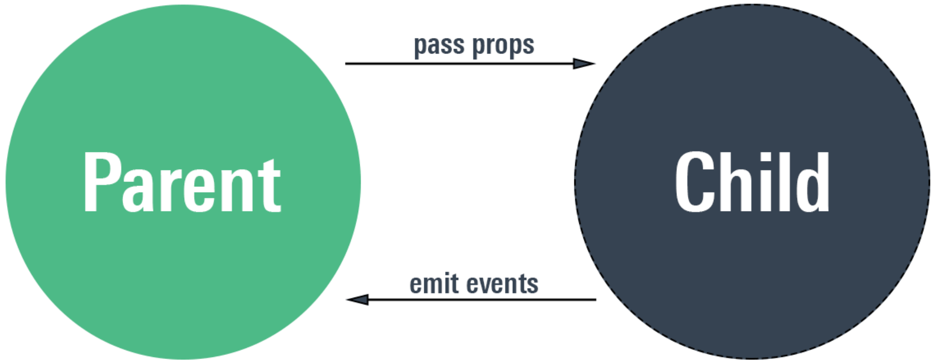
Cake

Cake is something sweet that tastes good.

Favorite

$emit() method

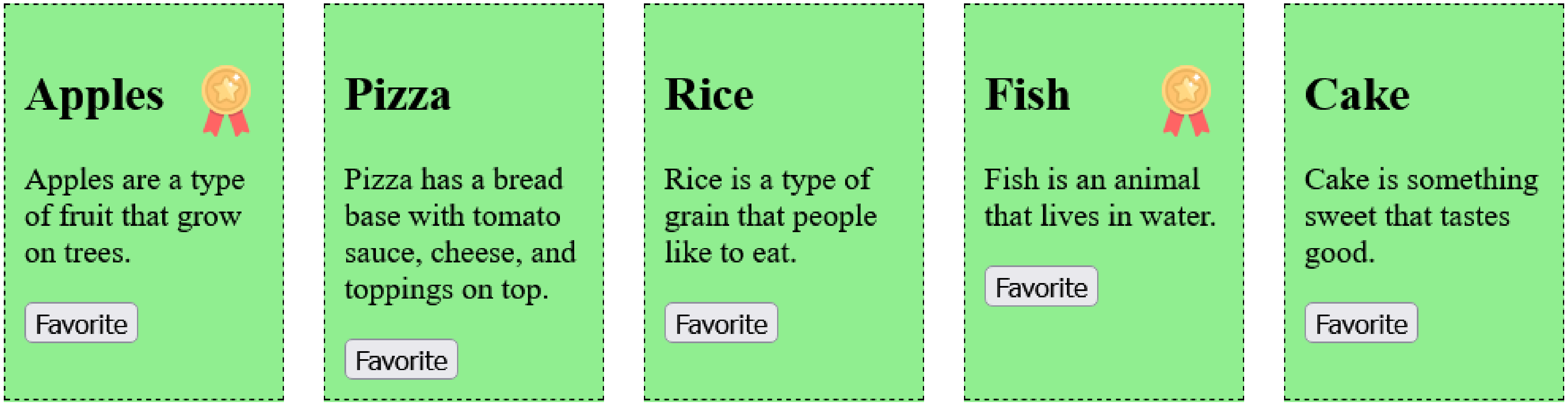
- **Can be create a custom event in the child component that can be captured in the parent element**
- **Props are used to send data from the parent element to the child component, and $emit() is used to do the opposite**
  - to pass information from the child component to the parent



$emit() method

- **In the FoodItem example**
  - the purpose of the things we will do next is to end up with the 'favorite' status of a food item to be changed in the parent App.vue instead of in the the FoodItem.vue child component where the change is currently happening
  - the reason for changing the favorite status in App.vue instead of in FoodItem.vue is that App.vue is where the favorite status is stored in the first place, so that needs to be updated
  - in a larger project the data might come from a database we have connection to in App.vue, and we want a change happening from the component to make a change in the database, so we need to communicate back to the parent from the child component

$emit() method



Apples

Apples are a type of fruit that grow on trees.

Favorite

Pizza

Pizza has a bread base with tomato sauce, cheese, and toppings on top.

Favorite

Rice

Rice is a type of grain that people like to eat.

Favorite

Fish

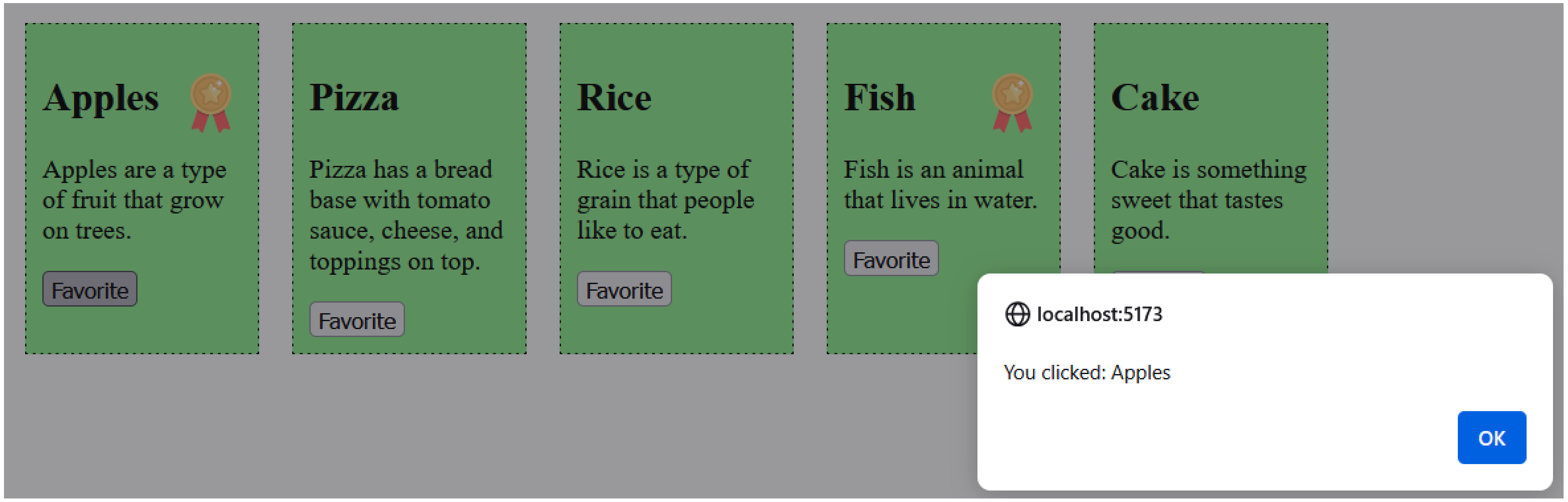
Fish is an animal that lives in water.

Favorite

Cake

Cake is something sweet that tastes good.

Favorite



localhost:5173

You clicked: Apples

OK

Fallthrough Attributes

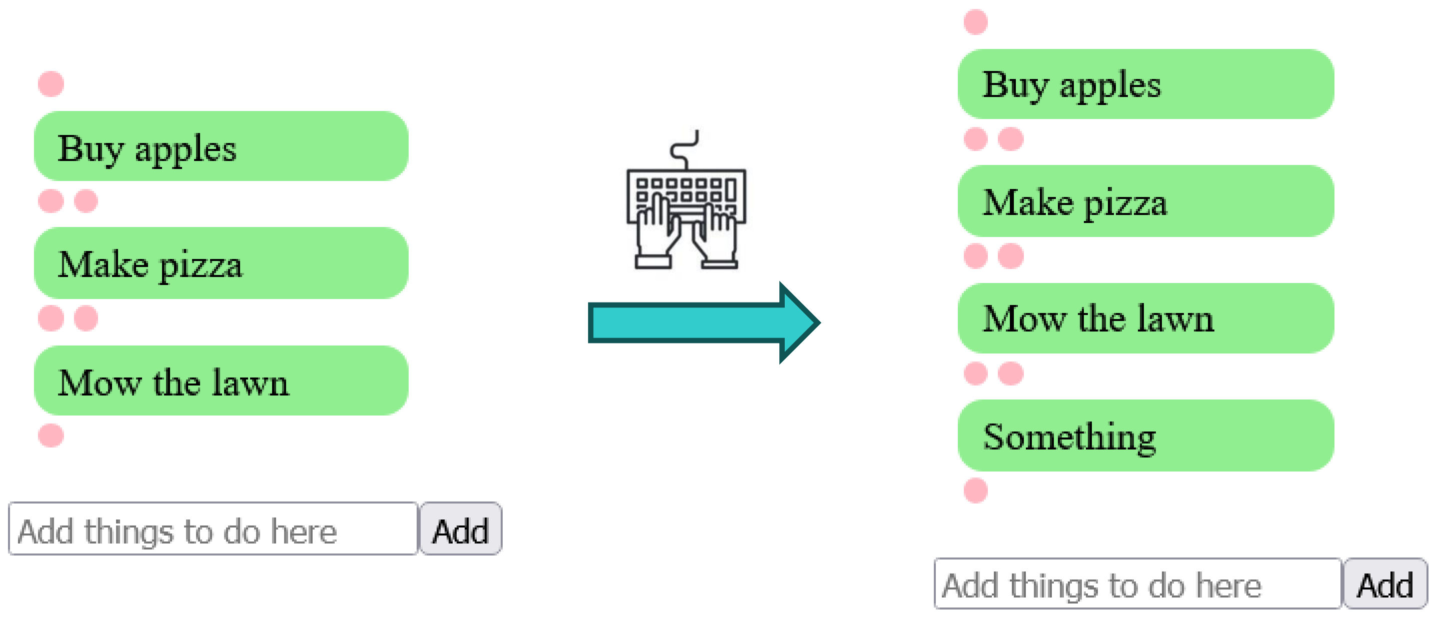
- It can be nice to for example **control the component styling from the parent rather than having the styling hidden away inside the component**

- Let's create a new example, a basic to-do list in Vue, and see how the style attribute falls through to the components representing things to do

- So, our `App.vue` should contain the list of things to do, and an `<input>` element and a `<button>` to add new things to do. Each list item is a `<todo-item/>` component

Fallthrough Attributes

Vue.js



Scoped Styling

- Styling defined inside the **`<style>` tag** in a component, or in `App.vue`, is actually available **globally in all components**
- To keep the styling limited **locally to just the component, can be use** the scope attribute on that component: **`<style scoped>`**
  - CSS written inside the `<style>` tag in any `*.vue` file works globally
  - To avoid that the styling in one component affects the styling of elements in other components we use the 'scoped' attribute on the `<style>` tag

Scoped Styling

CompOne.vue

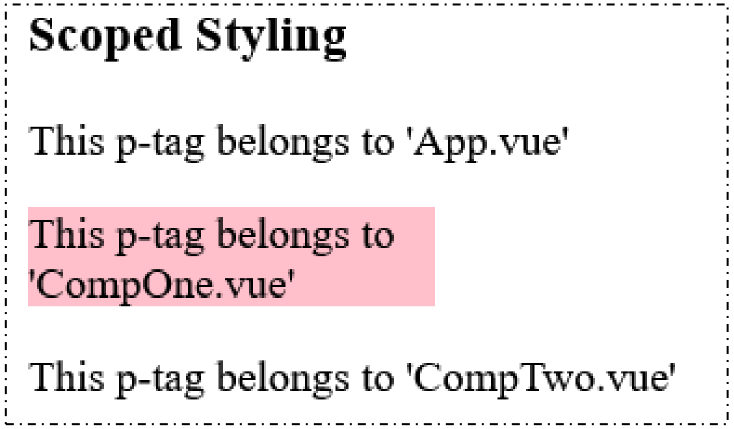
```
<template>
    <p>This p-tag belongs to 'CompOne.vue'</p>
</template>

<script></script>

<style scoped>
    p {
        background-color: pink;
        width: 150px;
    }
</style>
```

CompTwo.vue

```
<template>
    <p>This p-tag belongs to 'CompTwo.vue'</p>
</template>

<script></script>

<style></style>
```

App.vue

```
<template>
  <div>
    <h3>Scoped Styling</h3>
    <p>This p-tag belongs to 'App.vue'</p>
    <comp-one />
    <comp-two />
  </div>
</template>

<script></script>

<style></style>
```



Scoped Styling

This p-tag belongs to 'App.vue'

This p-tag belongs to 'CompOne.vue'

This p-tag belongs to 'CompTwo.vue'

Local vs. global components

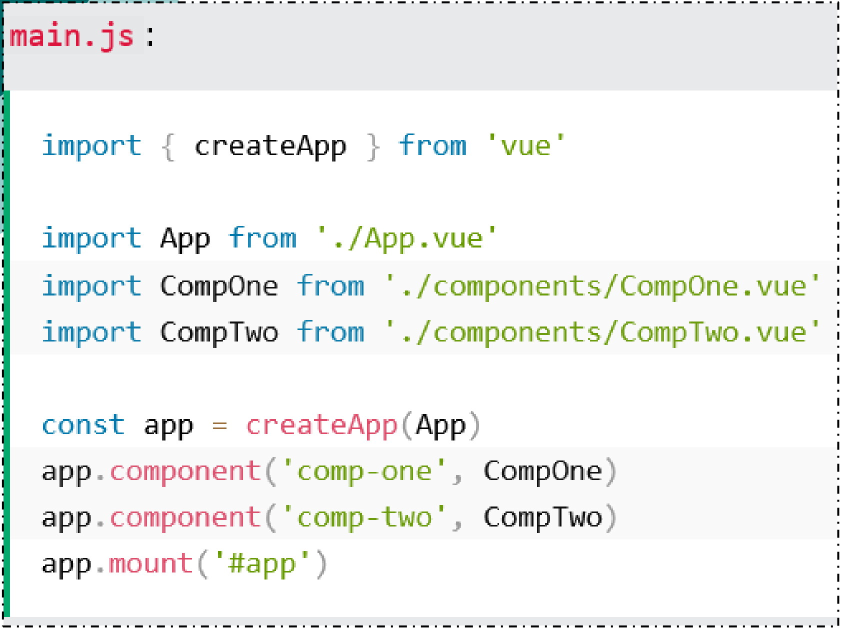
- The way we have included components so far makes **them accessible from all *.vue files in a project (global)**
  - the way we have included components inside main.js so far make the components accessible inside the `<template>` of all other `*.vue` files in that project

- Components can be made to be **local, meaning that they are only accessible inside a specific *.vue file**
  - we can include a component directly in the `<script>` tag in a `*.vue` file instead of including it in main.js
  - if we include a component directly in a `*.vue` file, the component becomes accessible only locally in that file

Local vs. global components

Vue.js

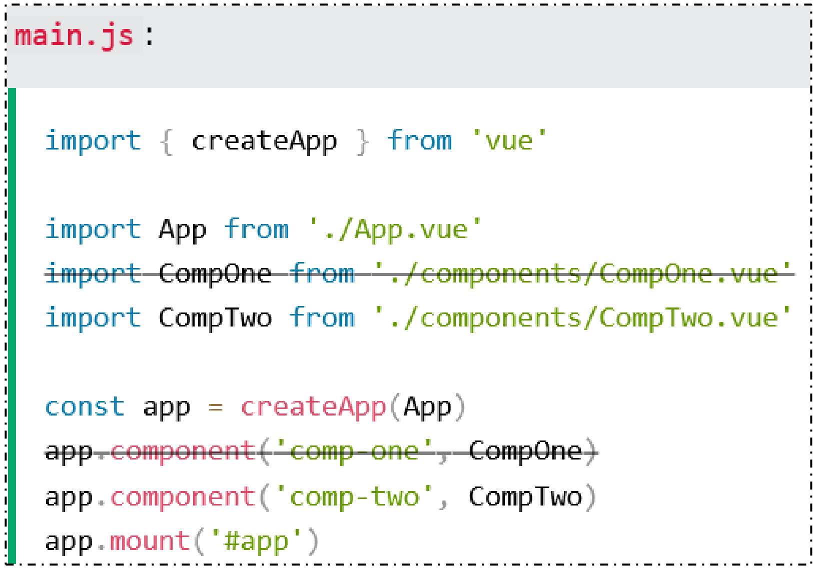
global



main.js:

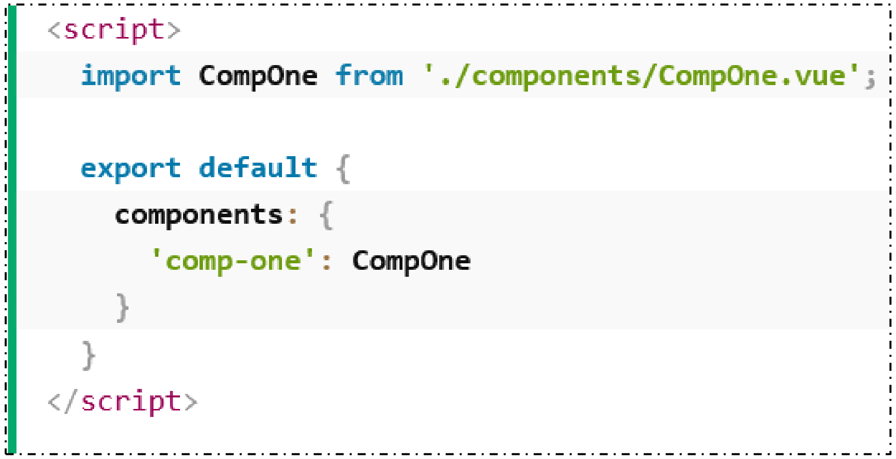
```
import { createApp } from 'vue'

import App from './App.vue'
import CompOne from './components/CompOne.vue'
import CompTwo from './components/CompTwo.vue'

const app = createApp(App)
app.component('comp-one', CompOne)
app.component('comp-two', CompTwo)
app.mount('#app')
```

local



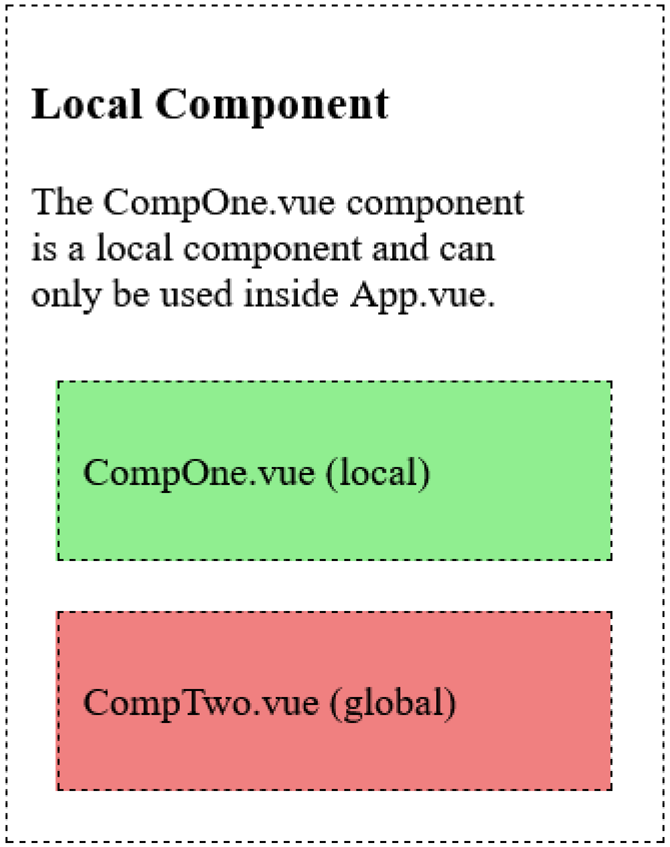
main.js:

```
import { createApp } from 'vue'

import App from './App.vue'
~~import CompOne from './components/CompOne.vue'~~
import CompTwo from './components/CompTwo.vue'

const app = createApp(App)
~~app.component('comp-one', CompOne)~~
app.component('comp-two', CompTwo)
app.mount('#app')
```



```
<script>
  import CompOne from './components/CompOne.vue';

  export default {
    components: {
      'comp-one': CompOne
    }
  }
</script>
```

Local vs. global components



Local Component

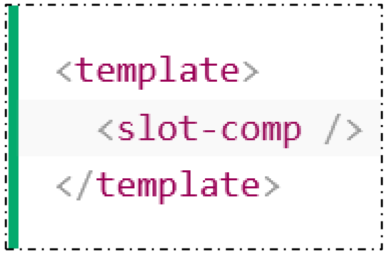
The CompOne.vue component is a local component and can only be used inside App.vue.

CompOne.vue (local)

CompTwo.vue (global)

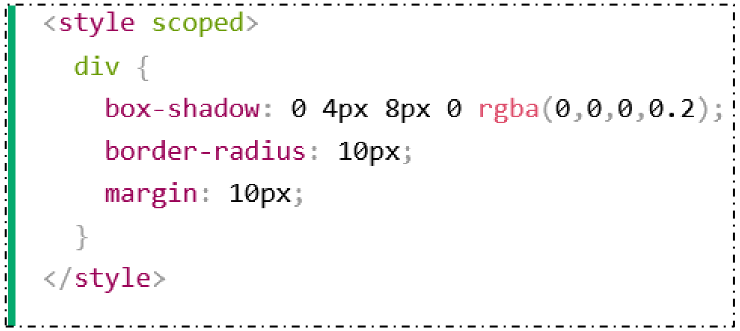
Slots

- Slots are a powerful feature in Vue that allow for more **flexible and reusable components**
- We use slots in Vue to send content from the parent into the `<template>` of a child component



```
<template>
  <slot-comp />
</template>
```

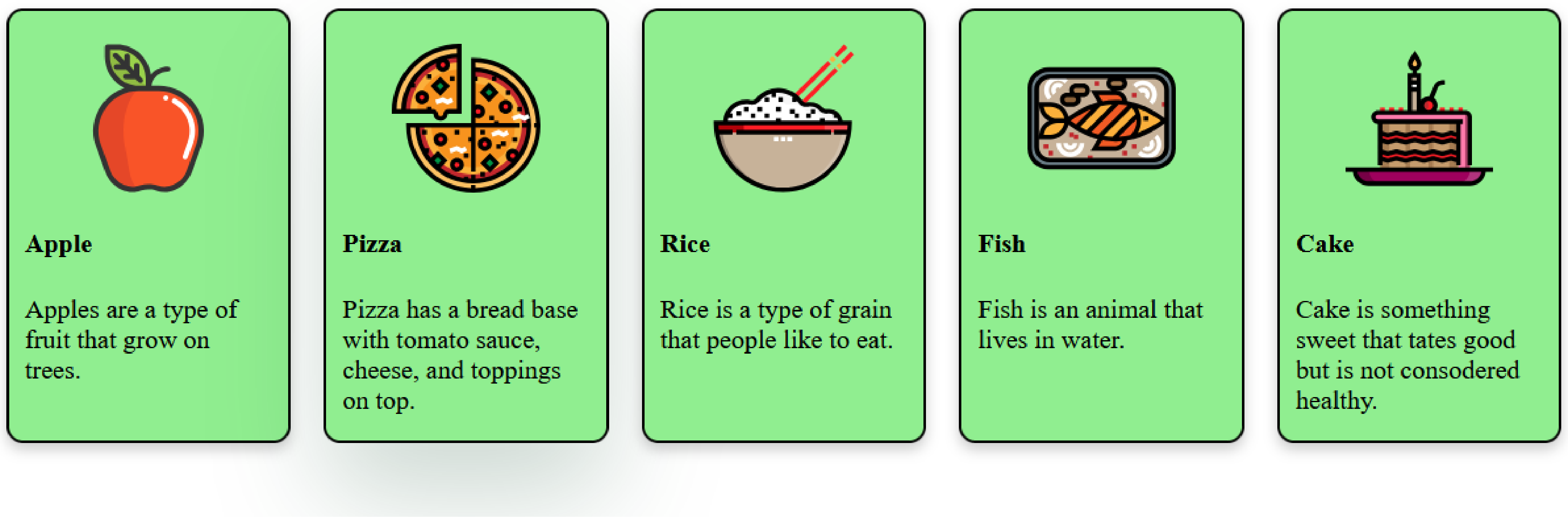
- Slots can also be **used to wrap around larger chunks of dynamic html content** to get a card-like appearance



```
<style scoped>
  div {
    box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
    border-radius: 10px;
    margin: 10px;
  }
</style>
```

Slots

Vue.js



Apple

Apples are a type of fruit that grow on trees.

Pizza

Pizza has a bread base with tomato sauce, cheese, and toppings on top.

Rice

Rice is a type of grain that people like to eat.

Fish

Fish is an animal that lives in water.

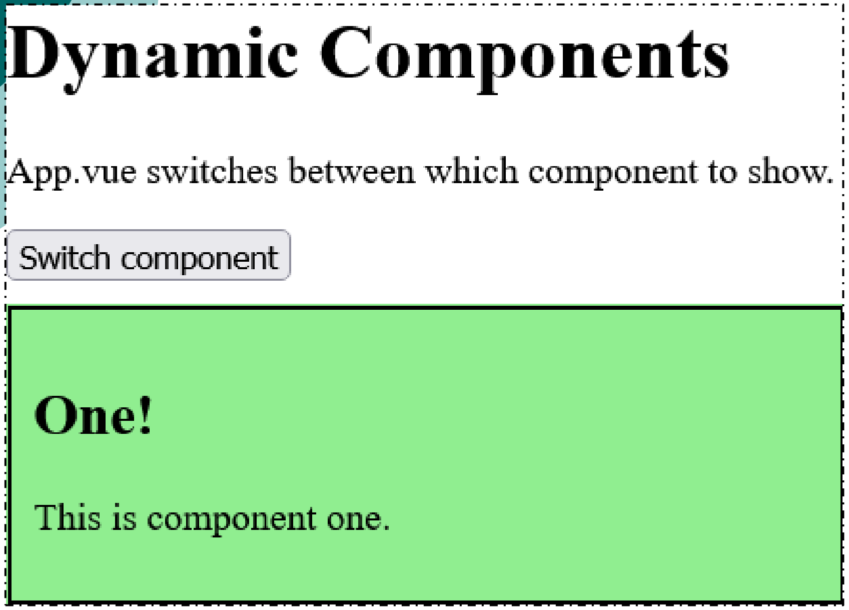
Cake

Cake is something sweet that tates good but is not consodered healthy.

Dynamic components

- **Dynamic Components can be used to flip through pages within your page**, like tabs in your browser, with the use of the 'is' attribute
- To make a dynamic component we **use the `<component>` tag to represent the active component**
- The 'is' attribute is tied to a value with v-bind, and we change that value to the name of the component we want to have active
- All components inside the `<KeepAlive>` tag will be kept alive by default
  - But we can also define only some components to be kept alive by using `include` or `exclude` attributes on the `<KeepAlive>` tag

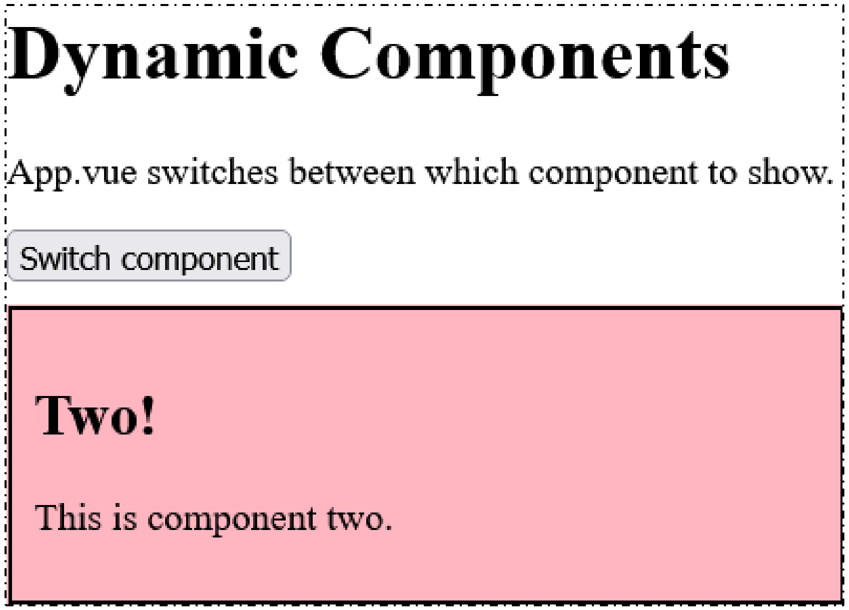
Dynamic components


Dynamic Components

App.vue switches between which component to show.

One!

This is component one.


Dynamic Components

App.vue switches between which component to show.

Two!

This is component two.

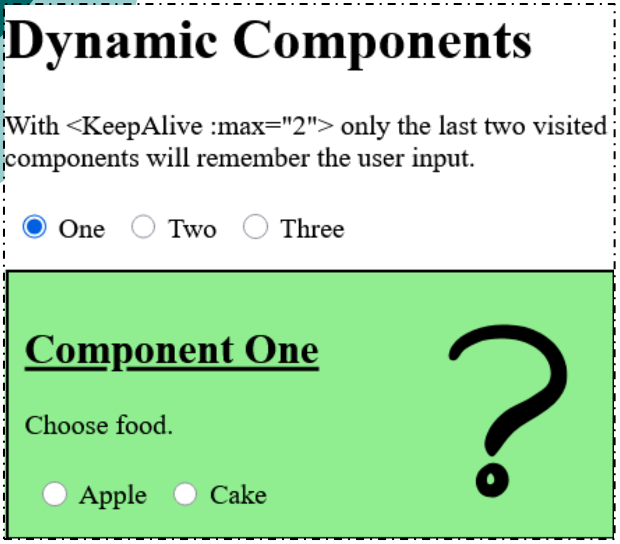
Dynamic components

- All components inside the `<KeepAlive>` tag will be kept alive by default
  - But we can also define only some components to be kept alive by using 'include' or 'exclude' attributes on the `<KeepAlive>` tag
- If we use the 'include' or '`exclude`' attributes on the `<KeepAlive>` tag we also need to give the components names with the '`name`' option

```
<script>
  export default {
    name: 'CompOne',
    data() {
      return {
        imgSrc: 'img_question.svg'
      }
    }
  }
</script>
```

```
<template>
  <h1>Dynamic Components</h1>
  <p>App.vue switches between which component to show.</p>
  <button @click="toggleValue = !toggleValue">
    Switch component
  </button>
  <KeepAlive include="CompOne">
    <component :is="activeComp"></component>
  </KeepAlive>
</template>
```

Dynamic components



Dynamic Components

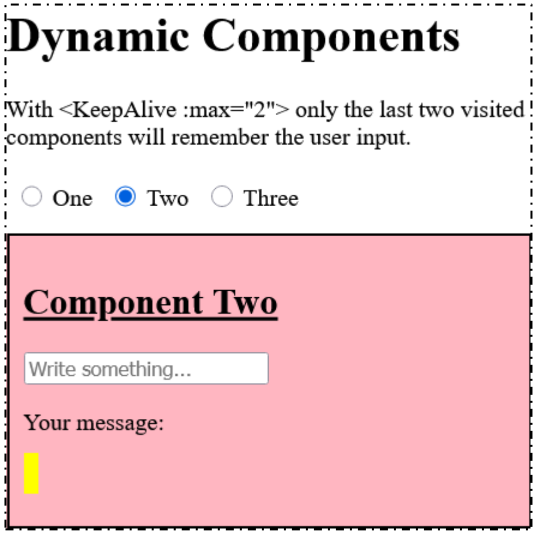
With <KeepAlive :max="2"> only the last two visited components will remember the user input.

[● One] ○ Two ○ Three

Component One

Choose food.

○ Apple ○ Cake



Dynamic Components

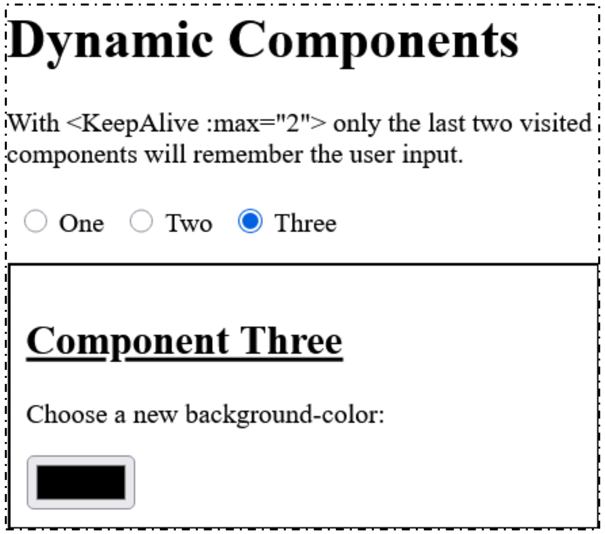
With <KeepAlive :max="2"> only the last two visited components will remember the user input.

○ One [● Two] ○ Three

Component Two

Write something...

Your message:



Dynamic Components

With <KeepAlive :max="2"> only the last two visited components will remember the user input.

○ One ○ Two [● Three]

Component Three

Choose a new background-color:

HTTP Requests

- **The HTTP request is a part of the communication between a client and a server**
- The client sends an HTTP request to the server, which handles the request and returns an HTTP response
- The most common kinds of HTTP requests are POST, GET, PUT, PATCH, and DELETE

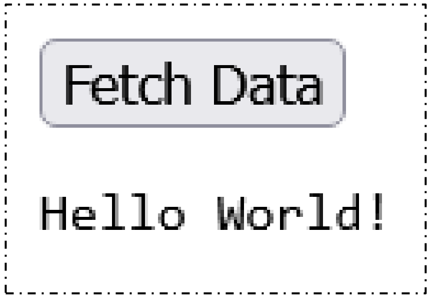
```
 methods: {
   async fetchData() {
     const response = await fetch("file.txt");
     this.data = await response.text();
   }
 }
};
```

```
 methods: {
   async fetchData() {
     const response = await fetch("bigLandMammals.json");
     this.data = await response.json();
   }
 }
};
```

HTTP Requests

- Fetch data from txt file:

```
methods: {
    async fetchData() {
       const response = await fetch("file.txt");
       this.data = await response.text();
    }
}
```



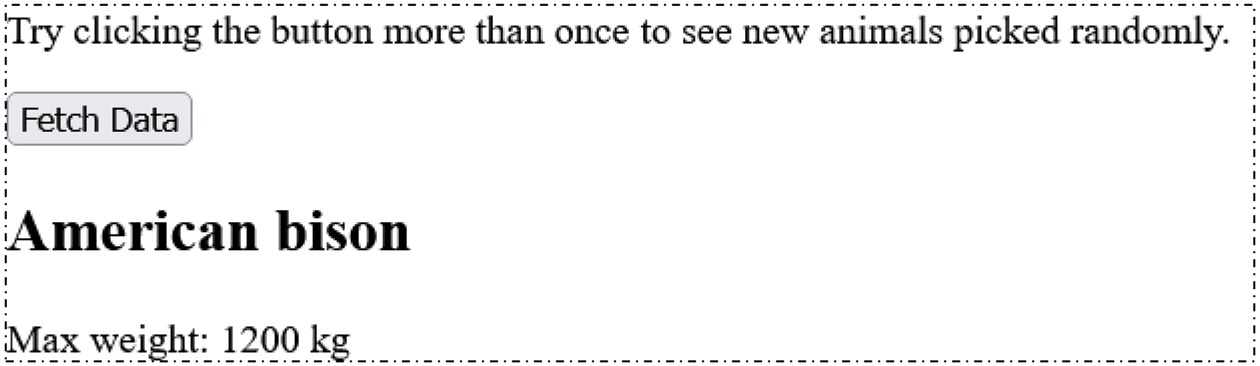
```
Hello World!
```

HTTP Requests

- Fetch data from JSON file:

```
methods: {
    async fetchData() {
      const response = await fetch("bigLandMammals.json");
      const data = await response.json();
      const randIndex = Math.floor(Math.random()*data.results.length);
      this.randomMammal = data.results[randIndex];
    }
}
```

```
{
"name": "American bison",
"maxWeight": 1200,
"carnivore": false,
"countries": [
  "USA",
  "Canada"
  ]
}
```


Try clicking the button more than once to see new animals picked randomly.

American bison

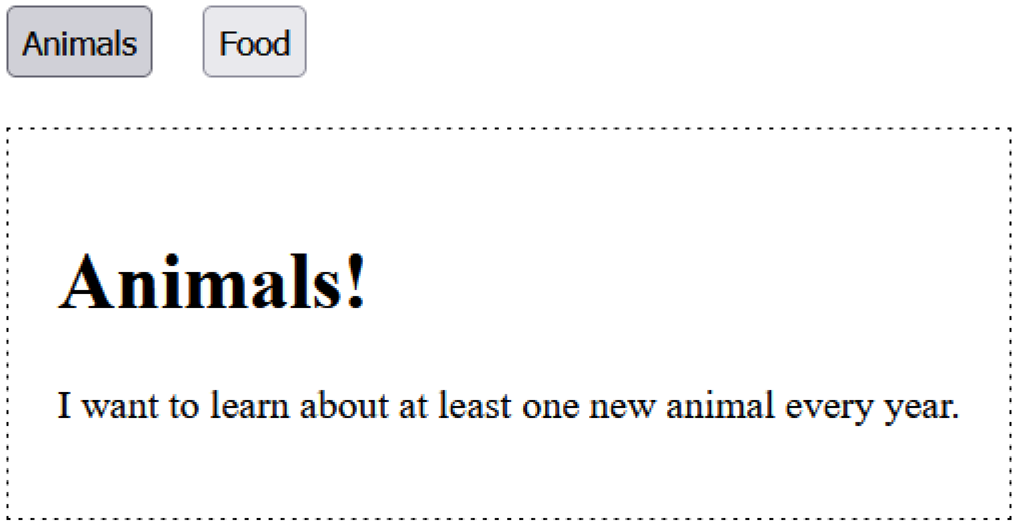
Max weight: 1200 kg

Routing

- **Routing in Vue is used to navigate the Vue application**
  - it happens on the client side (in the browser) without full page reload, which results in a faster user experience
- Routing is a **way to navigate**
- **With routing can be use the URL address to direct someone to a specific place in our Vue application**

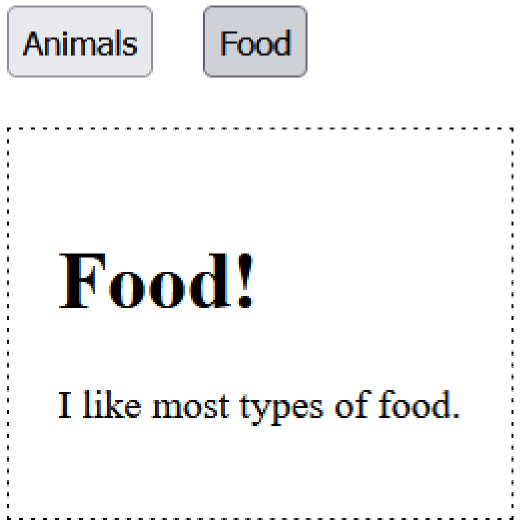
```
<button @click="activeComp = 'animal-collection'">Animals</button>
<button @click="activeComp = 'food-items'">Food</button><br>
```

Routing



Animals!

I want to learn about at least one new animal every year.



Food!

I like most types of food.

Forms

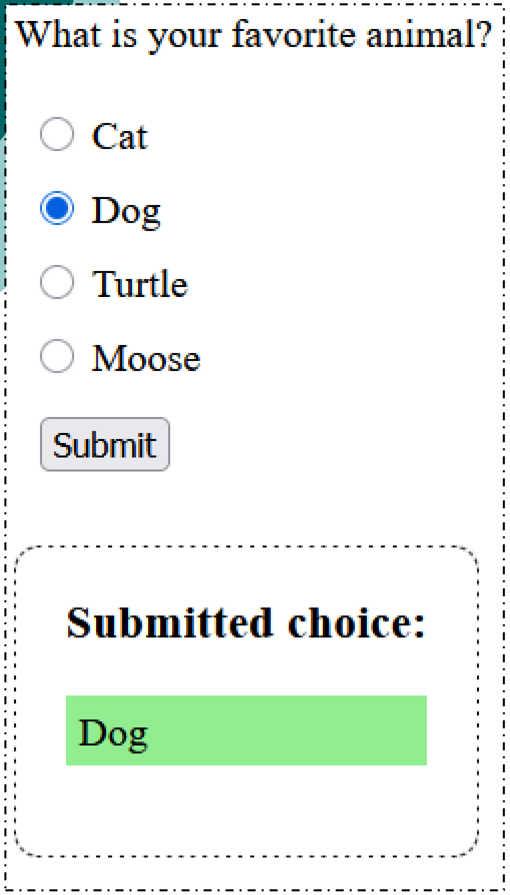
- **Radio buttons** that belong to the same choice must have the same name so that only one radio button can be chosen, `<input type="radio">` tag
- When **checkbox** inputs (`<input type="checkbox">`) are connected to the same array with v-model, the values for the checked checkboxes are gathered in that array
- A **drop-down list** consists of a `<select>` tag with `<option>` tags inside
- With the **multiple attribute** in the `<select>` tag, the drop-down list becomes expanded, and can be choose more than one option

Forms

- **Different form inputs:**

```
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="number">
<input type="password">
<input type="range">
<input type="search">
<input type="tel">
<input type="text">
<input type="time">
<textarea>
```

Forms



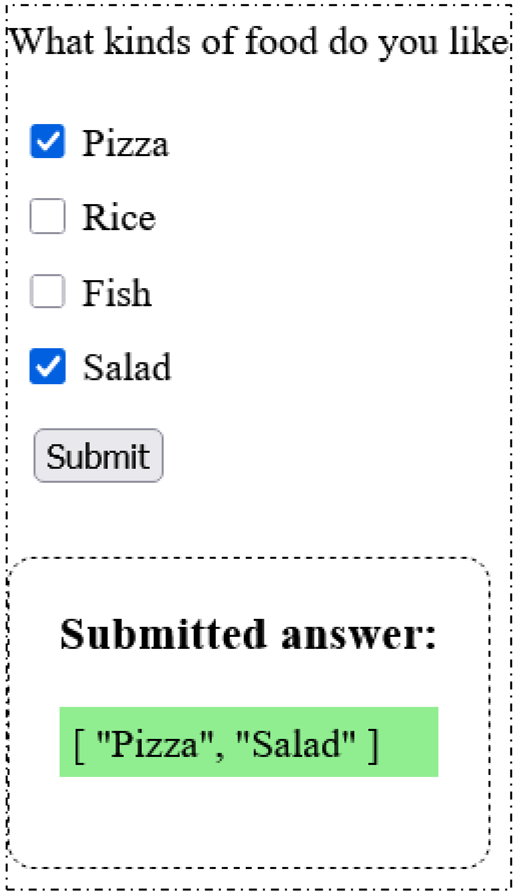
What is your favorite animal?

- Cat
- [✓ Dog]
- Turtle
- Moose

Submit

Submitted choice:

Dog



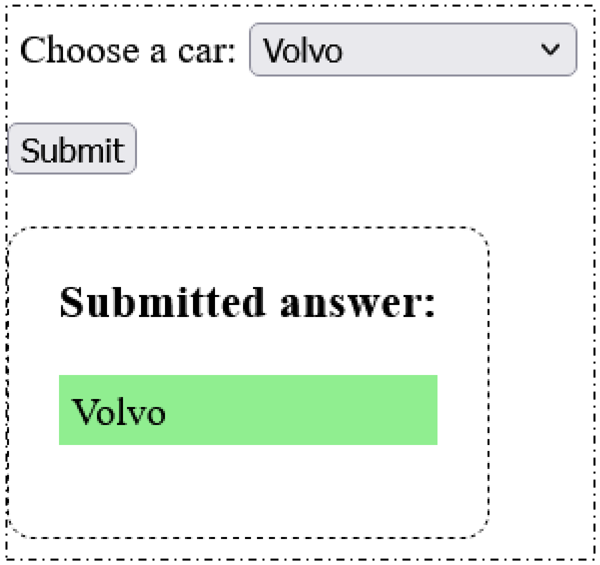
What kinds of food do you like

- [✓ Pizza]
- Rice
- Fish
- [✓ Salad]

Submit

Submitted answer:

["Pizza", "Salad"]



Choose a car: Volvo

Submit

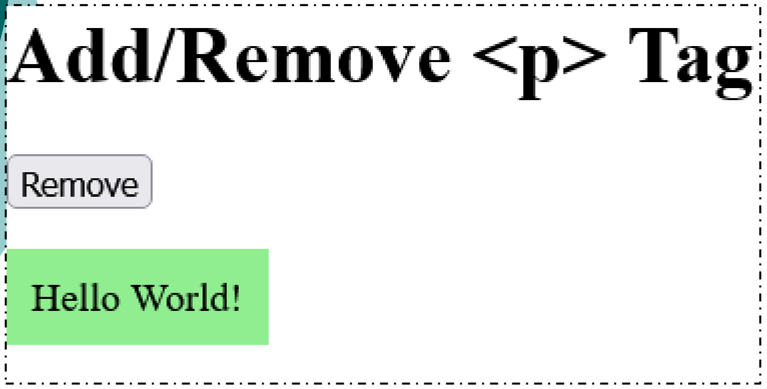
Submitted answer:

Volvo

Animations

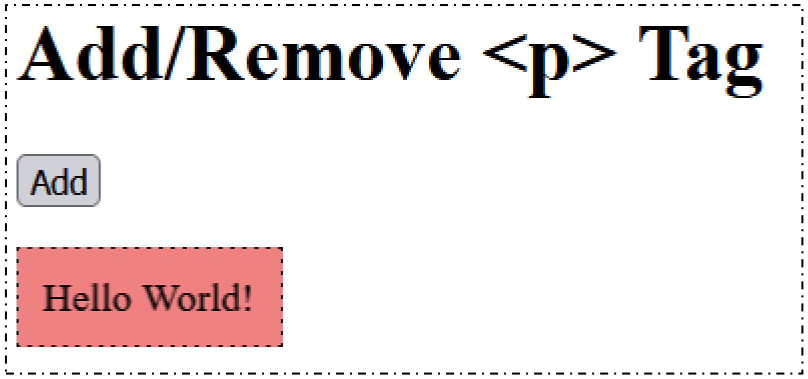
- The built-in `<Transition>` component in Vue helps us to do animations when elements are added or removed with v-if, v-show or with dynamic components
- There is nothing wrong with using plain CSS transitions and animations in other cases

Animations


Add/Remove <p> Tag

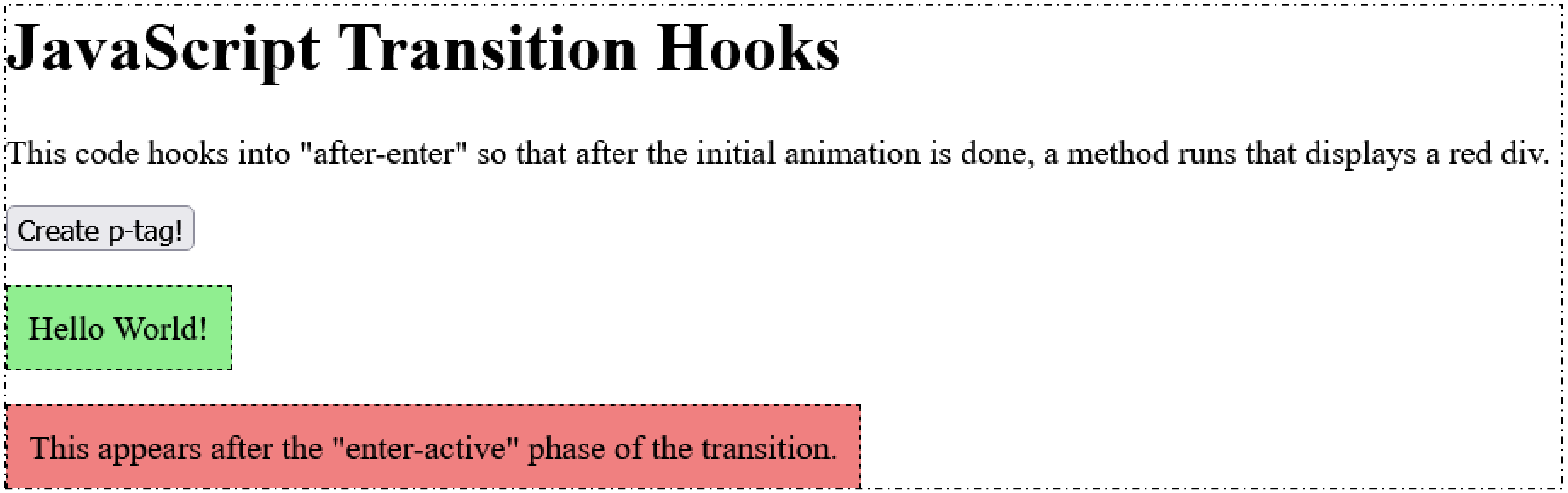
Remove

Hello World!


Add/Remove <p> Tag

Add

Hello World!


JavaScript Transition Hooks

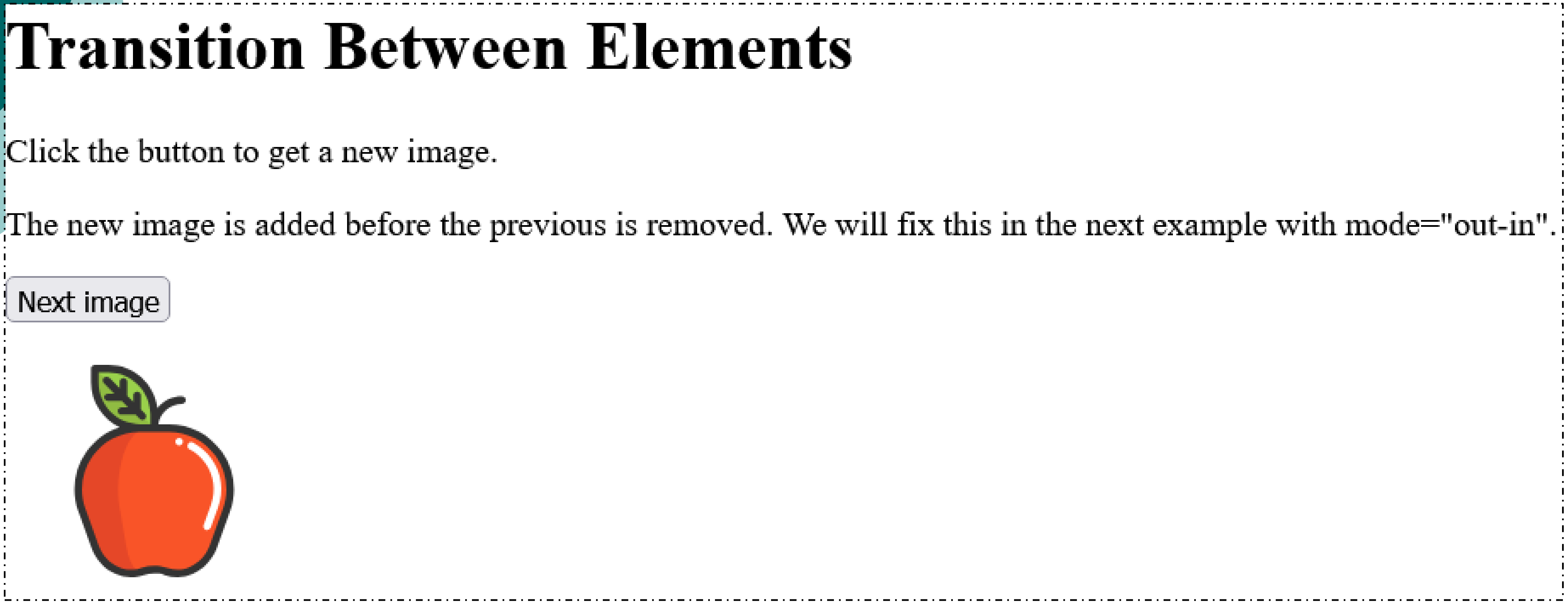
This code hooks into "after-enter" so that after the initial animation is done, a method runs that displays a red div.

Create p-tag!

Hello World!

This appears after the "enter-active" phase of the transition.

Animations



Transition Between Elements

Click the button to get a new image.

The new image is added before the previous is removed. We will fix this in the next example with mode="out-in".

Example with backend

- Create new folder with the name „`backend_example`”
  - `npm init -y`
  - `npm install express cors`
- Create a `server.js` file

`server.js`

```
const express = require('express');
const cors = require('cors');
const app = express();

app.use(cors());

app.get('/api/message', (req, res) => {
  res.json({ message: 'Hello from the backend!' });
});

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Backend is running on
http://localhost:${PORT}`);
});
```

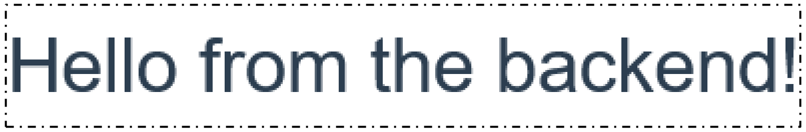
Example with backend

- Create VueJS app with the name „`frontend`”

App.vue

```
<template>
  <div class="app">
    <h1>{{ message }}</h1>
  </div>
</template>

<script>
export default {
  data() {
    return {
      message: 'Loading...'
    }
  },
  mounted() {
    fetch('http://localhost:3000/api/message')
      .then(res => res.json())
      .then(data => {
        this.message = data.message;
      });
  }
}
</script>

<style>
.app {
  font-family: Arial, sans-serif;
  text-align: center;
  margin-top: 50px;
}
</style>
```

- Run the backend: `node server.js`
- Run the VueJS frontend: `npm run dev`



```
Hello from the backend!
```

Thank you for your attention!