University of Miskolc
Faculty of Mechanical Engineering and Informatics

Web Front -end Full Stack Development

N13020104

VueJS basics

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

2025.

What is VueJS?

- VueJS is a **progressive JavaScript framework** used to develop interactive web interfaces
- Focus is more on the **view part (view layer) → front end**
- The installation of VueJS is fairly simple
- **Open source** progressive JavaScript framework
- **Created by Evan You**, an ex-employee from Google
- **The first version** of VueJS was released in **Feb 2014**
- It recently has clocked to **64,828 stars on GitHub**, making it very popular
  - open-source community support

Install Vue.js

- There are many ways to install VueJS
  - **Using the <script> tag directly in HTML file**

```html
<html>
<head>
<script type = "text/javascript" src = "vue.min.js"></script>
</head>
<body></body>
</html>
```

  - In this case have to download the `vue.min.js` file

Install Vue.js - CDN

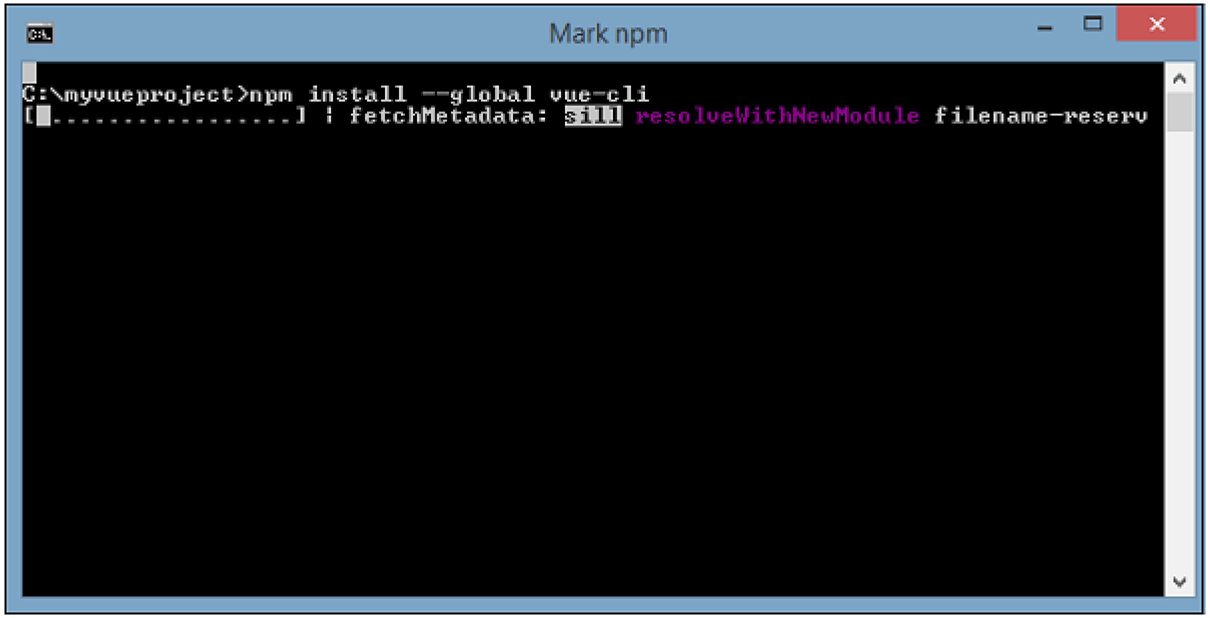
- There are many ways to install VueJS
  - **Using CDN**
    - using VueJS file from the CDN library
    - CDN: Content Delivery Network, a network of interconnected servers that speeds up webpage loading for data-heavy applications
    - The link https://unpkg.com/vue will give the latest version of VueJS
    - Can be download this vue.global.js file

```html
<html>
<head>
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
</head>
<body></body>
</html>
```

Install Vue.js - NPM

- There are many ways to install VueJS
  - **Using NPM**

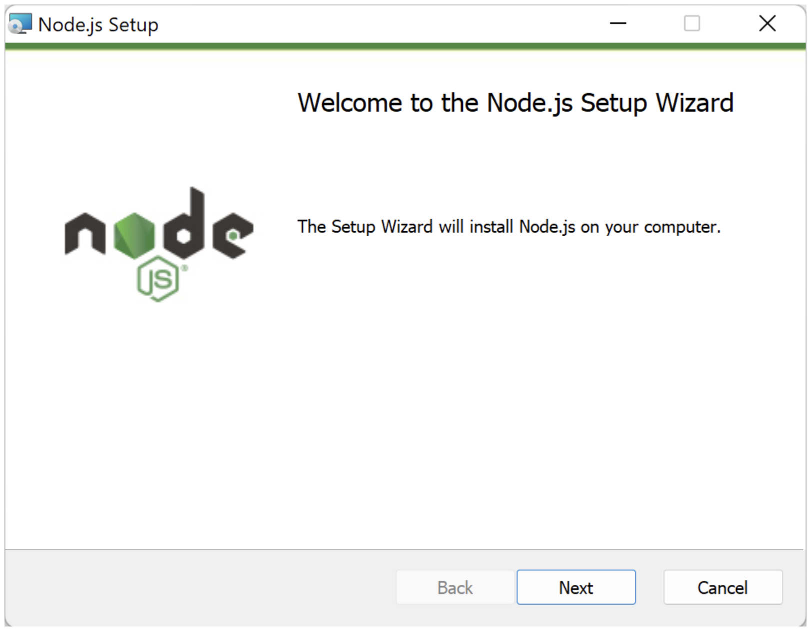
```
npm install vue
npm install --global vue-cli
```



```
C:\myvueproject>npm install --global vue-cli
[█.................] | fetchMetadata: sill resolveWithNewModule filename-reserv
```

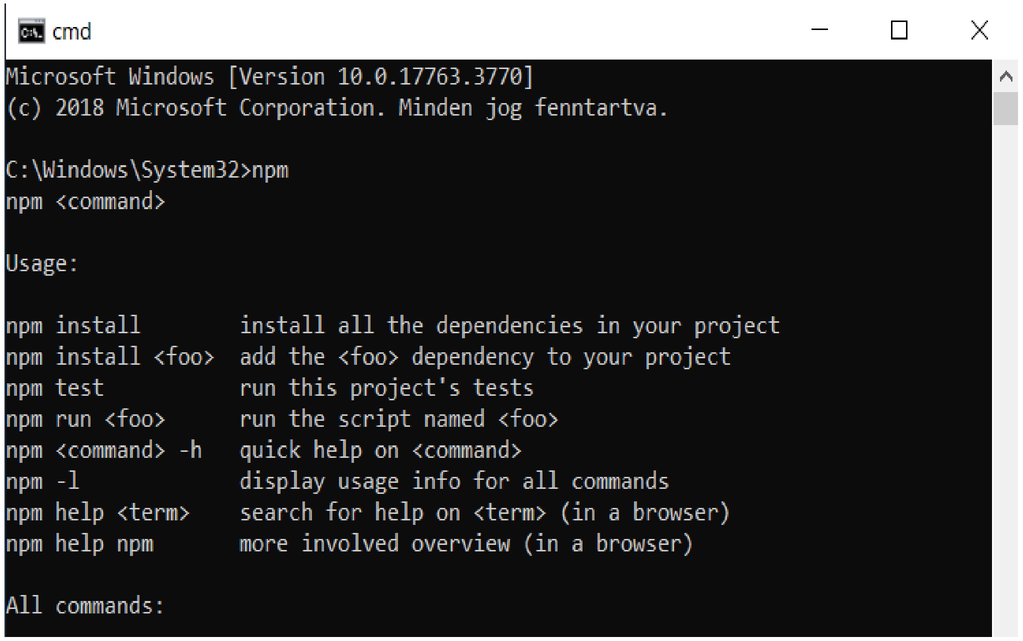
Install Node.js - NPM

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



Welcome to the Node.js Setup Wizard

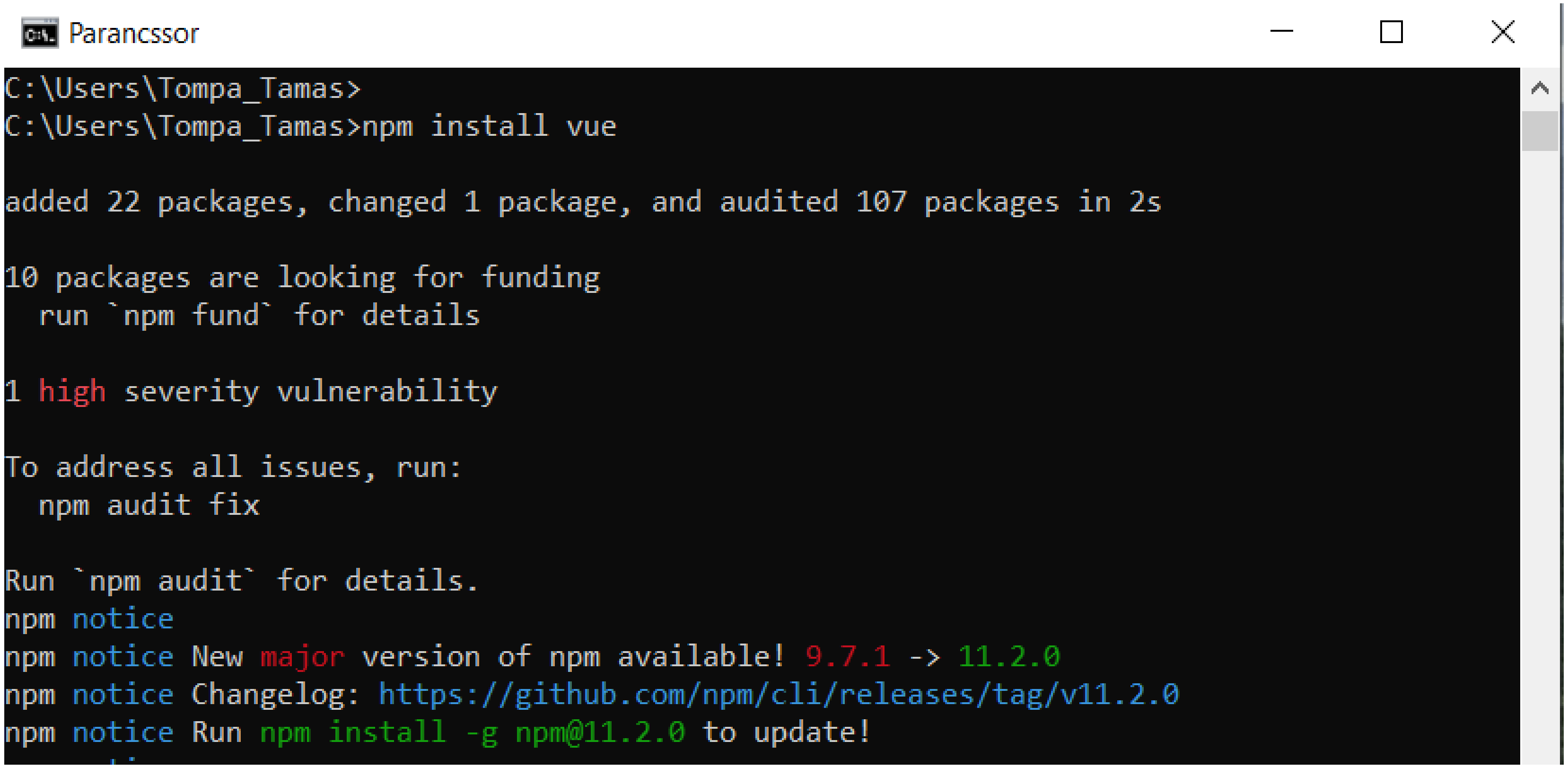
The Setup Wizard will install Node.js on your computer.



```
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

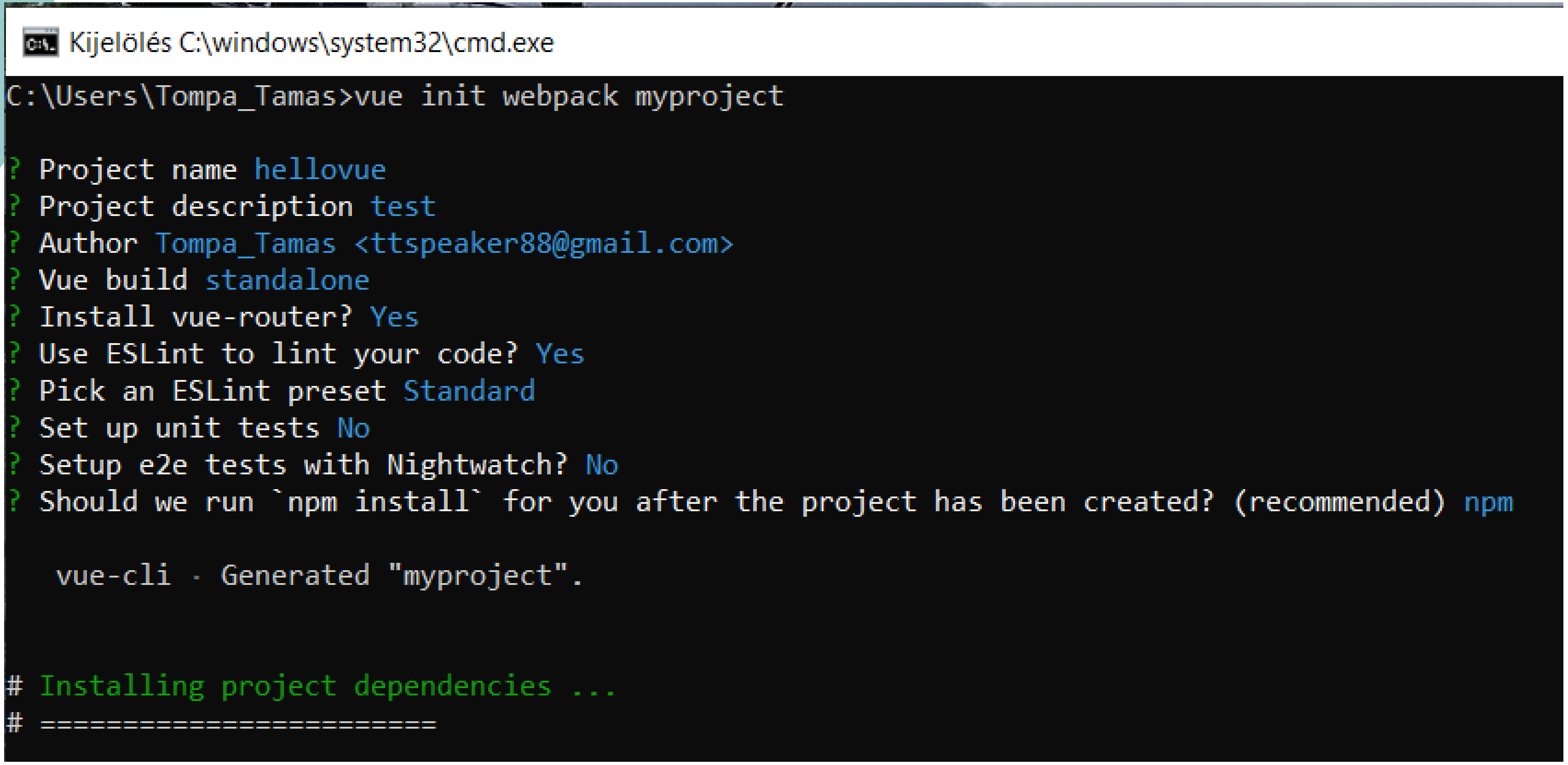
- There are many ways to install VueJS
  - **Using NPM:** `npm install vue`



```
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 HelloWorld project

Create „myproject" Vue project: `vue init webpack myproject`



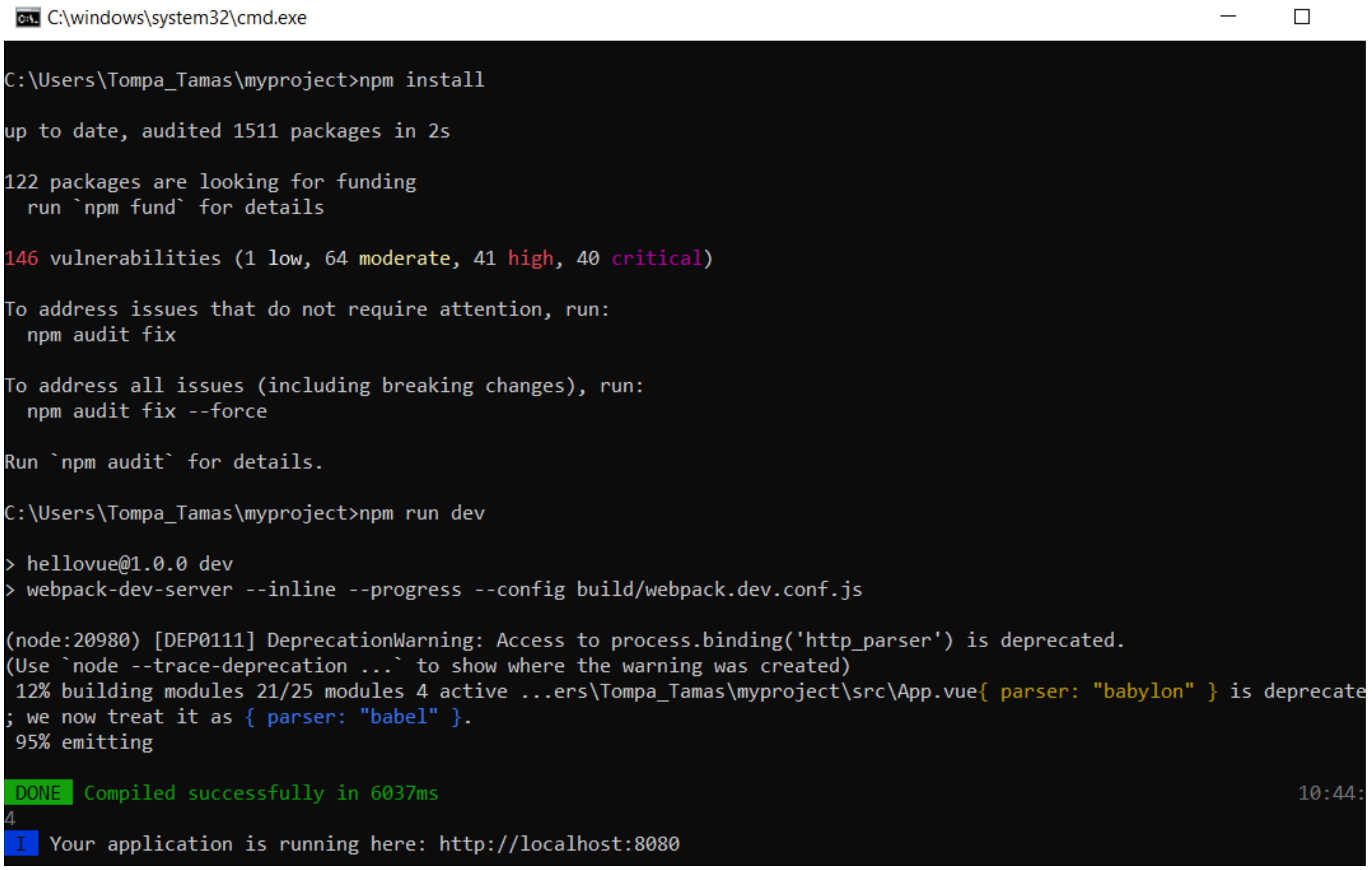
Kijelölés C:\windows\system32\cmd.exe

```
C:\Users\Tompa_Tamas>vue init webpack myproject

? Project name hellovue
? Project description test
? Author Tompa_Tamas <ttspeaker88@gmail.com>
? Vue build standalone
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Set up unit tests No
? Setup e2e tests with Nightwatch? No
? Should we run `npm install` for you after the project has been created? (recommended) npm

   vue-cli · Generated "myproject".


# Installing project dependencies ...
# ========================
```

Build HelloWorld project

Build the project:

- `cd myproject`
- `npm install`
- `npm run dev`

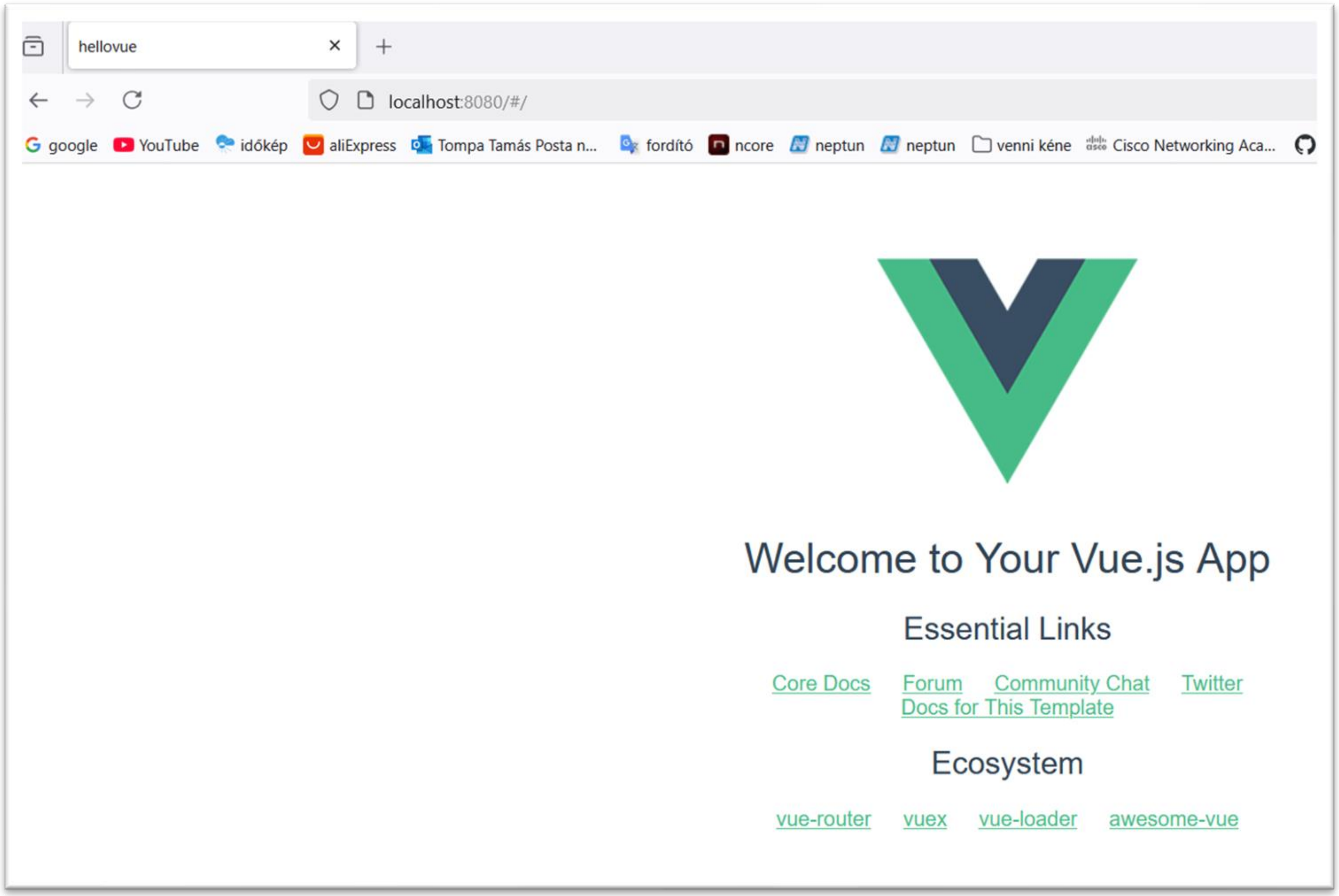


C:\windows\system32\cmd.exe

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

up to date, audited 1511 packages in 2s

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

146 vulnerabilities (1 low, 64 moderate, 41 high, 40 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\Tompa_Tamas\myproject>npm run dev

> hellovue@1.0.0 dev
> webpack-dev-server --inline --progress --config build/webpack.dev.conf.js

(node:20980) [DEP0111] DeprecationWarning: Access to process.binding('http_parser') is deprecated.
(Use `node --trace-deprecation ...` to show where the warning was created)
 12% building modules 21/25 modules 4 active ...ers\Tompa_Tamas\myproject\src\App.vue{ parser: "babylon" } is deprecate
; we now treat it as { parser: "babel" }.
 95% emitting

 DONE  Compiled successfully in 6037ms                                                                        10:44:
4
 I  Your application is running here: http://localhost:8080
```

Run HelloWorld project

http://localhost:8080/#/


hellovue

localhost:8080/#/

Welcome to Your Vue.js App

Essential Links

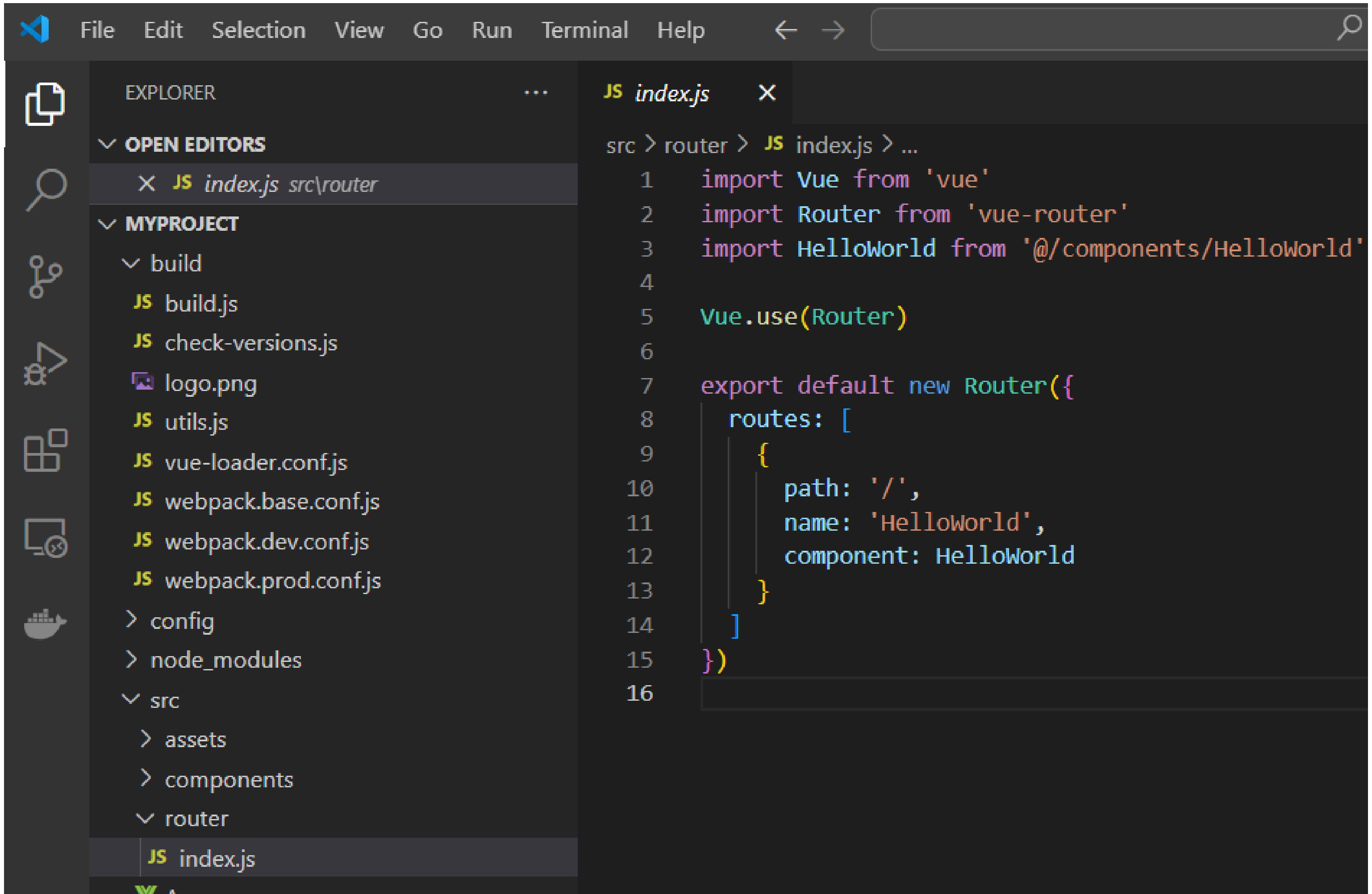
Core Docs Forum Community Chat Twitter
Docs for This Template

Ecosystem

vue-router vuex vue-loader awesome-vue

Open HelloWorld project

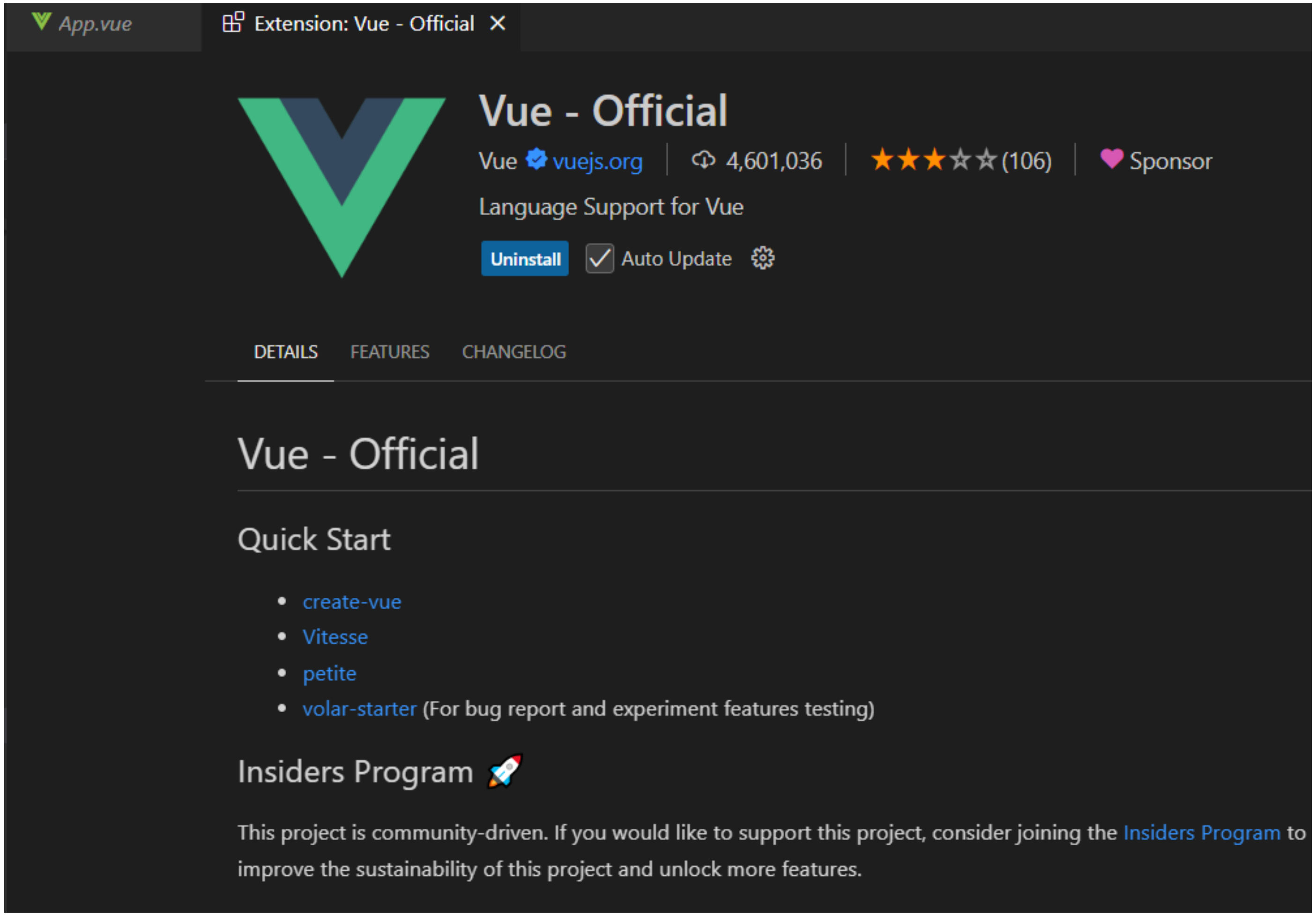
- Open this project in the VSCode



```
import Vue from 'vue'
import Router from 'vue-router'
import HelloWorld from '@/components/HelloWorld'

Vue.use(Router)

export default new Router({
  routes: [
    {
      path: '/',
      name: 'HelloWorld',
      component: HelloWorld
    }
  ]
})

```

Open HelloWorld project

- Install the Vuejs plugin in the VSCode



Vue - Official

Vue vuejs.org | 4,601,036 | ★★★☆☆ (106) | Sponsor

Language Support for Vue

Vue - Official

Quick Start

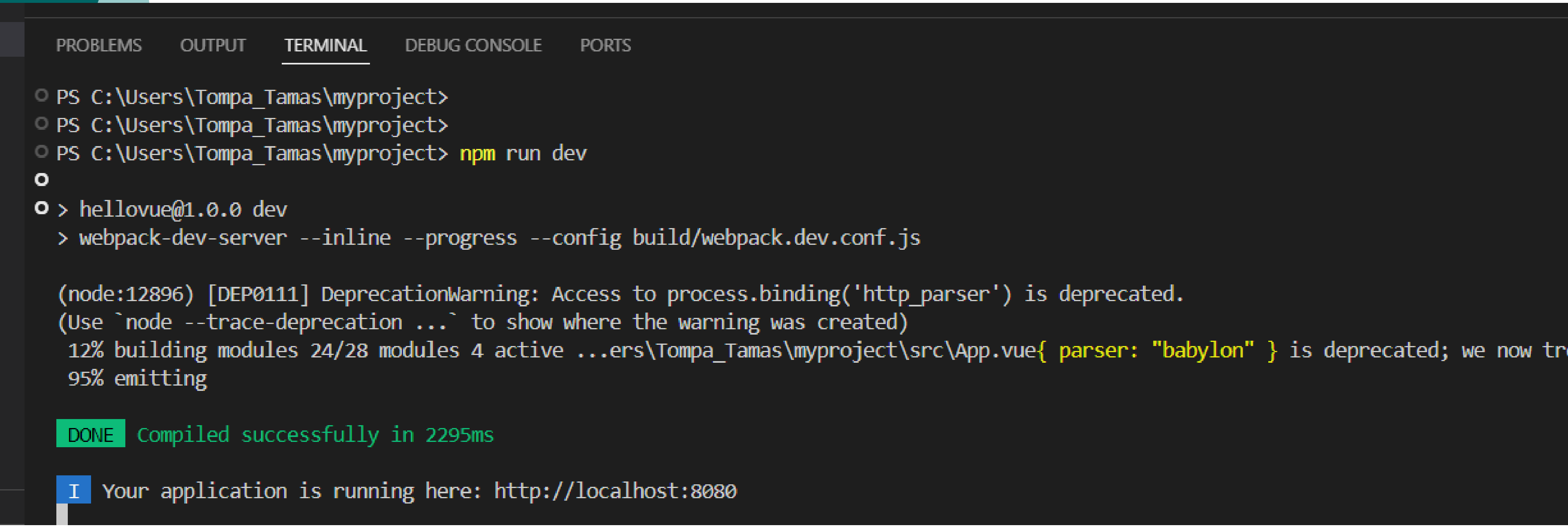
- create-vue
- Vitesse
- petite
- volar-starter (For bug report and experiment features testing)

Insiders Program 🚀

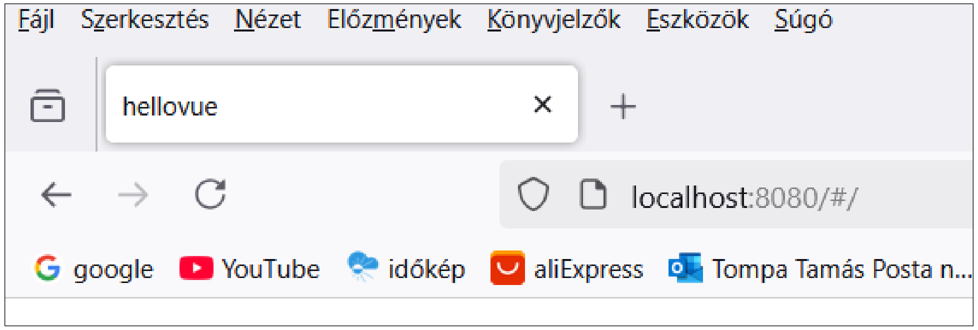
This project is community-driven. If you would like to support this project, consider joining the Insiders Program to improve the sustainability of this project and unlock more features.

Build HelloWorld project

- Build the project using by the VSCode



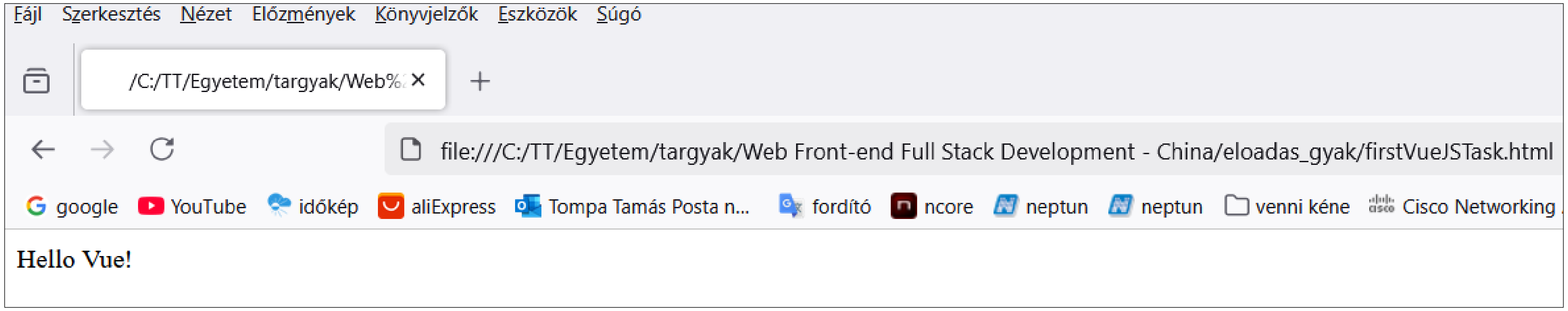
```
PS C:\Users\Tompa_Tamas\myproject>
PS C:\Users\Tompa_Tamas\myproject>
PS C:\Users\Tompa_Tamas\myproject> npm run dev

> hellovue@1.0.0 dev
> webpack-dev-server --inline --progress --config build/webpack.dev.conf.js

(node:12896) [DEP0111] DeprecationWarning: Access to process.binding('http_parser') is deprecated.
(Use `node --trace-deprecation ...` to show where the warning was created)
 12% building modules 24/28 modules 4 active ...ers\Tompa_Tamas\myproject\src\App.vue{ parser: "babylon" } is deprecated; we now tr
 95% emitting

 DONE  Compiled successfully in 2295ms

 I  Your application is running here: http://localhost:8080
```



Example1 using CDN

```html
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<div id="app">{{ message }}</div>

<script>
  const { createApp, ref } = Vue

  createApp({
    setup() {
      const message = ref('Hello Vue!')
      return {
        message
      }
    }
  }).mount('#app')
</script>
```

- Save into HTML file (HelloVue.html)


Fájl Szerkesztés Nézet Előzmények Könyvjelzők Eszközök Súgó
/C:/TT/Egyetem/targyak/Web%
google YouTube időkép aliExpress Tompa Tamás Posta n... fordító ncore neptun neptun venni kéne Cisco Networking

file:///C:/TT/Egyetem/targyak/Web Front-end Full Stack Development - China/eloadas_gyak/firstVueJSTask.html

```
Hello Vue!
```

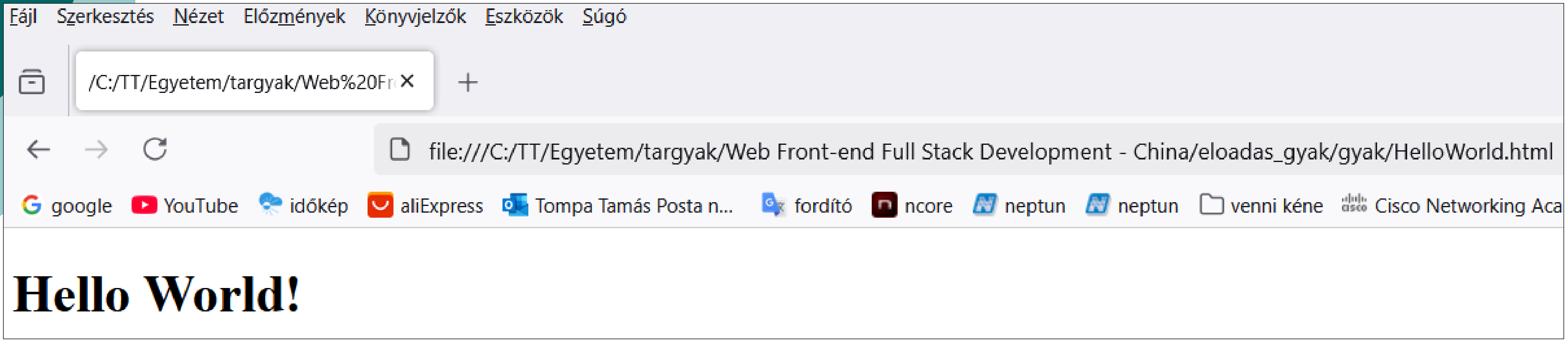
Example2 using CDN

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

  <script
src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <script>
    const app = Vue.createApp({
      data() {
        return {
          message: "Hello World!"
        }
      }
    });

    app.mount('#app');
  </script>
```

- Save into HTML file (HelloWorld.html)

Example2 using CDN

- Result (HelloWorld.html):



file:///C:/TT/Egyetem/targyak/Web Front-end Full Stack Development - China/eloadas_gyak/gyak/HelloWorld.html

Hello World!

Steps to create simple page

- **5 basic steps to create simple page:**
  - 1. Start with a basic HTML file

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>My first Vue page</title>
</head>
<body>

</body>
</html>
```

  - 2. Add a `<div>` tag with `id="app"` for Vue to connect with

```html
<body>
  <div id="app"></div>
</body>
```

  - 3. Tell the browser how to handle Vue code by adding a <script> tag with a link to Vue

```html
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

Steps to create simple page

- **5 basic steps to create simple page:**
  - 4. Add a `<script>` tag with the Vue instance inside

```
const app = Vue.createApp({
   data() {
     return {
       message: "Hello World!"
     }
   }
 })

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

  - 5. Connect the Vue instance to the `<div id="app">` tag

```
<div id="app"> {{ message }} </div>
```

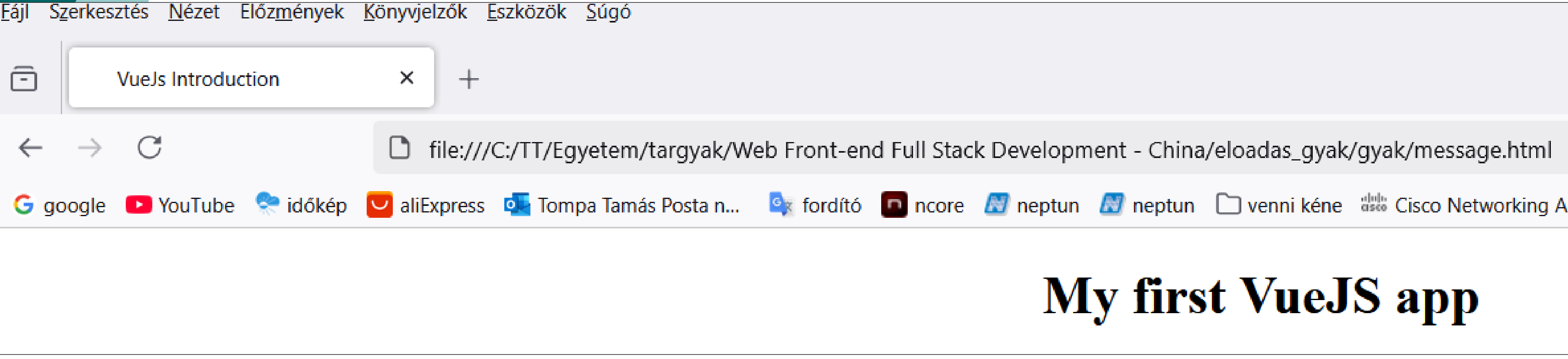
Example3 using CDN

```
<html>
   <head>
      <title>VueJs Introduction</title>
      <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
   </head>
   <body>
      <div id="intro" style="text-align:center;">
         <h1>{{ message }}</h1>
      </div>
      <script type="text/javascript">
         const app = Vue.createApp({
            data() {
               return {
                  message: 'My first VueJS app'
               };
            }
         });

         app.mount('#intro');
      </script>
   </body>
</html>
```

- Save into HTML file (Message.html)

Example3 using CDN

- Result (Message.html):


Fájl Szerkesztés Nézet Előzmények Könyvjelzők Eszközök Súgó
VueJs Introduction
file:///C:/TT/Egyetem/targyak/Web Front-end Full Stack Development - China/eloadas_gyak/gyak/message.html
google YouTube időkép aliExpress Tompa Tamás Posta n... fordító ncore neptun neptun venni kéne Cisco Networking A

My first VueJS app

Text interpolation

- Text interpolation is **when text is taken from the Vue instance to show on the web page**:

```
<div id="app"> {{ message }} </div>
```

- Then the browser finds the text inside the 'message' property of the Vue instance and translates the Vue code into this:

```
<div id="app">Hello World!</div>
```

Instances

- To start with VueJS, we need to create the instance of Vue, which is called the root Vue Instance

```
const { createApp } = Vue;

const app = createApp({
   // options
});

app.mount("#app");
```

- The Firstname: {{firstname}} value will be replaced inside the interpolation, i.e. {{}} with the value assigned in the data object, i.e. Tamas (The same goes for last name)

- data(): a function that returns an object. Vue makes the object's properties reactive by creating getters and setters, ensuring that changes automatically update the DOM

```
const app = Vue.createApp({
    data() {
        return _obj;
    }
});
```

Instances

Save into HTML file (Instances.html)

```
<html>
   <head>
      <title>VueJs Instance</title>
      <script
src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
   </head>
   <body>
      <div id="vue_det">
         <h1>Firstname : {{ firstname }}</h1>
         <h1>Lastname : {{ lastname }}</h1>
         <h1>{{ mydetails() }}</h1>
      </div>
      <script type="text/javascript"
src="vue_instance.js"></script>
   </body>
</html>
```

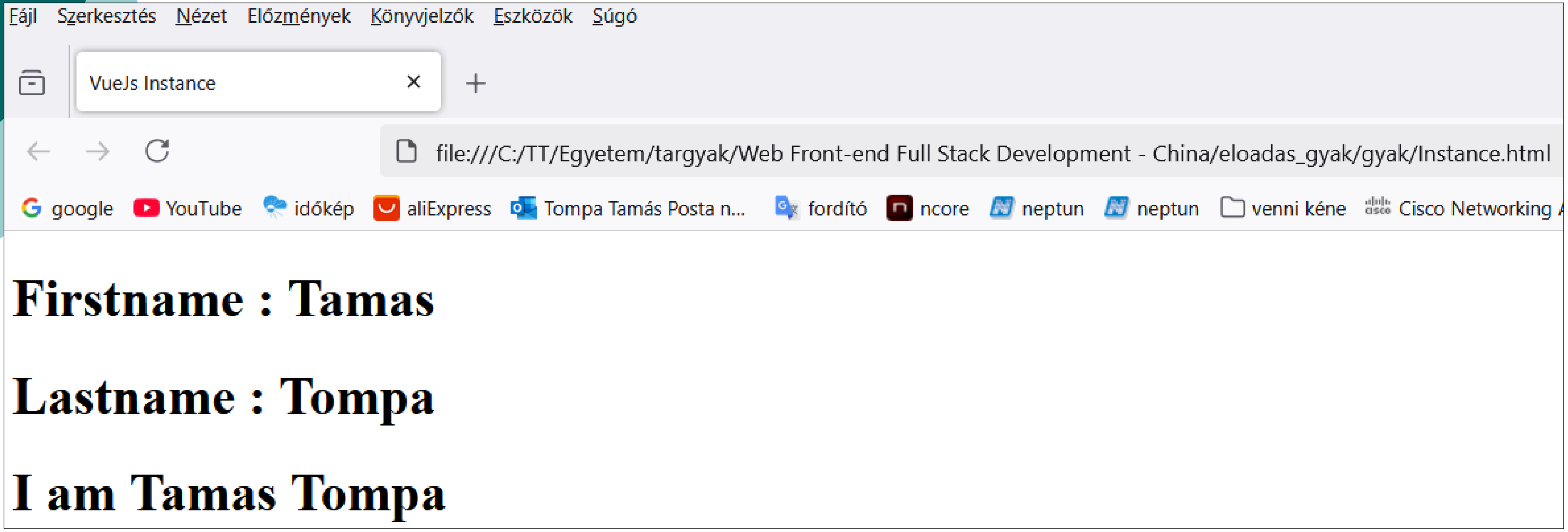
Instances

Save into JS file (vue_instance.js)

```
const { createApp } = Vue;

const app = createApp({
   data() {
      return {
         firstname: „Tamas",
         lastname: „Tompa",
         address: „Hungary"
      };
   },
   methods: {
      mydetails() {
         return "I am " + this.firstname + " " + this.lastname;
      }
   }
});

app.mount("#vue_det");
```

Instances

Result: (Instances.html, vue_instance.js)


Fájl Szerkesztés Nézet Előzmények Könyvjelzők Eszközök Súgó
VueJs Instance
file:///C:/TT/Egyetem/targyak/Web Front-end Full Stack Development - China/eloadas_gyak/gyak/Instance.html
google YouTube időkép aliExpress Tompa Tamás Posta n... fordító ncore neptun neptun venni kéne Cisco Networking

Firstname : Tamas

Lastname : Tompa

I am Tamas Tompa

Directives

- Vue directives are **special HTML attributes with the prefix v-** that give the HTML tag extra functionality

- Vue directives **connect to the Vue instance** to create dynamic and reactive user interfaces

- With Vue, creating responsive pages is much easier and requires less code compared to traditional JavaScript methods

Directives

Vue.js

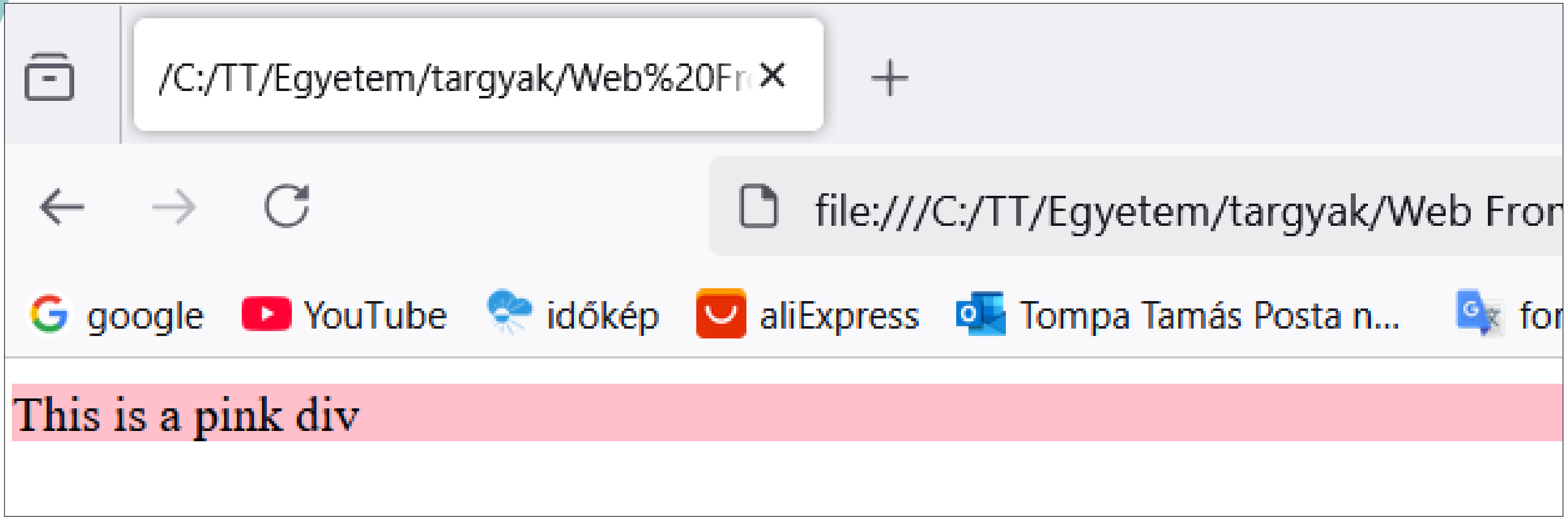
Directive	Details
v-bind	Connects an attribute in an HTML tag to a data variable inside the Vue instance.
v-if	Creates HTML tags depending on a condition. Directives v-else-if and v-else are used together with the v-if directive.
v-show	Specifies if an HTML element should be visible or not depending on a condition.
v-for	Creates a list of tags based on an array in the Vue instance using a for-loop.
v-on	Connects an event on an HTML tag to a JavaScript expression or a Vue instance method. We can also define more specifically how our page should react to a certain event by using event-modifiers.
v-model	Used in HTML forms with tags like <form>, <input> and <button>. Creates a two way binding between an input element and a Vue instance data property.

Example: v-bind Directive

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    .pinkBG {
      background-color: pink;
    }
  </style>
</head>
<body>
<div id="app">
    <div v-bind:class="vueClass"></div>
  </div>

  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <script>
    const app = Vue.createApp({
      data() {
        return {
          vueClass: "pinkBG"
        }
      }
    })
    app.mount('#app')
  </script>
</body>
</html>
```

Example: v-bind Directive

V-bind_directive.html


/C:/TT/Egyetem/targyak/Web%20Fr
file:///C:/TT/Egyetem/targyak/Web Fron
google
YouTube
időkép
aliExpress
Tompa Tamás Posta n...
for

This is a pink div

v-bind Directive

- The v-bind directive lets us **bind an HTML attribute to data in the Vue instance**
- This makes it easy to **change the attribute value dynamically**
- Syntax:

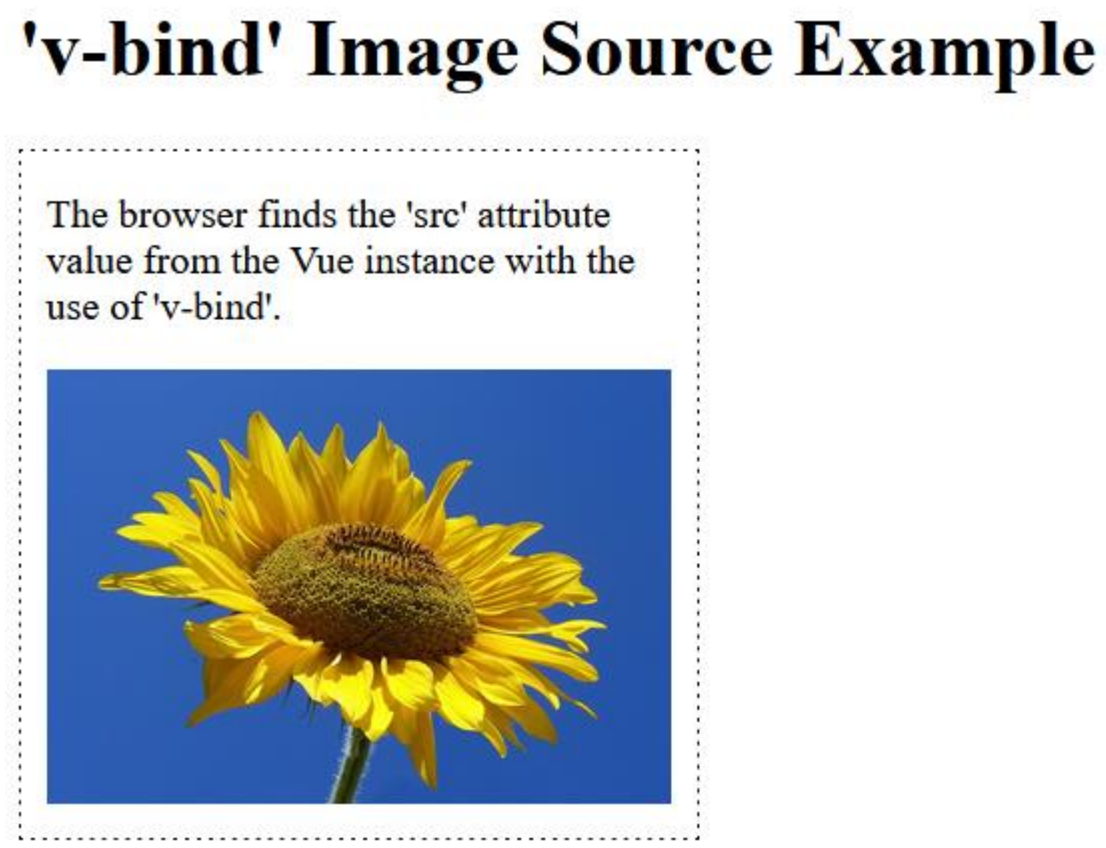
```
<div v-bind:[attribute]="[Vue data]"></div>
```

- Example:

```
<img v-bind:src="url">
```

Example: v-bind Directive

V-bind_image.html



'v-bind' Image Source Example

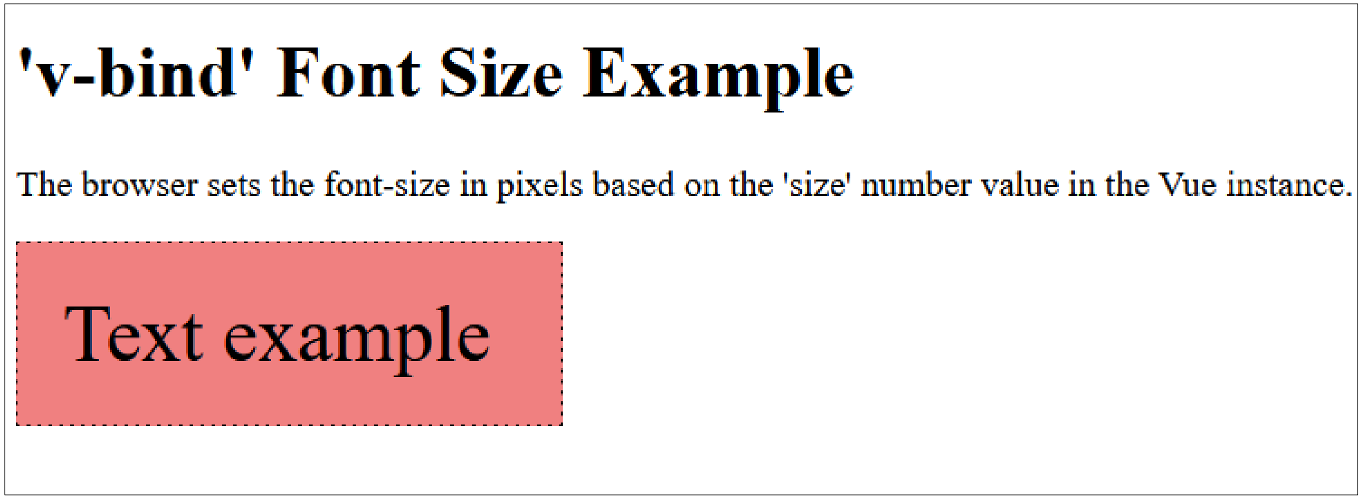
The browser finds the 'src' attribute value from the Vue instance with the use of 'v-bind'.

Example: v-bind Directive

- The font size number value is stored the Vue data property ‚size’:

```
<div v-bind:style="{ fontSize: size + 'px' }">
  Text example
</div>
```

V-bind_font-size.html



'v-bind' Font Size Example

The browser sets the font-size in pixels based on the 'size' number value in the Vue instance.

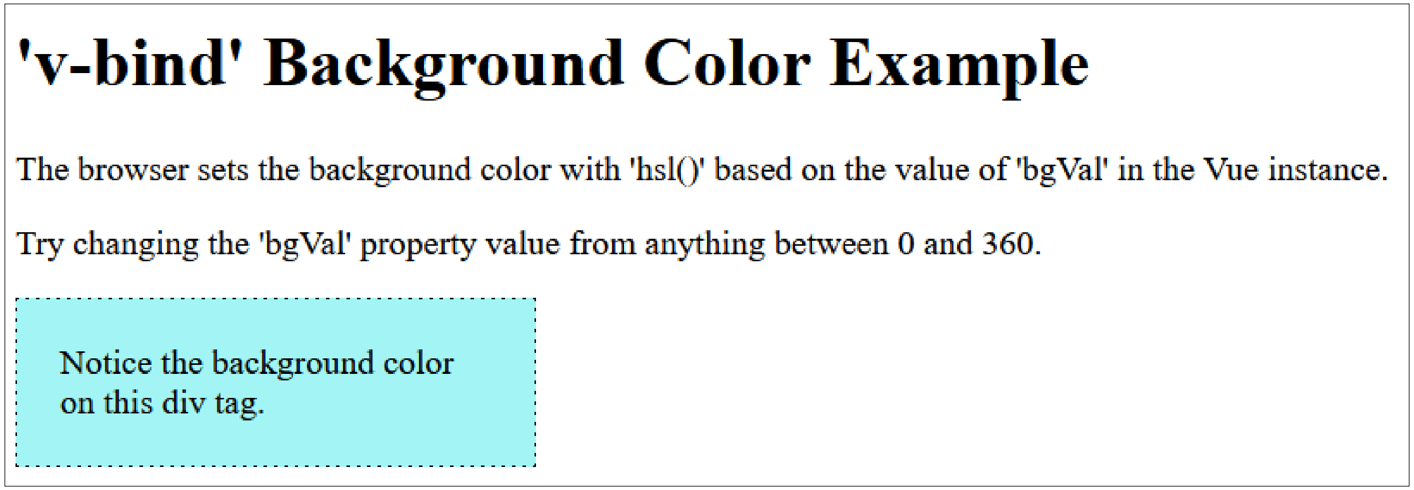
Text example

Example: v-bind Directive

- The background color depends on the 'bgVal' data property value inside the Vue instance:

```
<div v-bind:style="{ backgroundColor: 'hsl('+bgVal+',80%,80%)' }">
  Notice the background color on this div tag.
</div>
```

V-bind_bg-color.html



'v-bind' Background Color Example

The browser sets the background color with 'hsl()' based on the value of 'bgVal' in the Vue instance.

Try changing the 'bgVal' property value from anything between 0 and 360.

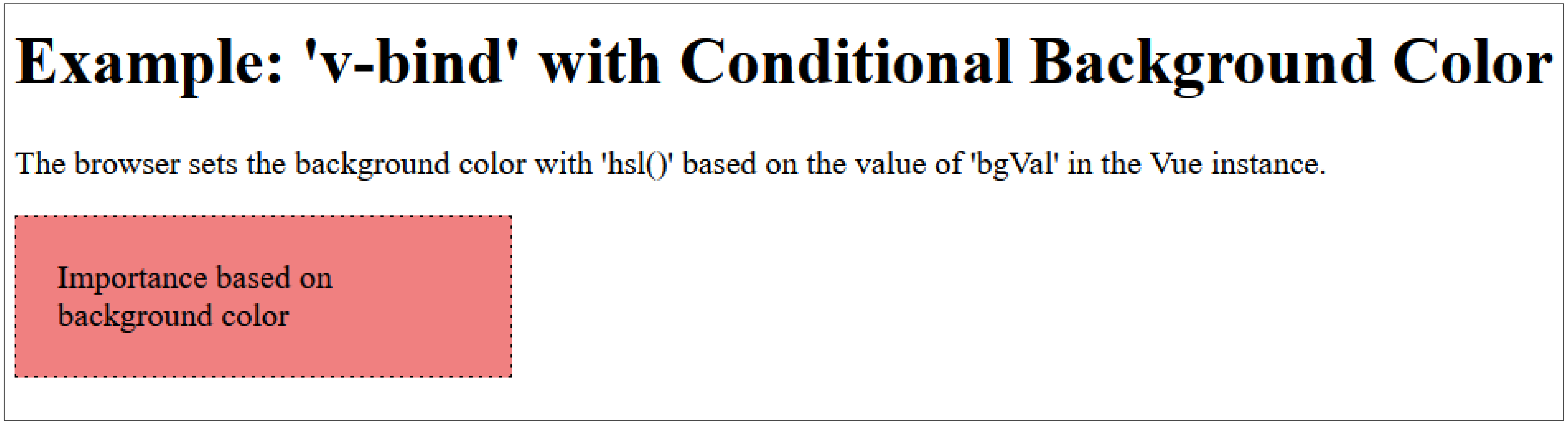
Notice the background color on this div tag.

Example: v-bind Directive

- The background color is set with a JavaScript conditional (ternary) expression depending on whether the 'isImportant' data property value is ‚true' or ‚false':

```
<div v-bind:style="{ backgroundColor: isImportant ? 'lightcoral' : 'lightgray' }">
  Conditional background color
</div>
```

V-bind_bg-color-elvis.html



Example: 'v-bind' with Conditional Background Color

The browser sets the background color with 'hsl()' based on the value of 'bgVal' in the Vue instance.

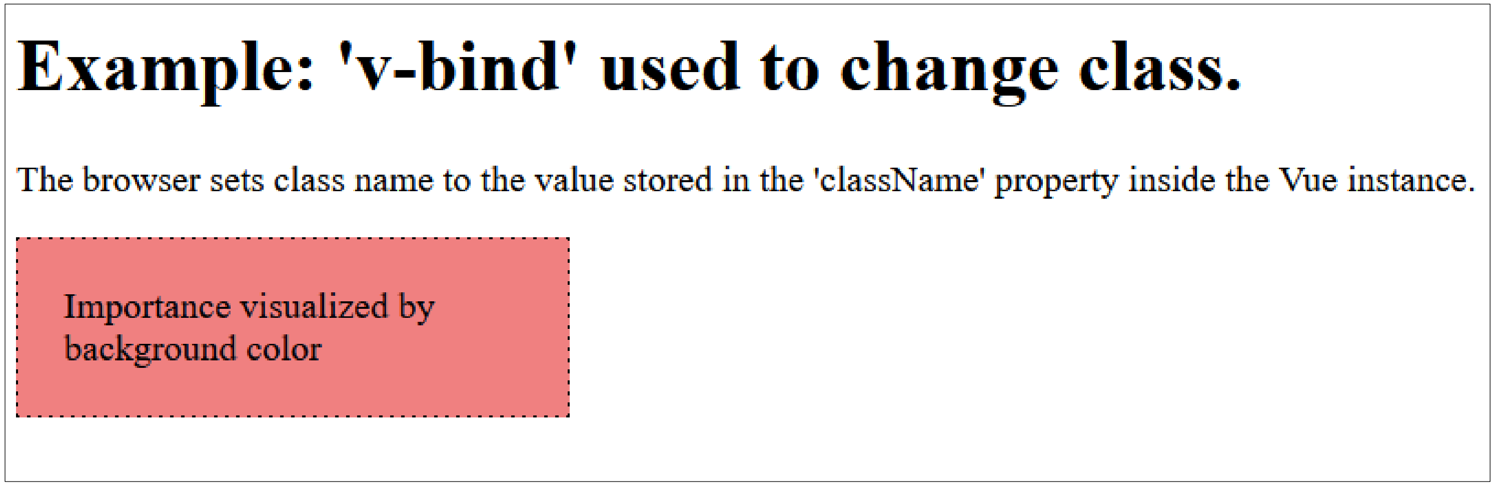
Importance based on background color

Example: v-bind Directive

- We can use v-bind to change the class attribute
- The value of `v-bind:class` can be a variable:

```
<div v-bind:class="className">
  The class is set with Vue
</div>
```

V-bind_class-change.html



Example: 'v-bind' used to change class.

The browser sets class name to the value stored in the 'className' property inside the Vue instance.

Importance visualized by background color

Example: v-bind Directive

- The shorthand for 'v-bind:' is simply ':'
- Here we just write ':' instead of 'v-bind:':

```
<div :class="{ impClass: isImportant }">
  The class is set conditionally to change the background color
</div>
```

v-if Directive

- It is a lot easier to **create an HTML element depending on a condition** in Vue with the v-if directive than with plain JavaScript
- Just write the if-statement directly in the HTML element you want to create conditionally
- Conditional rendering in Vue is done by using the **v-if, v-else-if** and **v-else** directives

```
<p v-if="typewritersInStock">
  in stock
</p>

<p v-else>
  not in stock
</p>
```

v-if Directive

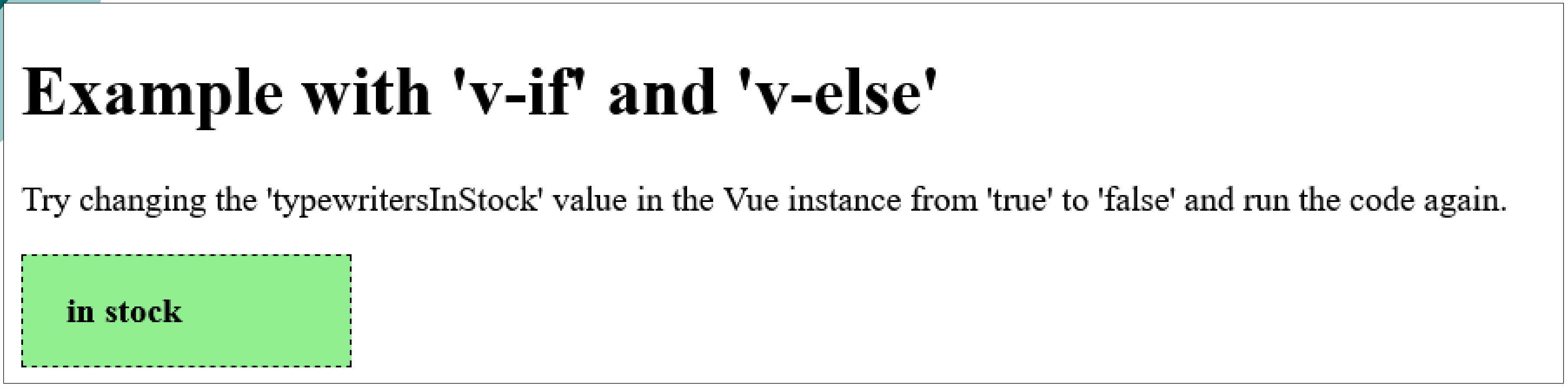
- A condition, or "if-statement", is something that is either true or false
- We use comparison operators like **<, >=** or **!==** to do such checks
- Comparison checks can also be combined with logical operators such as **&&** or **||**

```
<p v-if="typewriterCount > 0">
  in stock
</p>

<p v-else>
  not in stock
</p>
```

v-if Directive

- Example

  V_if-else.html



Example with 'v-if' and 'v-else'

Try changing the 'typewritersInStock' value in the Vue instance from 'true' to 'false' and run the code again.

in stock

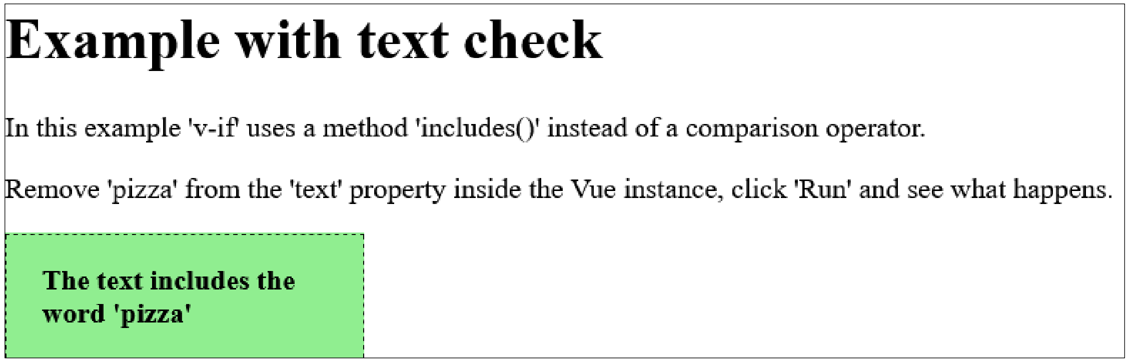
v-if Directive

- Example
  - In this example **'v-if' uses a method 'includes()'** instead of a comparison operator
  - **Remove 'pizza' from the 'text' property** inside the Vue instance, run again and see what happens

v-if Directive

- Example

  V-if_includes().html



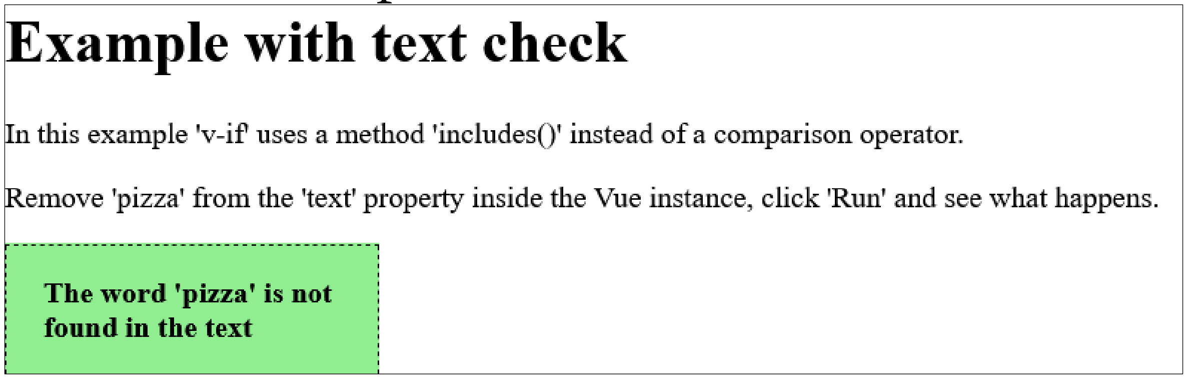
Example with text check

In this example 'v-if' uses a method 'includes()' instead of a comparison operator.

Remove 'pizza' from the 'text' property inside the Vue instance, click 'Run' and see what happens.

The text includes the word 'pizza'

- After deleted the „pizza" word:



Example with text check

In this example 'v-if' uses a method 'includes()' instead of a comparison operator.

Remove 'pizza' from the 'text' property inside the Vue instance, click 'Run' and see what happens.

The word 'pizza' is not found in the text

v-if Directive

- Example

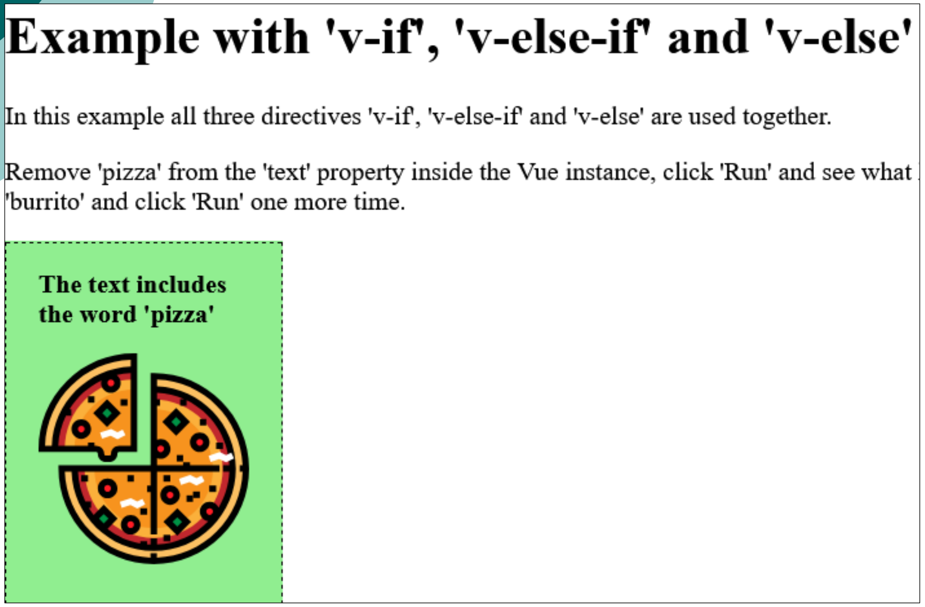
  V-if_v-else_image.html

  - In this example all three directives **'v-if', 'v-else-if' and 'v-else' are used together**
  - **Remove 'pizza'** from the 'text' property inside the Vue instance, and see what happens. Then **remove 'burrito'** and **see what happens** one more time
  - **Loading image depending on the content of the given text**

v-if Directive

- Example

  V-if_v-else_image.html

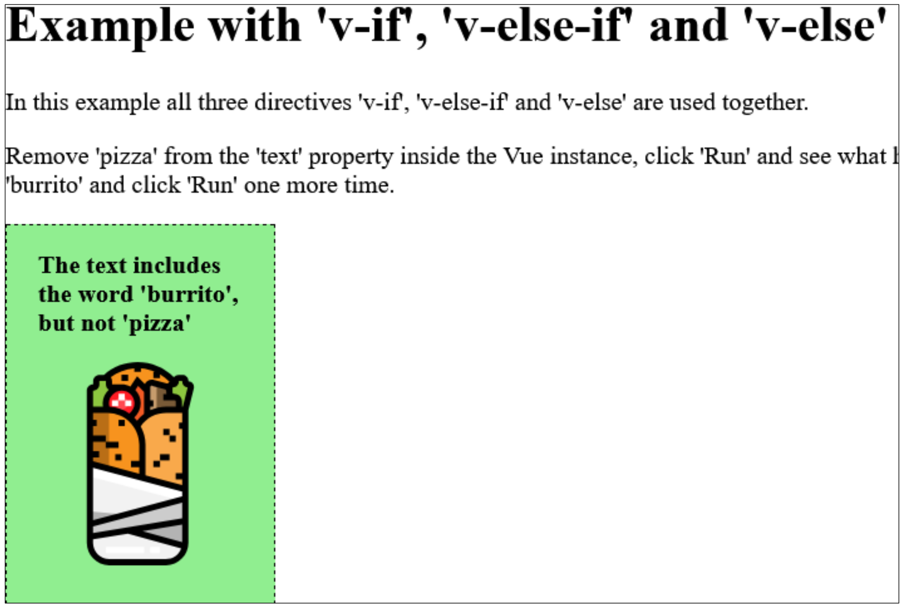


Example with 'v-if', 'v-else-if' and 'v-else'

In this example all three directives 'v-if', 'v-else-if' and 'v-else' are used together.

Remove 'pizza' from the 'text' property inside the Vue instance, click 'Run' and see what 'burrito' and click 'Run' one more time.

The text includes the word 'pizza'



Example with 'v-if', 'v-else-if' and 'v-else'

In this example all three directives 'v-if', 'v-else-if' and 'v-else' are used together.

Remove 'pizza' from the 'text' property inside the Vue instance, click 'Run' and see what h 'burrito' and click 'Run' one more time.

The text includes the word 'burrito', but not 'pizza'

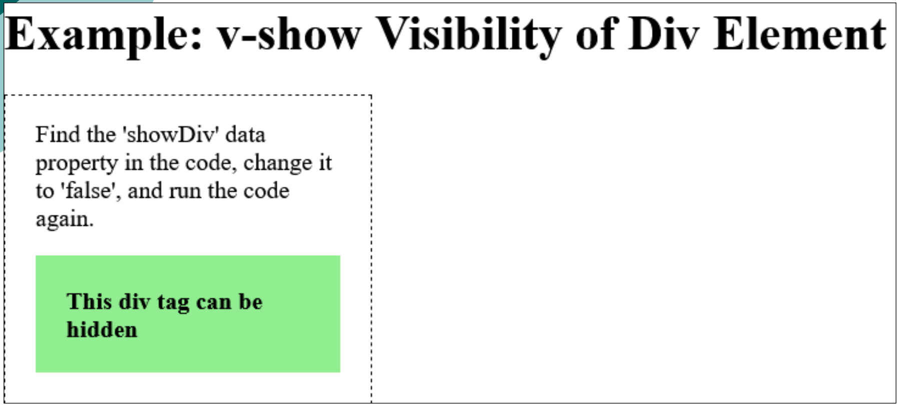
v-show Directive

- It **hides an element** when the condition is 'false' by setting the CSS 'display' property value to 'none’
- After writing v-show as an HTML attribute we must give a conditon to decide to have the tag visible or not
- Syntax:

```
<div v-show="showDiv">This div tag can be hidden</div>
```

v-show Directive

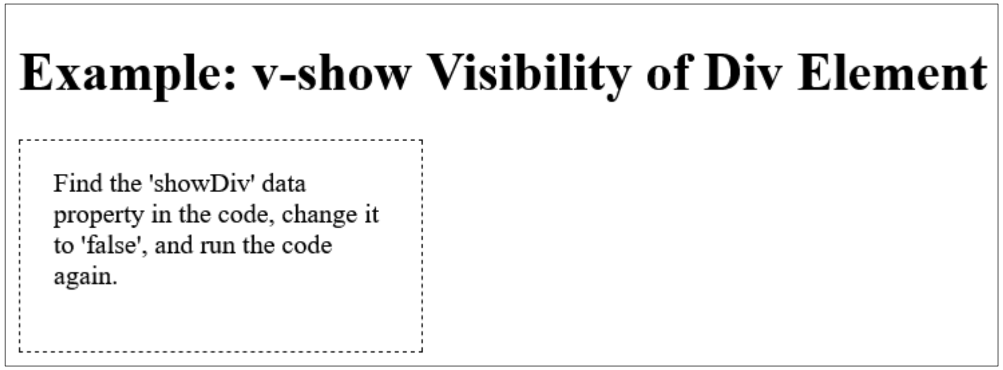
- Example V-show_div.html
  - Display the <div> element only if the showDiv property is set to 'true'



Example: v-show Visibility of Div Element

Find the 'showDiv' data property in the code, change it to 'false', and run the code again.

This div tag can be hidden



Example: v-show Visibility of Div Element

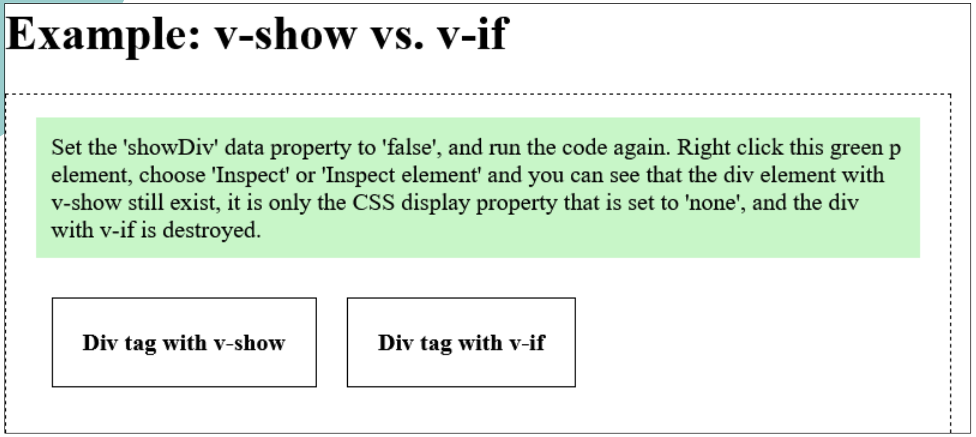
Find the 'showDiv' data property in the code, change it to 'false', and run the code again.

```
const app = Vue.createApp({
  data() {
    return {
      showDiv: true
    }
  }
})
```

```
const app = Vue.createApp({
  data() {
    return {
      showDiv: false
    }
  }
})
```

v-show Directive

- Example V-show_div2.html
  - Display the <div> element only if the showDiv property is set to 'true'

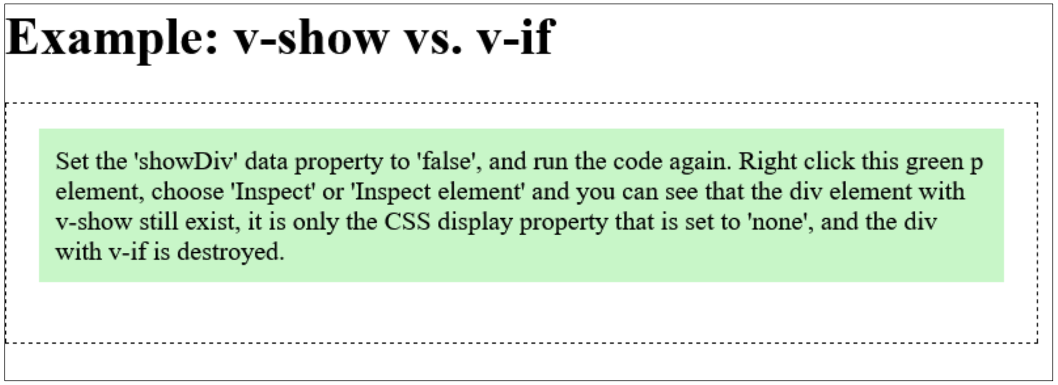


Example: v-show vs. v-if

Set the 'showDiv' data property to 'false', and run the code again. Right click this green p element, choose 'Inspect' or 'Inspect element' and you can see that the div element with v-show still exist, it is only the CSS display property that is set to 'none', and the div with v-if is destroyed.

Div tag with v-show **Div tag with v-if**

```
const app = Vue.createApp({
  data() {
    return {
      showDiv: true
    }
  }
})
```



Example: v-show vs. v-if

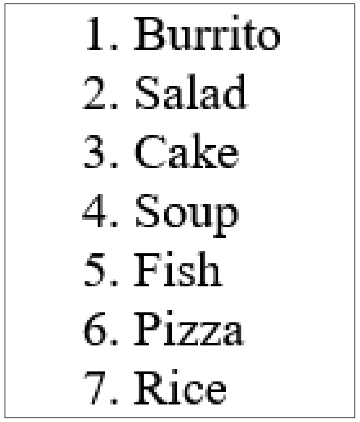
Set the 'showDiv' data property to 'false', and run the code again. Right click this green p element, choose 'Inspect' or 'Inspect element' and you can see that the div element with v-show still exist, it is only the CSS display property that is set to 'none', and the div with v-if is destroyed.

```
const app = Vue.createApp({
  data() {
    return {
      showDiv: false
    }
  }
})
```

v-for Directive

- Attribute, **refer to the array inside the Vue instance**, and let Vue take care of the rest
- The elements created with v-for will **automatically update when the array changes**
- List ordering example: (v-for_list.html)

```
<ol>
  <li v-for="x in manyFoods">{{ x }}</li>
</ol>
```



```
1. Burrito
2. Salad
3. Cake
4. Soup
5. Fish
6. Pizza
7. Rice
```

v-for Directive

- The 'v-for' directive is used to create images based on the 'manyFoods' array in the Vue instance

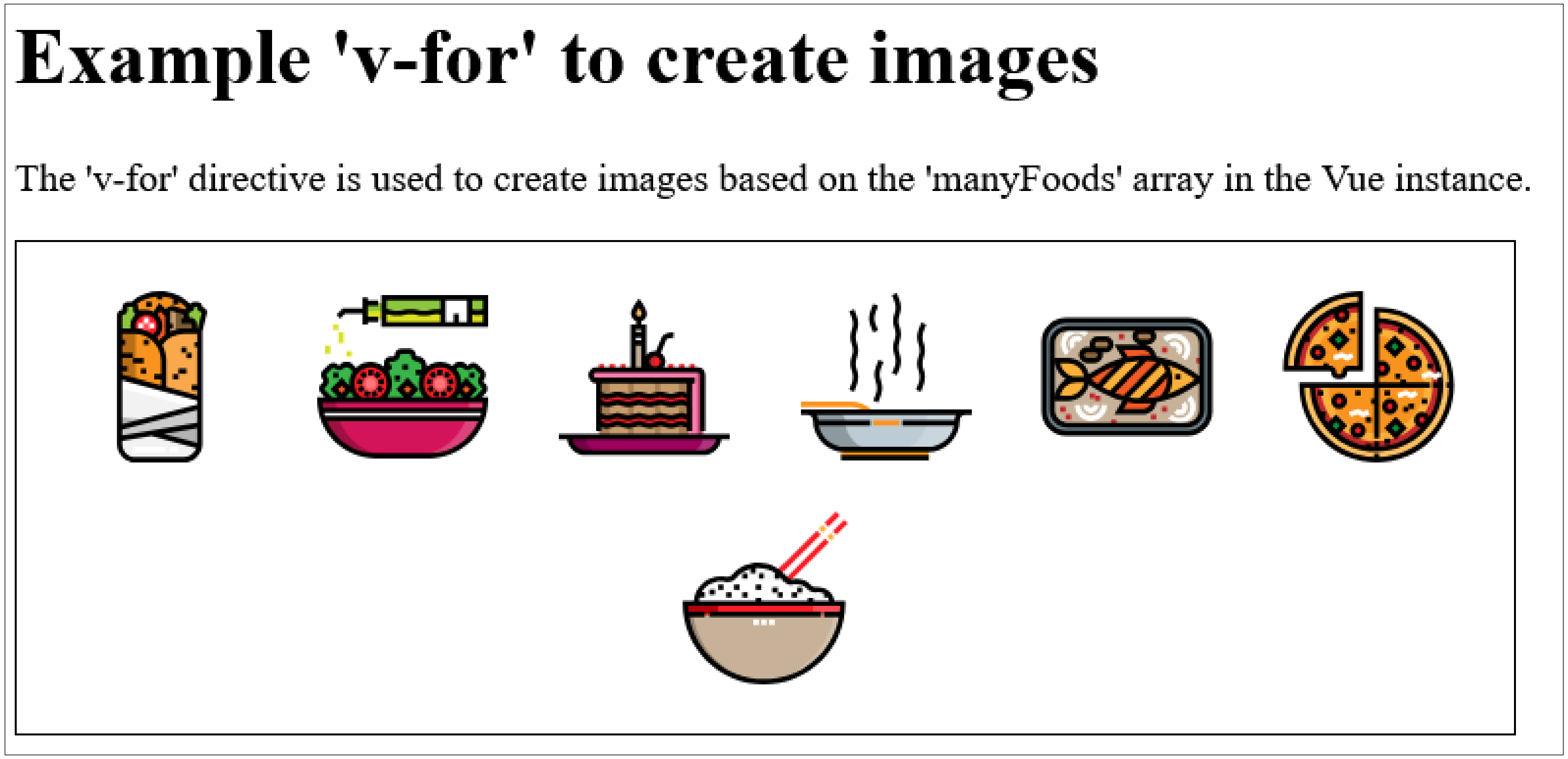
```
<div>
  <img v-for="x in manyFoods" v-bind:src="x">
</div>
```

```
const app = Vue.createApp({
 data() {
  return {
    manyFoods: [
      'img_burrito.svg',
      'img_salad.svg',
      'img_cake.svg',
      'img_soup.svg',
      'img_fish.svg',
      'img_pizza.svg',
      'img_rice.svg'
    ]
   }
  }
})
```

v-for Directive

- The 'v-for' directive is used to create images based on the 'manyFoods' array in the Vue instance

V-for_image.html


Example 'v-for' to create images

The 'v-for' directive is used to create images based on the 'manyFoods' array in the Vue instance.

v-for Directive

- The 'v-for' directive is used to create images and text based on the 'manyFoods' array in the Vue instance

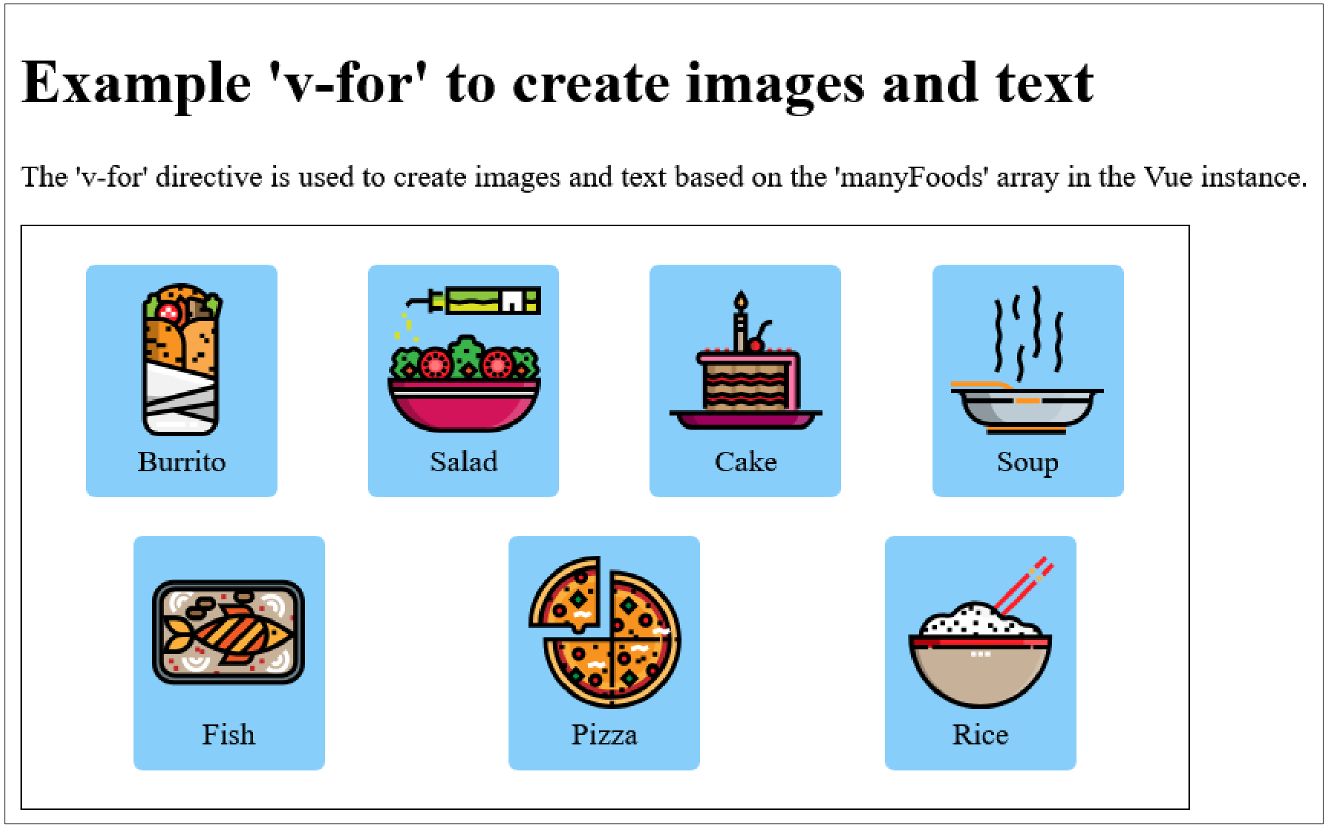
```
<div>
  <figure v-for="x in manyFoods">
    <img v-bind:src="x.url">
    <figcaption>{{ x.name }}</figcaption>
  </figure>
</div>
```

```
const app = Vue.createApp({
 data() {
  return {
    manyFoods: [
      {name: 'Burrito', url: 'img_burrito.svg'},
      {name: 'Salad', url: 'img_salad.svg'},
      {name: 'Cake', url: 'img_cake.svg'},
      {name: 'Soup', url: 'img_soup.svg'},
      {name: 'Fish', url: 'img_fish.svg'},
      {name: 'Pizza', url: 'img_pizza.svg'},
      {name: 'Rice', url: 'img_rice.svg'}
    ]
  }
 }
})
```

v-for Directive

- The 'v-for' directive is used to create images and text based on the 'manyFoods' array in the Vue instance

V-for_image_text.html



Example 'v-for' to create images and text

The 'v-for' directive is used to create images and text based on the 'manyFoods' array in the Vue instance.

Burrito

Salad

Cake

Soup

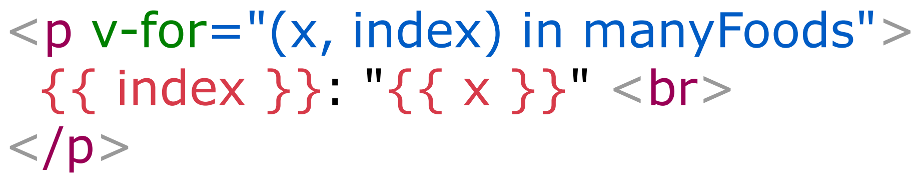
Fish

Pizza

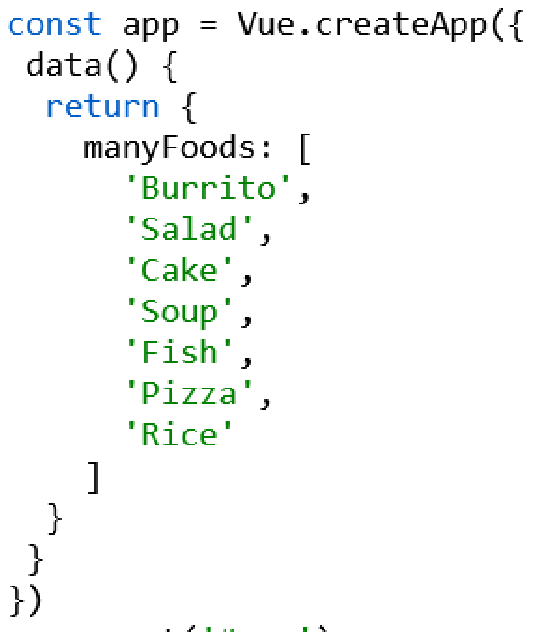
Rice

v-for Directive

- Show index number and food name of elements in the 'manyFoods' array in the Vue instance



```html
<p v-for="(x, index) in manyFoods">
 {{ index }}: "{{ x }}" <br>
</p>
```

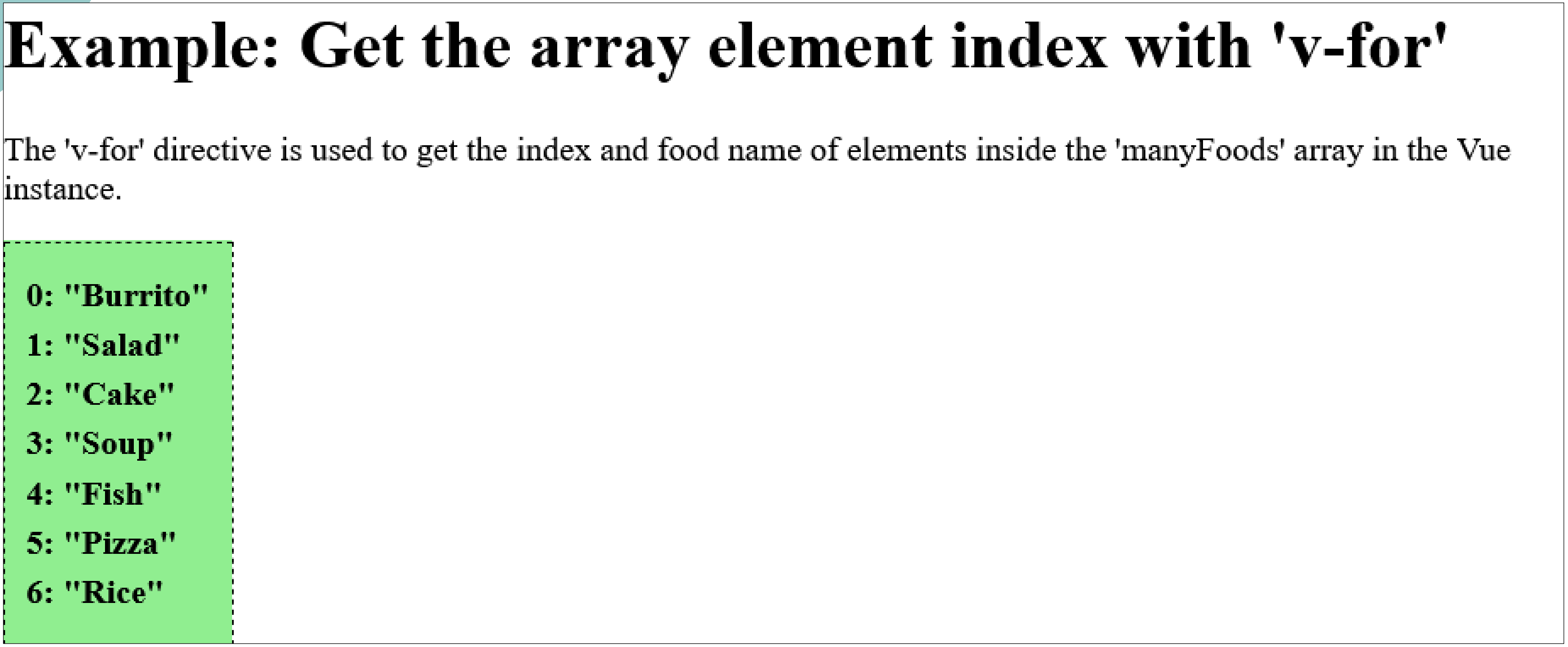


```javascript
const app = Vue.createApp({
 data() {
  return {
    manyFoods: [
      'Burrito',
      'Salad',
      'Cake',
      'Soup',
      'Fish',
      'Pizza',
      'Rice'
    ]
  }
 }
})
```

v-for Directive

- Show index number and food name of elements in the 'manyFoods' array in the Vue instance

V_for_array-element.html



Example: Get the array element index with 'v-for'

The 'v-for' directive is used to get the index and food name of elements inside the 'manyFoods' array in the Vue instance.

```
0: "Burrito"
1: "Salad"
2: "Cake"
3: "Soup"
4: "Fish"
5: "Pizza"
6: "Rice"
```

v-for Directive

- Show both the array element index number, and text from the objects in the 'manyFoods' array

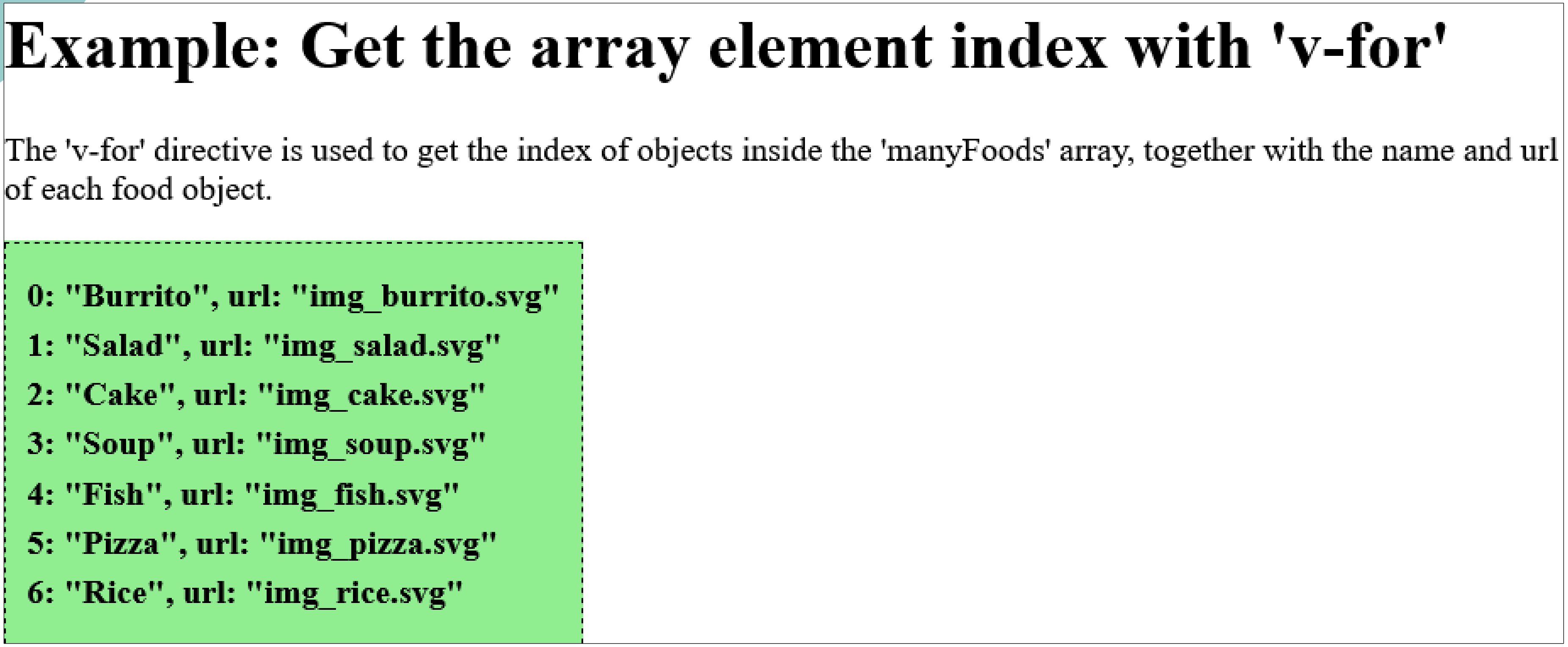
```
<p v-for="(x, index) in manyFoods">
 {{ index }}: "{{ x.name }}", url: "{{ x.url }}" <br>
</p>
```

```
const app = Vue.createApp({
 data() {
  return {
    manyFoods: [
      {name: 'Burrito', url: 'img_burrito.svg'},
      {name: 'Salad', url: 'img_salad.svg'},
      {name: 'Cake', url: 'img_cake.svg'},
      {name: 'Soup', url: 'img_soup.svg'},
      {name: 'Fish', url: 'img_fish.svg'},
      {name: 'Pizza', url: 'img_pizza.svg'},
      {name: 'Rice', url: 'img_rice.svg'}
    ]
  }
 }
})
```

v-for Directive

- Show both the array element index number, and text from the objects in the 'manyFoods' array

V-for_element-index.html



Example: Get the array element index with 'v-for'

The 'v-for' directive is used to get the index of objects inside the 'manyFoods' array, together with the name and url of each food object.

```
0: "Burrito", url: "img_burrito.svg"
1: "Salad", url: "img_salad.svg"
2: "Cake", url: "img_cake.svg"
3: "Soup", url: "img_soup.svg"
4: "Fish", url: "img_fish.svg"
5: "Pizza", url: "img_pizza.svg"
6: "Rice", url: "img_rice.svg"
```

Events

- Event handling in Vue is done **with the v-on directive**, so that we **can make something happen when** for example a button is clicked
- Event handling is when HTML elements are set up to run a certain code when a certain event happens
- Events in Vue are easy to use and will make **our page truly responsive**
- Vue **methods are code that can be set up to run when an event happens**
- With v-on modifiers you **can describe in more detail how to react to an event**

```
<p v-on:click="changeColor">Click me</p>
```

Events

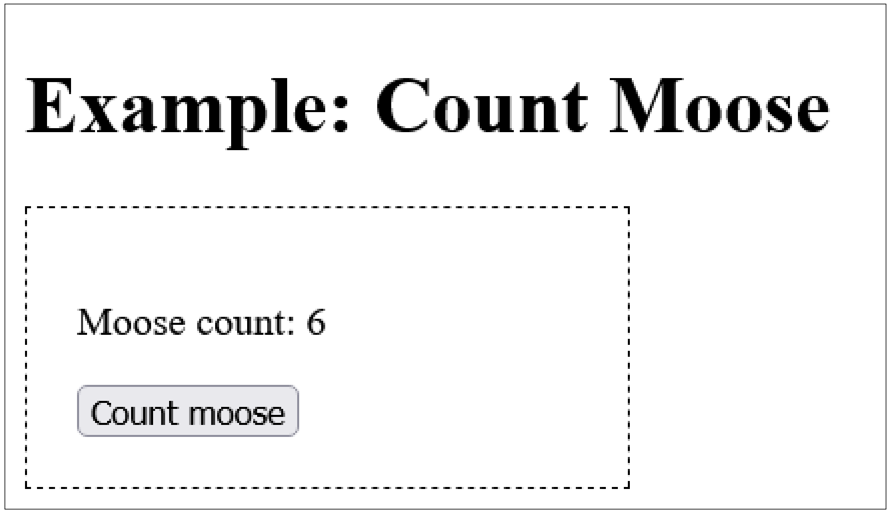
V-on_click.html

```
<div id="app">
  <p>{{ "Moose count: " + count }}</p>
  <button v-on:click="count++">Count moose</button>
</div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        count: 0
      }
    }
  })
 app.mount('#app')
</script>
```

- A benefit that comes with Vue is that the number of moose in the **<p> tag is updated automatically**

Events

V-on_click.html



Example: Count Moose

Moose count: 6

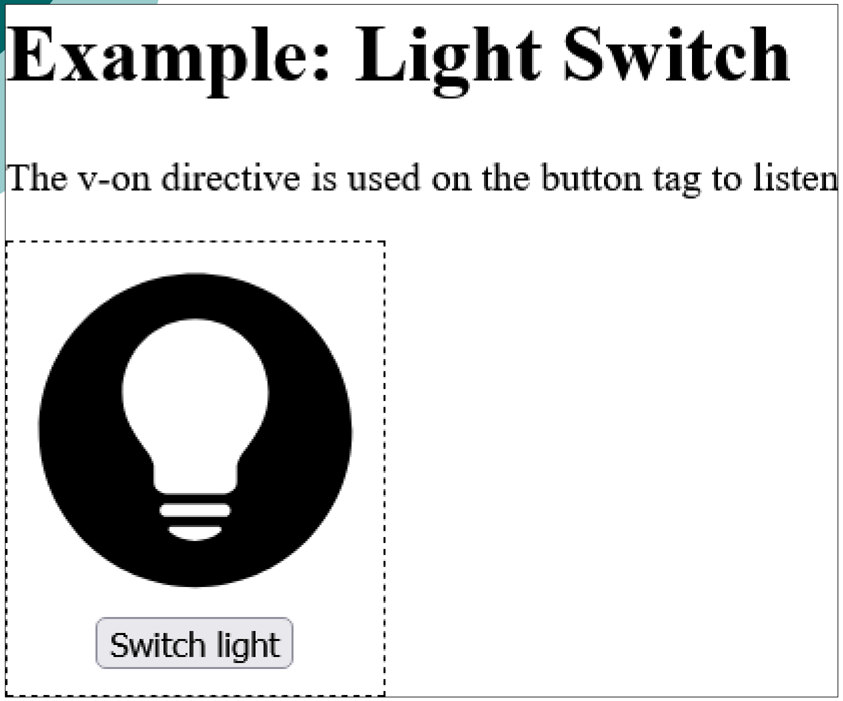
Events – V-on:click

- The v-on directive allows us to **perform actions based on specified events**
- Use **v-on:click to perform action when the element is clicked**

```
<div id="app">
  <div id="lightDiv">
    <div v-show="lightOn"></div>
    <img src="img_lightBulb.svg">
  </div>
  <button v-on:click="lightOn = !lightOn">Switch light</button>
</div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        lightOn: false
      }
    }
  })
  app.mount('#app')
</script>
```

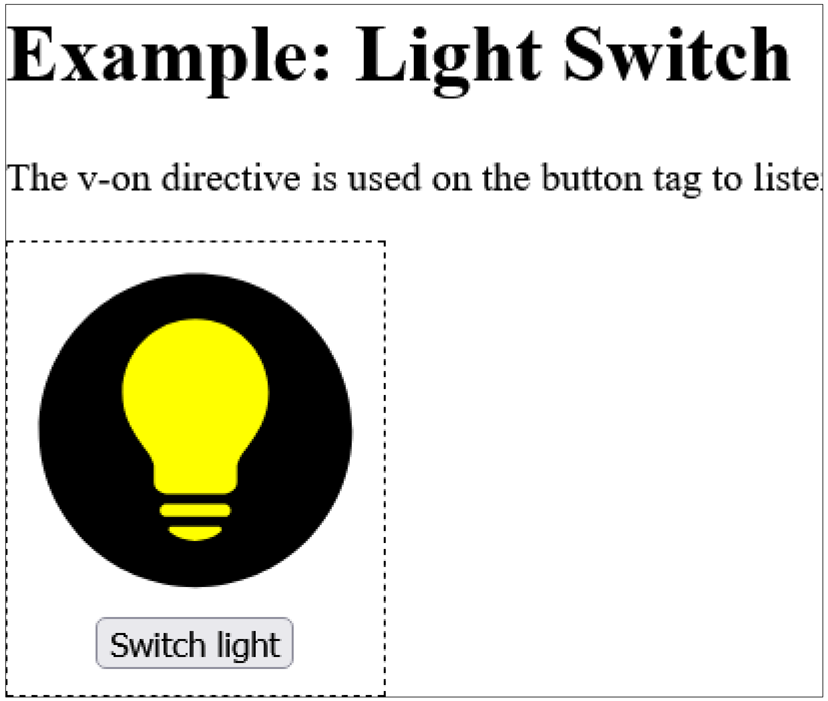
Events – V-on:click

V-on_click2.html



Example: Light Switch

The v-on directive is used on the button tag to listen



Example: Light Switch

The v-on directive is used on the button tag to liste

Events – V-on:input

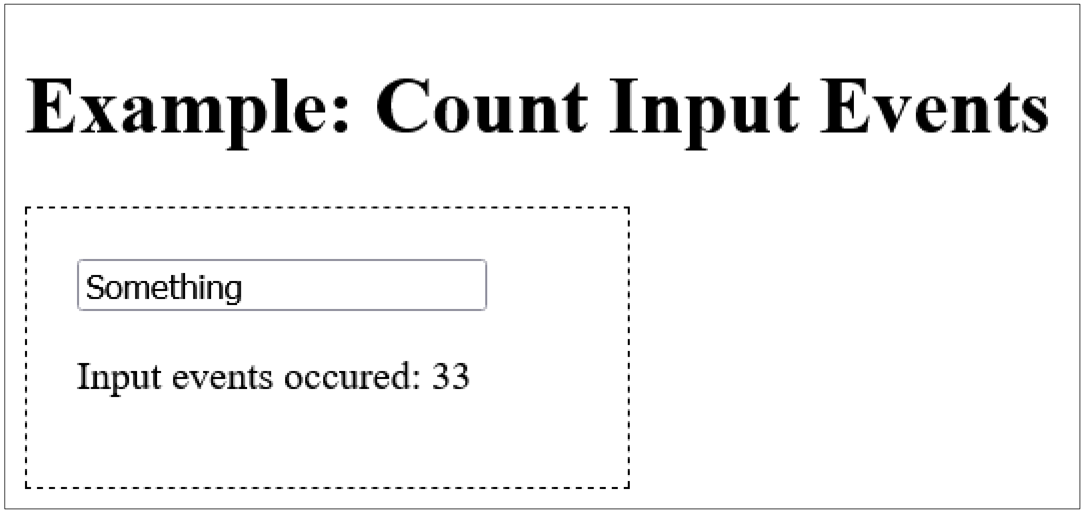
- Use **v-on:input** to perform action **when the element gets an input**
  - like a keystroke inside a text field

```
<div id="app">
  <input v-on:input="inpCount++">
  <p>{{ 'Input events occured: ' + inpCount }}</p>
</div>

<script
src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    data() {
      return {
        inpCount: 0
      }
    }
  })
  app.mount('#app')
</script>
```

Events – V-on:input

- Use **v-on:input** to perform action **when the element gets an input**
  - like a keystroke inside a text field

V-on_input.html



Example: Count Input Events

Something

Input events occured: 33

Events – V-on:mousemove

- Use **v-on:mousemove to perform action when the mouse pointer moves over an element**

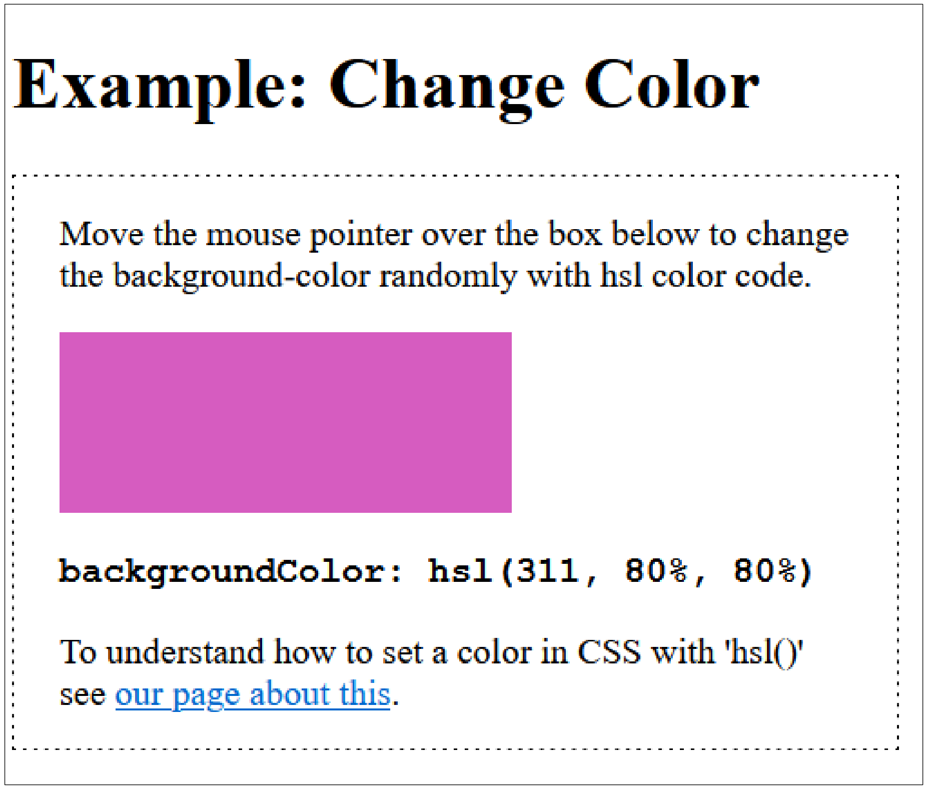
```
const app =
Vue.createApp({
   data() {
     return {
       colorVal: 50
     }
   }
  })
  app.mount('#app')
```

```
<div v-on:mousemove="colorVal=Math.floor(Math.random()*360)"
     v-bind:style="{backgroundColor:'hsl('+colorVal+',80%,80%)'}">
  </div>
```

Events – V-on:mousemove

- Use **v-on:mousemove to perform action when the mouse pointer moves over an element**

V-on_mousemove.html



Example: Change Color

Move the mouse pointer over the box below to change the background-color randomly with hsl color code.

```
backgroundColor: hsl(311, 80%, 80%)
```

To understand how to set a color in CSS with 'hsl()' see our page about this.

Events – V-on and v-for

- Can be also use the v-on directive **inside a v-for loop**
- The items of the array are available for each iteration inside the v-on value

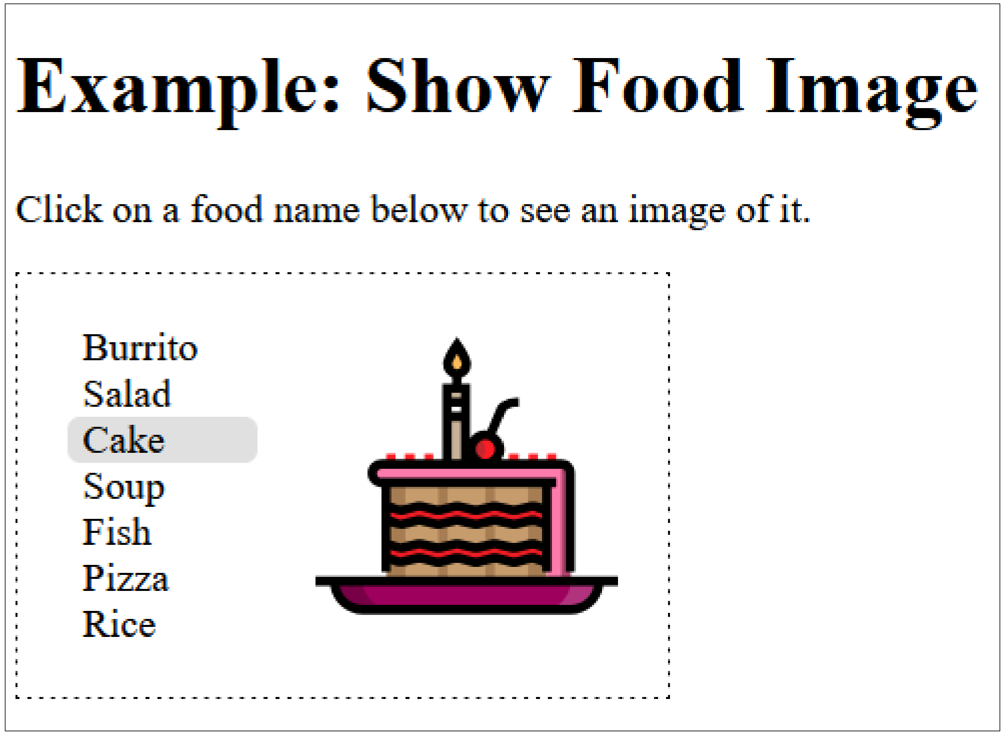
```
<div id="app">
  <img v-bind:src="imgUrl">
  <ol>
    <li v-for="food in manyFoods" v-on:click="imgUrl=food.url">
      {{ food.name }}
    </li>
  </ol>
</div>
```

```
const app = Vue.createApp({
    data() {
      return {
        imgUrl: 'img_salad.svg',
        manyFoods: [
          {name: 'Burrito', url: 'img_burrito.svg'},
          {name: 'Salad', url: 'img_salad.svg'},
          {name: 'Cake', url: 'img_cake.svg'},
          {name: 'Soup', url: 'img_soup.svg'}
        ]
      }
    }
  })
```

Events – V-on and v-for

- Can be also use the v-on directive **inside a v-for loop**
- The items of the array are available for each iteration inside the v-on value

V-on_v-for.html



Example: Show Food Image

Click on a food name below to see an image of it.

Burrito
Salad
Cake
Soup
Fish
Pizza
Rice

Methods

- Vue methods are **functions that belong to the Vue instance under the 'methods' property**
- Vue **methods are great to use with event handling** (v-on) to do more complex things
- Vue methods can also be used to do other things than event handling

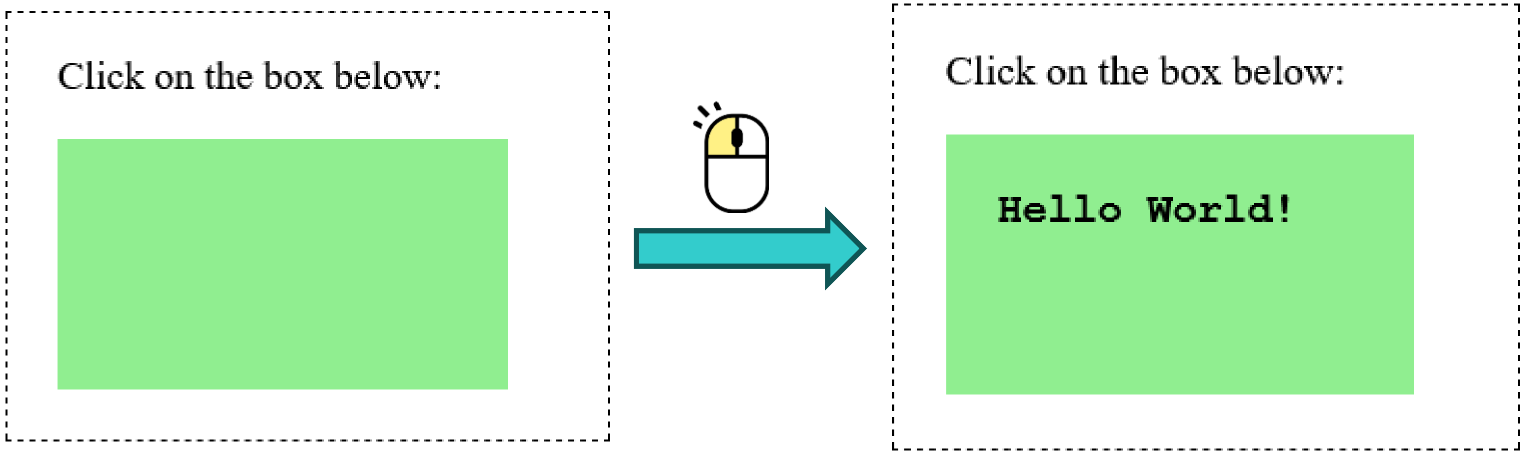
```
const app = Vue.createApp({
  data() {
    return {
      text: ''
    }
  },
  methods: {
    writeText() {
      this.text = 'Hello World!'
    }
  }
})
```

```
<div v-on:click="writeText"></div>
```

Methods

- The **v-on directive is used on the <div> element to listen to the 'click' event**
- **When the 'click' event occurs the 'writeText' method is called** and the text is changed

Methods_click.html


Click on the box below:

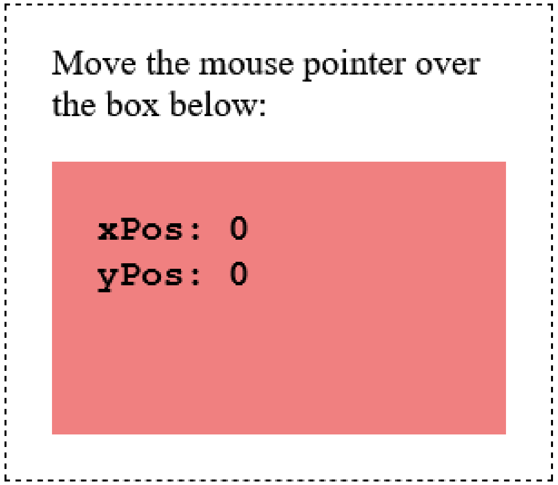
Click on the box below:

Hello World!

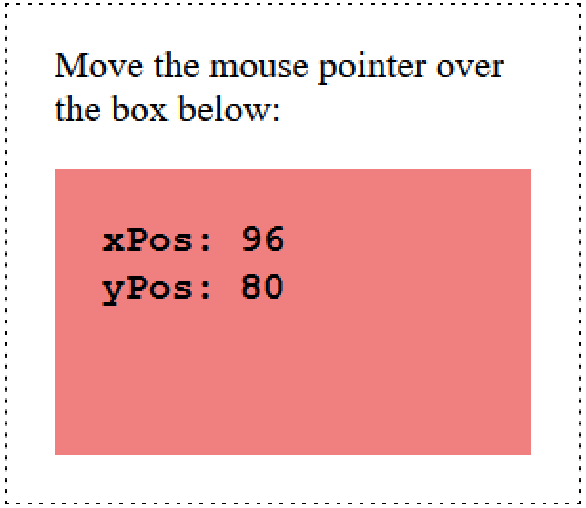
Methods

- The **v-on directive is used on the <div> element to listen to the 'mousemove' event**
- **When the 'mousemove' event occurs the 'mousePos' method is called** and the event object is sent with the method by default so we can get the mouse pointer position

Methods_mouse_pointer.html


Move the mouse pointer over the box below:

```
xPos: 0
yPos: 0
```

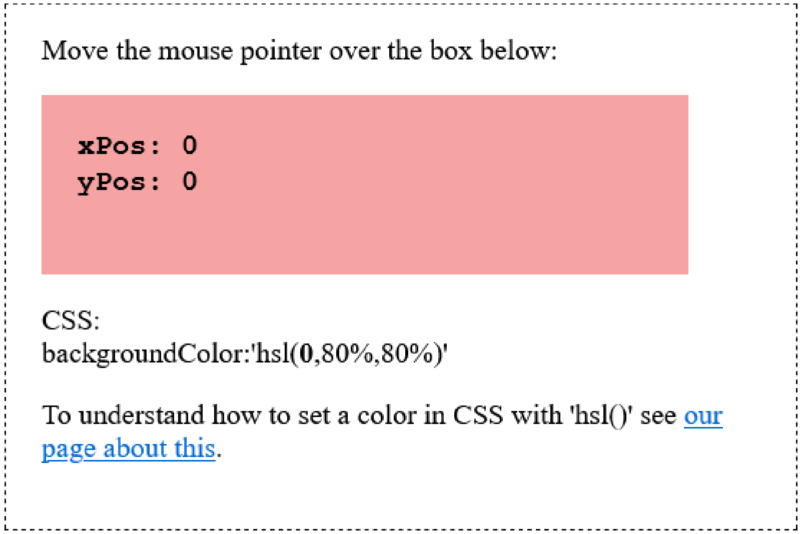

Move the mouse pointer over the box below:

```
xPos: 96
yPos: 80
```

Methods

- The difference here from the example above is that the **background color** is bound to 'xPos' with v-bind so that hsl 'hue' value is set equal to 'xPos'

Methods_mouse_pointer_color.html

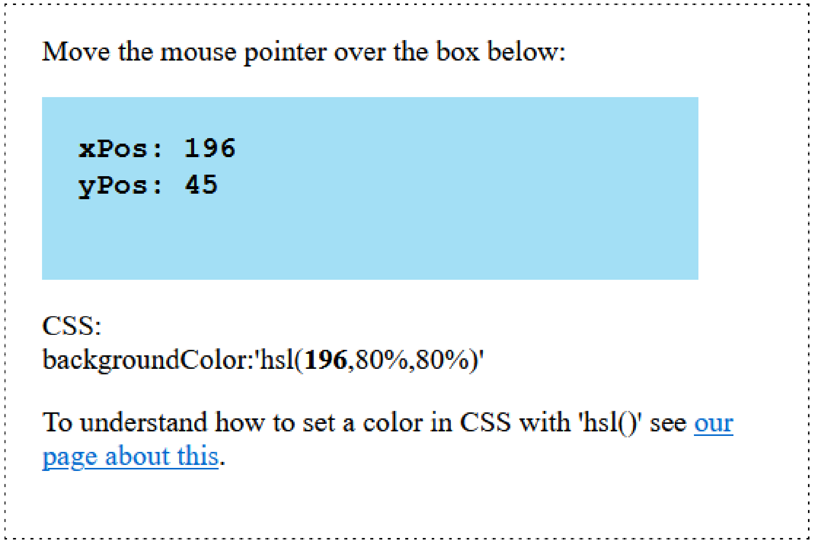


Move the mouse pointer over the box below:

```
xPos: 0
yPos: 0
```

CSS:
backgroundColor:'hsl(**0**,80%,80%)'

To understand how to set a color in CSS with 'hsl()' see our page about this.



Move the mouse pointer over the box below:

```
xPos: 196
yPos: 45
```

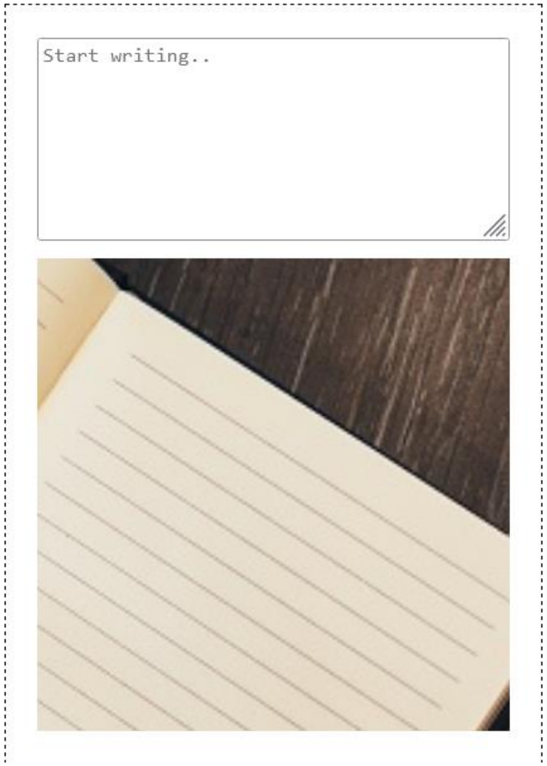
CSS:
backgroundColor:'hsl(**196**,80%,80%)'

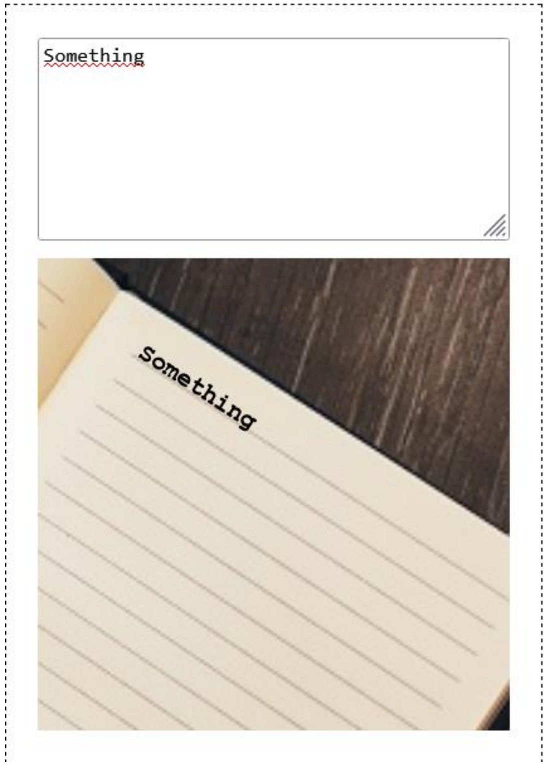
To understand how to set a color in CSS with 'hsl()' see our page about this.

Methods

- The **v-on directive is used on the <textarea> tag to listen to the 'input' event** which occurs whenever there is a **change in the text inside the textarea element**

Methods_textarea.html





Methods

- **Sometimes we want to pass an argument with the method when an event occurs**
- Add buttons to count sightings '+1' and '+5', and a '-1' button in case we have counted too many

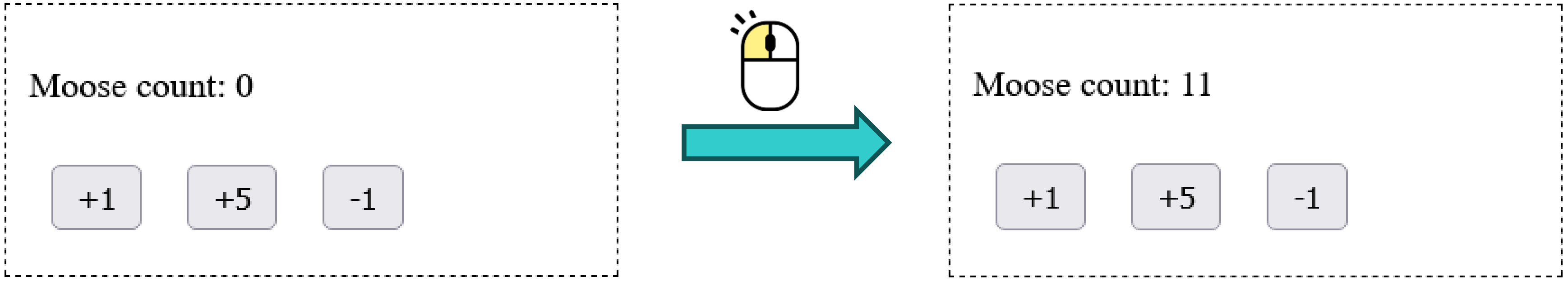
```
<button v-on:click="addMoose(5)">+5</button>
```

```
methods: {
  addMoose(number) {
    this.count = this.count + number
  }
}
```

Methods

- Sometimes we want to pass an argument with the method when an event occurs
- **Add buttons to count sightings '+1' and '+5', and a '-1' button in case we have counted too many**

Methods_mouse_click.html



Methods

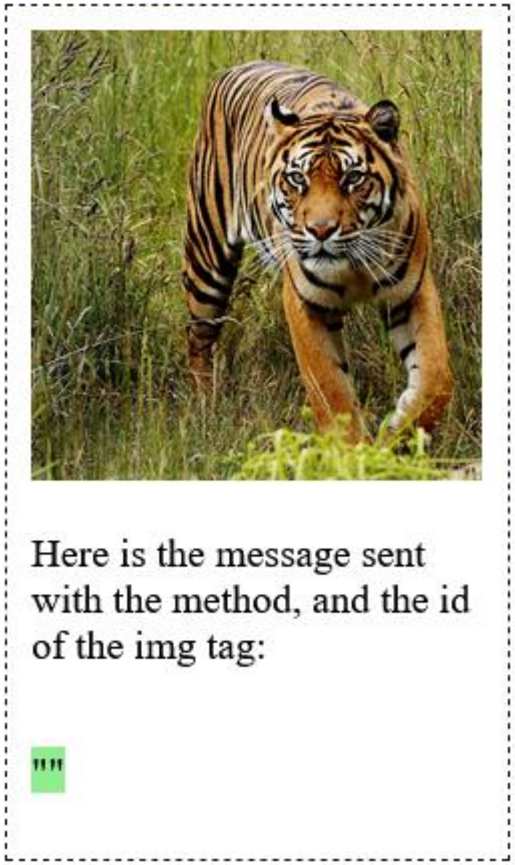
- If we want to pass both the event object and another argument, **there is a reserved name '$event' we can use where the method is called**:

```
<button v-on:click="addAnimal($event, 5)">+5</button>
```

```
methods: {
  addAnimal(e, number) {
    if(e.target.parentElement.id==="tigers"){
      this.tigers = this.tigers + number
    }
  }
}
```

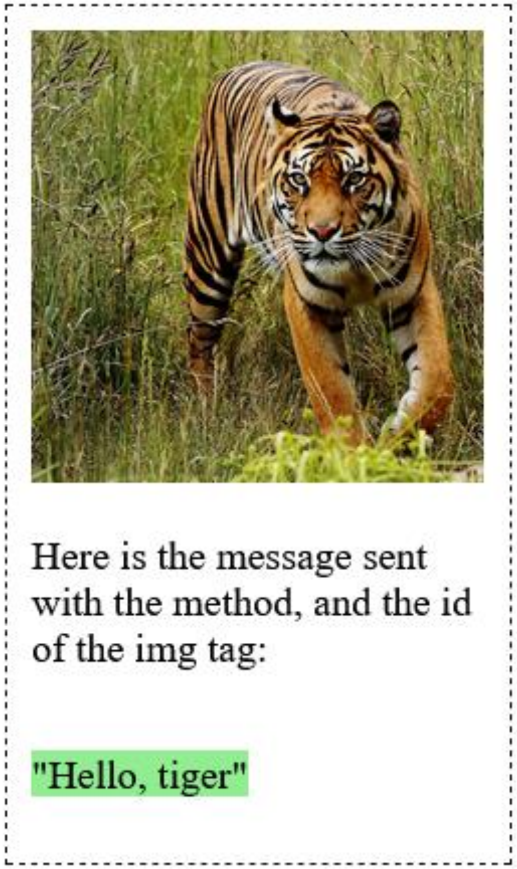
Methods

Methods_event_object.html



Here is the message sent with the method, and the id of the img tag:

""



Here is the message sent with the method, and the id of the img tag:

"Hello, tiger"

Event Modifiers

- **Event modifiers modify how events trigger the running of methods** and help us handle events in a more efficient and straightforward way
- Event modifiers are **used together with the Vue v-on directive**, to for example:
  - **Prevent** the default submit behavior of HTML forms
    - **v-on:submit.prevent**
  - Make sure that an event can **only run once** after the page is loaded
    - **v-on:click.once**
  - Specify **what keyboard key to use** as an event to run a method
    - **v-on:keyup.enter**

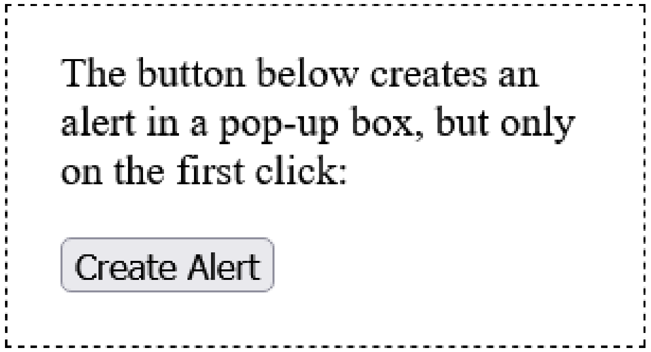
Event Modifiers

- **How To Modify The v-on Directive**

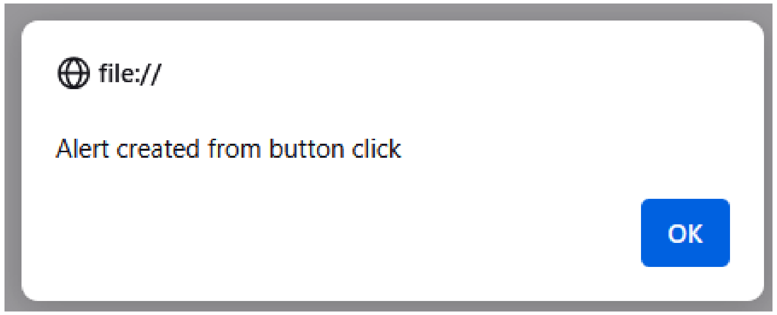
```
<button v-on:click="createAlert">Create alert</button>
```

```
<button v-on:click.once="createAlert">Create alert</button>
```

Event_modifiers_once.html



The button below creates an alert in a pop-up box, but only on the first click:



Alert created from button click

Event Modifiers

- **Keyboard Key Event Modifiers**
  - We have three different keyboard event types **keydown, keypress, and keyup**
  - With each key event type, we can specify exactly what key to listen to after a key event occurs. We have .space, .enter, .w and .up to name a few

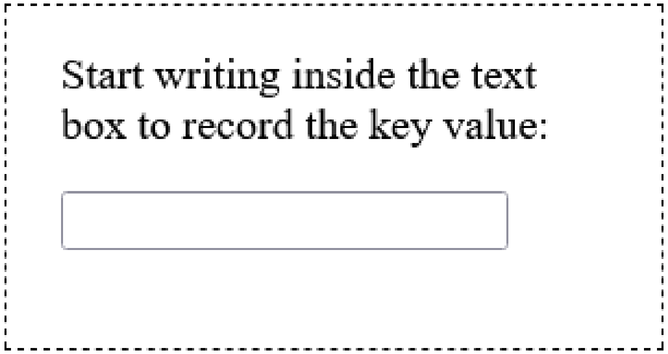
```
<input v-on:keydown="getKey">
<p> {{ keyValue }} </p>
```

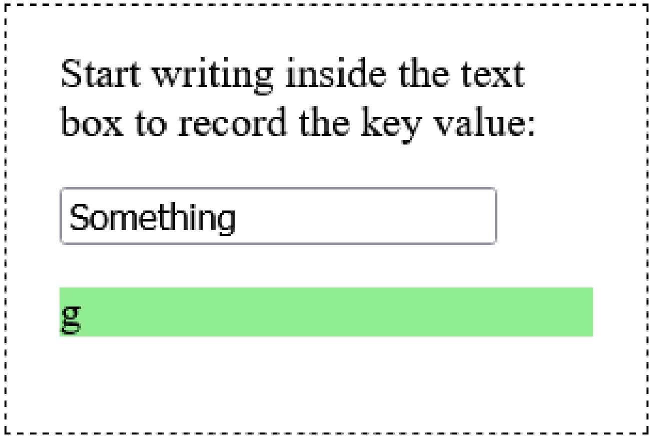
```
data() {
  return {
    keyValue = ''
  }
},
methods: {
  getKey(evt) {
    this.keyValue = evt.key
    console.log(evt.key)
  }
}
```

Event Modifiers

- **Keyboard Key Event Modifiers**
  - The keydown keyboard event triggers the 'getKey' method, and the value 'key' from the event object is written to the console and to the web page

Event_modifiers_key.html


Start writing inside the text box to record the key value:


Start writing inside the text box to record the key value:

Something

g

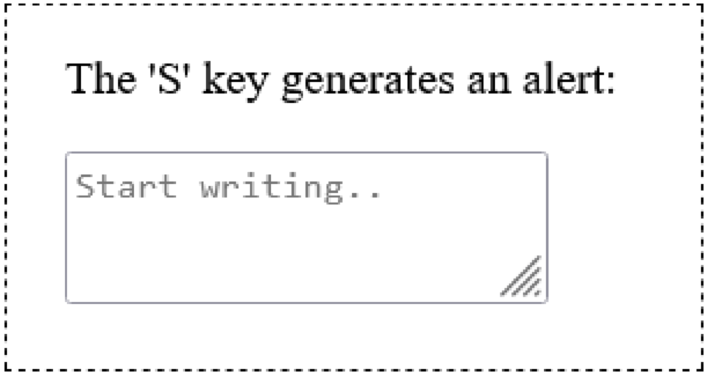
Event Modifiers

Key Modifier	Details
.[Vue key alias]	•The most common keys have their own aliases in Vue: .enter •.tab •.delete •.esc •.space •.up •.down •.left •.right
.[letter]	Specify the letter that comes when you press the key. As an example: use the .s key modifier to listen to the 'S' key.
.[system modifier key]	.alt, .ctrl, .shift or .meta. These keys can be used in combination with other keys, or in combination with mouse clicks.

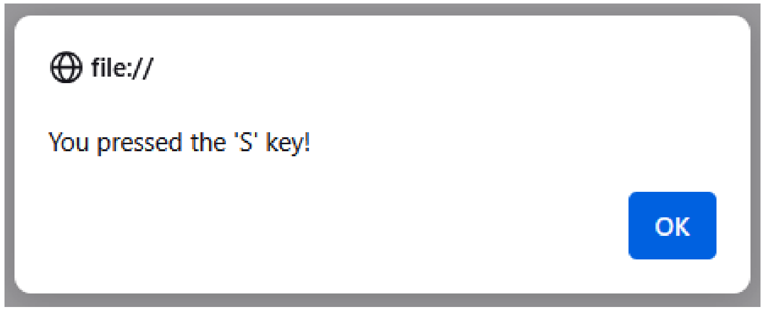
Event Modifiers

- **Keyboard Key Event Modifiers**
  - Use the .s modifier to create an alert when the user writes an 's' inside the <textarea> tag

Event_modifiers_key_alert.html


The 'S' key generates an alert:

Start writing..


file://

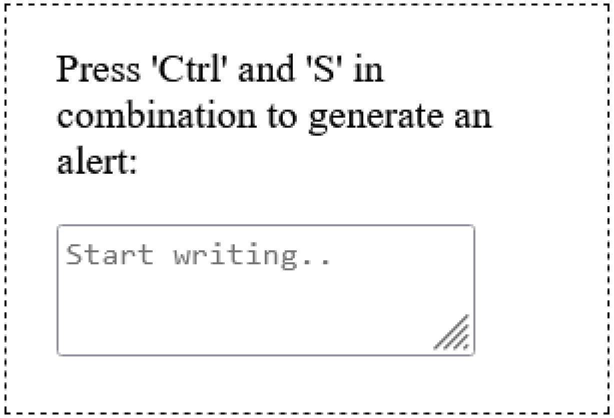
You pressed the 'S' key!

Event Modifiers

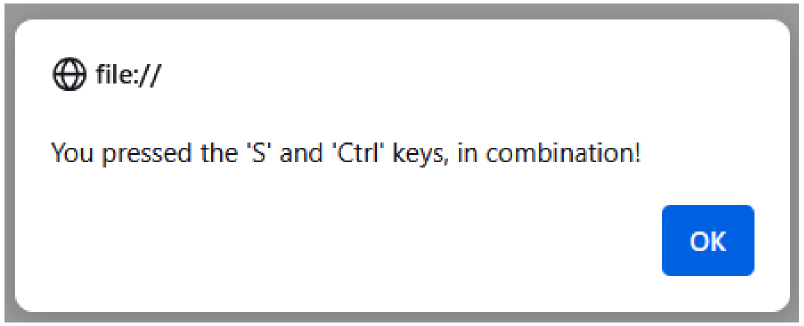
- **Keyboard Key Event Modifiers**
  - Use the .s and .ctrl modifiers in combination to create an alert when **'s' and 'ctrl' are pressed simultaneously** inside the <textarea> tag

```
<textarea v-on:keydown.ctrl.s="createAlert"></textarea>

createAlert() {
        alert("You pressed the 'S' and 'Ctrl' keys, in combination!")
      }
```

Event_modifiers_key_alert_simultaneously.html



Press 'Ctrl' and 'S' in combination to generate an alert:



file://

You pressed the 'S' and 'Ctrl' keys, in combination!

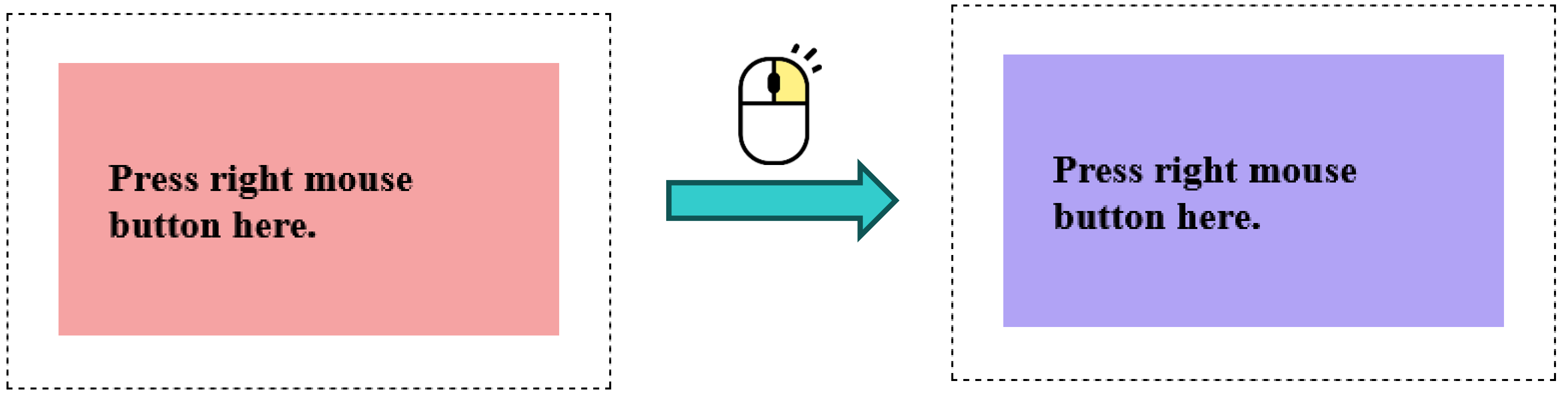
Event Modifiers

Mouse Button Modifiers

- To react on a mouse click, we can write v-on:click, but to specify which mouse button that was clicked, **we can use .left, .center or .right modifiers**

```
<div v-on:click.right="changeColor"
     v-bind:style="{backgroundColor:'hsl('+bgColor+',80%,80%)'}">
  <p>Press right mouse button here.</p>
</div>
```

Event_mouse_right_click.html



Press right mouse button here.

Press right mouse button here.

Event Modifiers

- **Mouse Button Modifiers**
  - Hold the 'shift' keyboard key and press **left mouse button** on the <img> tag **to change image**

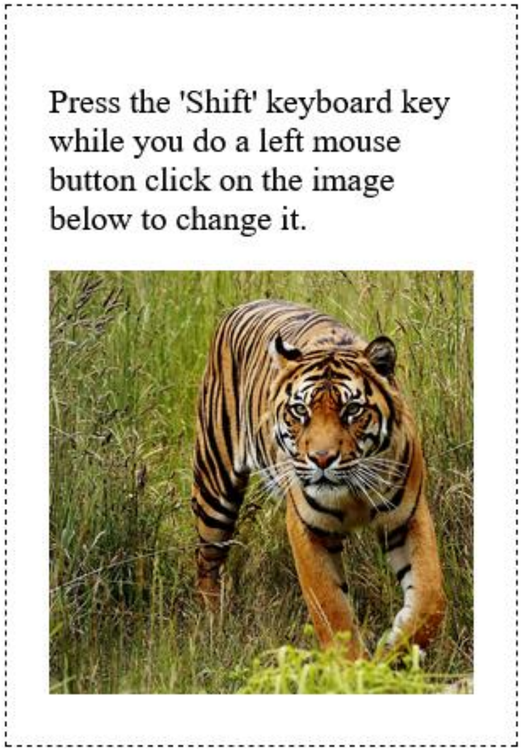
```
<img v-on:click.left.shift="changeImg" v-bind:src="imgUrl">
```

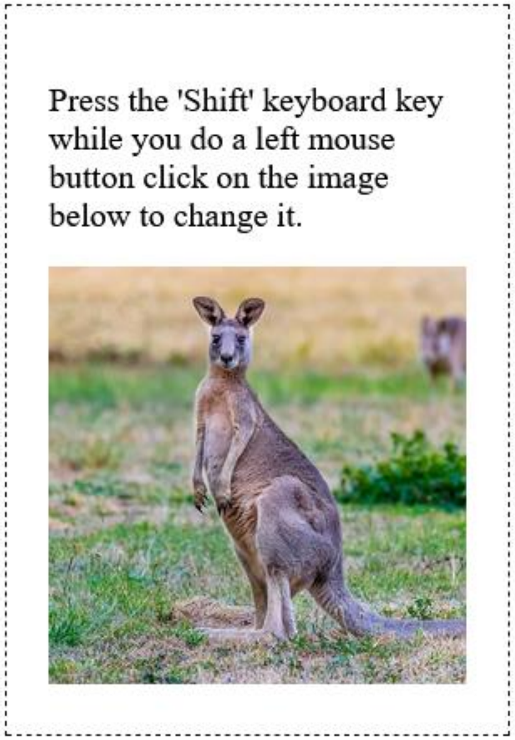
```
changeImg() {
      this.imgUrlIndex++
      if(this.imgUrlIndex>=3){
        this.imgUrlIndex=0
      }
      this.imgUrl = this.images[this.imgUrlIndex]
    }
  }
```

Event Modifiers

- **Mouse Button Modifiers**
  - Hold the 'shift' keyboard key and press **left mouse button** on the <img> tag **to change image**

Event_mouse_click_image.html


Press the 'Shift' keyboard key while you do a left mouse button click on the image below to change it.


Press the 'Shift' keyboard key while you do a left mouse button click on the image below to change it.

Forms

- Vue gives us an **easy way to** improve the user experience with forms by **adding extra functionality like responsiveness** and form validation
- Vue **uses the v-model** directive when handling forms
  - v-model **updates the Vue instance data** when the HTML input change
  - v-model also **updates the HTML input** when the Vue instance data changes

Forms

- **How Vue can be used to create a form:**
  - 1. Add standard HTML form elements

```
<form>
  <p>Add item</p>
  <p>Item name: <input type="text" required></p>
  <p>How many: <input type="number"></p>
  <button type="submit">Add item</button>
</form>
```

Forms

- **How Vue can be used to create a form:**
  - 2. Create the Vue instance with the current item name

```
<form>
  <p>Add item</p>
  <p>Item name: <input type="text" required v-model="itemName"></p>
  <p>How many: <input type="number" v-model="itemNumber"></p>
  <button type="submit">Add item</button>
 </form>
```

```
const app = Vue.createApp({
  data() {
    return {
      itemName: null,
      itemNumber: null,
      shoppingList: [
        { name: 'Tomatoes', number: 5 }
      ]
    }
  }
 })
```

Forms

- **How Vue can be used to create a form:**
  - 3. Call the method to add the given item, and prevent the default browser refresh on submit

```
<form v-on:submit.prevent="addItem">
```

  - 4. Create the method that adds the item and clears the form:

```
methods: {
  addItem() {
    let item = {
      name: this.itemName,
      number: this.itemNumber
      }
    this.shoppingList.push(item);
    this.itemName = null
    this.itemNumber = null
  }
}
```

Forms

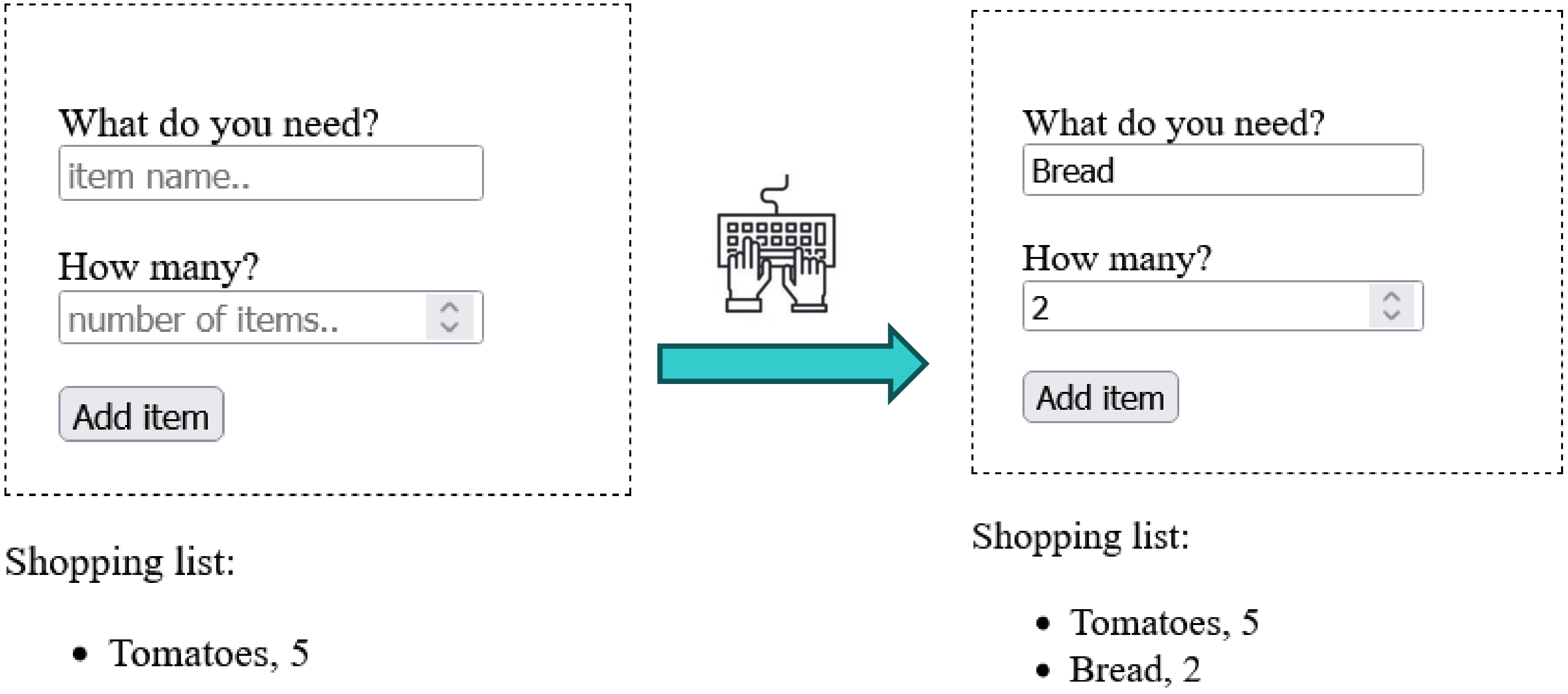
- **How Vue can be used to create a form:**
  - 5. Use v-for to show an automatically updated shopping list below the form

```
<p>Shopping list:</p>
<ul>
  <li v-for="item in shoppingList">{{item.name}}, {{item.number}}</li>
</ul>
```

Forms

- **How Vue can be used to create a form:**

Froms.html



v-model Directive

- V-model **creates a link between the input element value attribute and a data value** in the Vue instance
- When change an input, the data updates and when the data changes, the input updates as well
  - called: two-way binding

- **Two-way binding**
  - the form input elements update the Vue data instance and a change in the Vue instance data updates the inputs

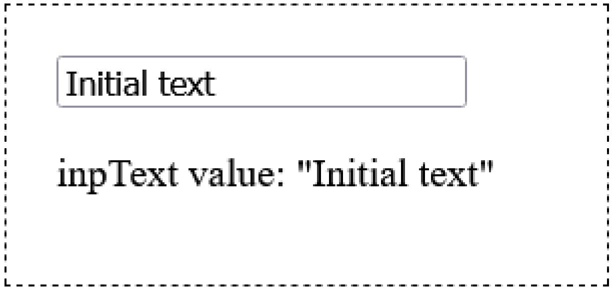
v-model Directive

```html
<input type="text" v-model="inpText">
 <p> {{ inpText }} </p>
```

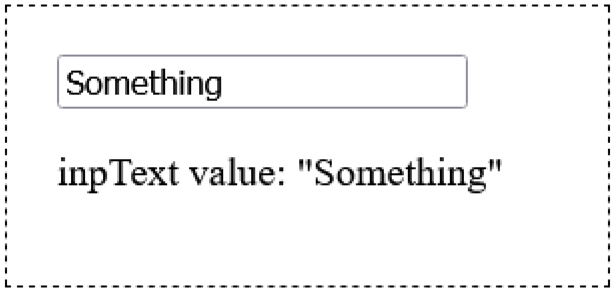
```js
const app = Vue.createApp({
    data() {
      return {
        inpText: 'Initial text'
      }
    }
  })
```

- Try changing the input field and see how the Vue property value updates
  - **v-bind:value to update the input element** from the Vue instance data
  - **v-on:input to update the Vue instance data** from the input

Two-way-binding.html


Initial text

inpText value: "Initial text"


Something

inpText value: "Something"

v-model Directive

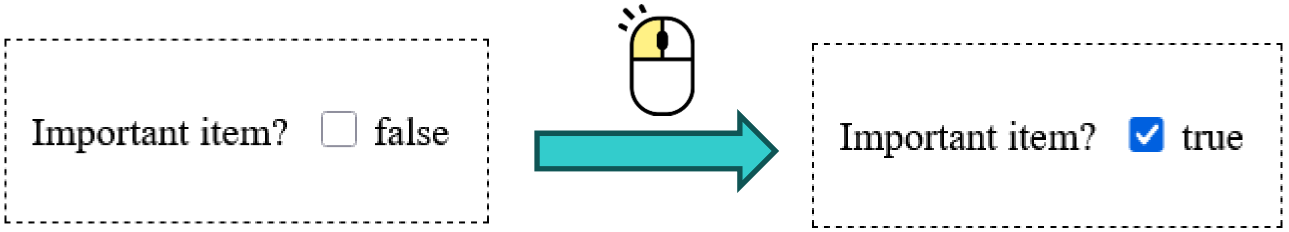
- **A Dynamic Checkbox**
  - use v-model to add this dynamic checkbox and text to improve user interaction
  - We need:
    - **a boolean value** in the Vue instance data property called 'important'
    - **a checkbox** where the user can check if the item is important
    - **a dynamic feedback** text so that the user can see if the item is important

```
<input type="checkbox" v-model="important">
     {{ important }}
```

```
data() {
    return {
     important: false
    }
  }
```

v-model Directive

Dynamic_checkbox.html

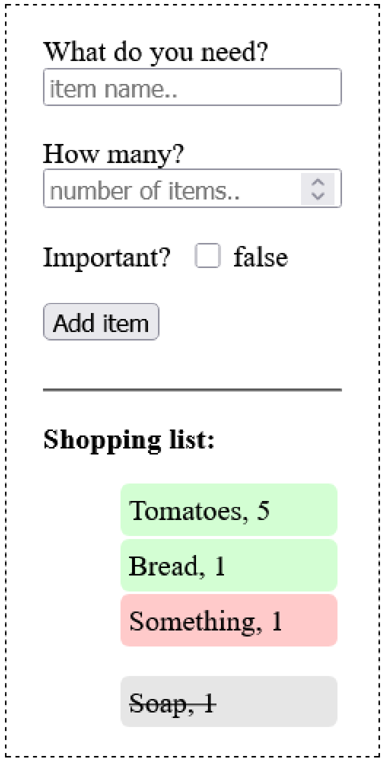

Important item? ☐ false

Important item? ☑ true

v-model Directive

- **Shopping list**
  - the **list items to react on click**
  - to **change the status of the clicked item** to 'found', and use this to visually move the item away and strike it through with CSS

Shopping_list.html


What do you need?
item name..

How many?
number of items..

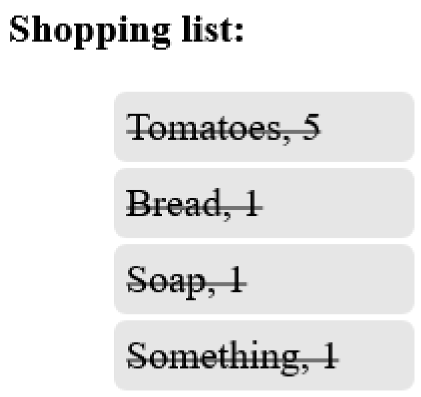
Important? ☐ false

Add item

Shopping list:

Tomatoes, 5
Bread, 1
Something, 1

~~Soap, 1~~


Shopping list:

~~Tomatoes, 5~~
~~Bread, 1~~
~~Soap, 1~~
~~Something, 1~~

v-model Directive

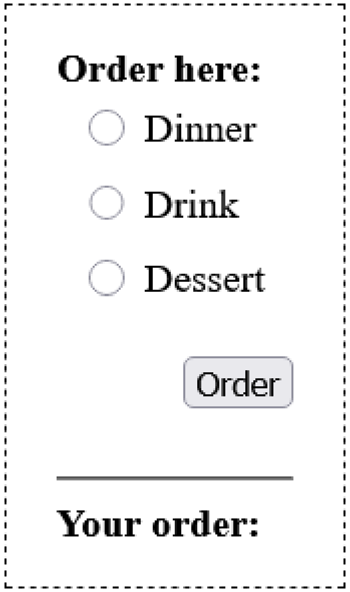
- **Restaurant Order**
  - **A form**, with relevant input tags and 'Order' button
  - **Radio-buttons** to select **'Dinner'**, **'Drink'** or **'Dessert'**
  - **After category is chosen, a dropdown menu appears with all the items in that category**
  - **When an item is chosen you see an image of it**, you can choose how many and add it to the order
  - The **form is reset when the item is added** to the order

Shopping_list.html

v-model Directive

- **Restaurant Order**

Restaurant_order.html



Order here:

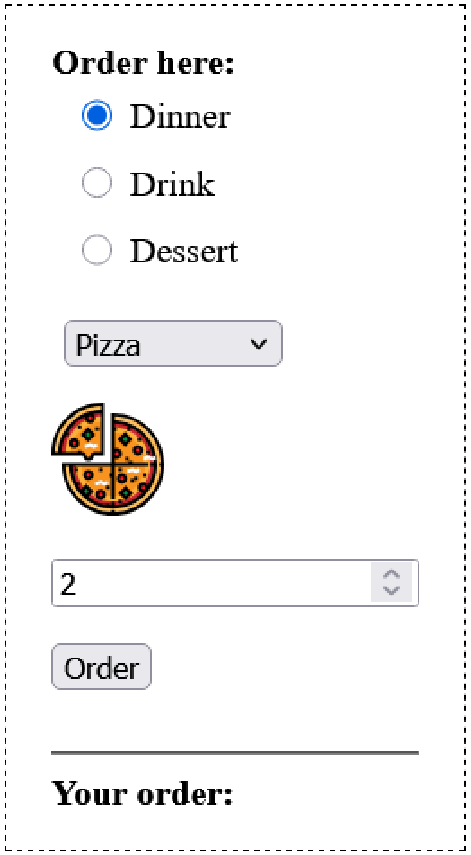
( ) Dinner

( ) Drink

( ) Dessert

Order

Your order:



Order here:

(●) Dinner

( ) Drink

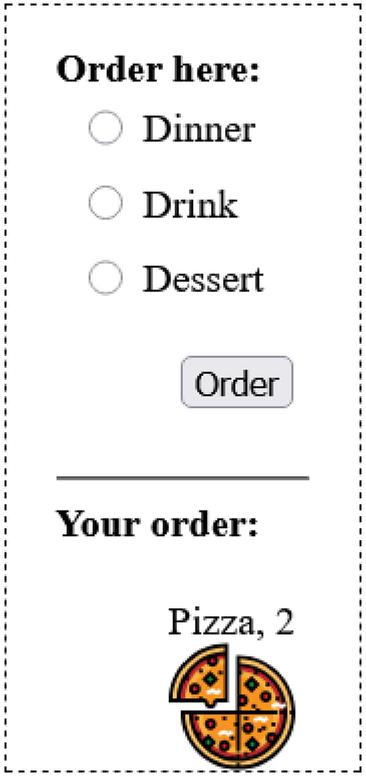
( ) Dessert

Pizza

2

Order

Your order:



Order here:

( ) Dinner

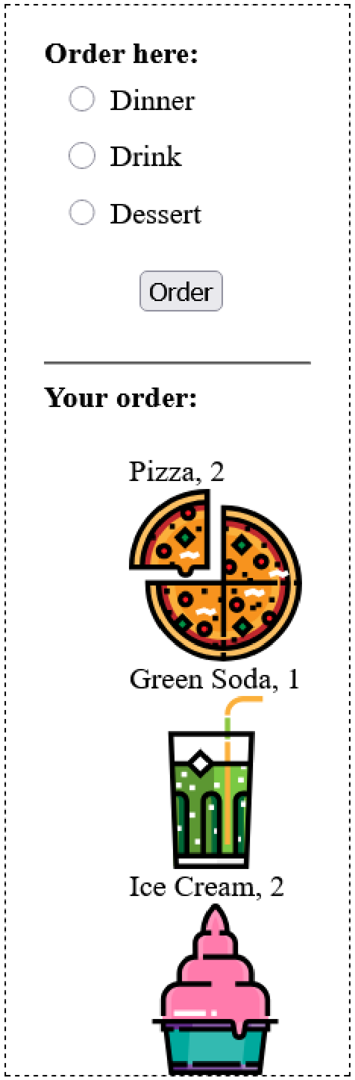
( ) Drink

( ) Dessert

Order

Your order:

Pizza, 2



Order here:

( ) Dinner

( ) Drink

( ) Dessert

Order

Your order:

Pizza, 2

Green Soda, 1

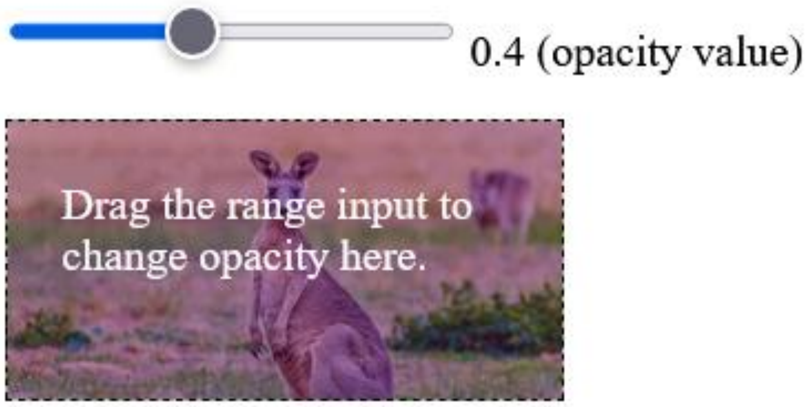
Ice Cream, 2

CSS Binding

- **Inline Styling**
  - Can be use **v-bind:style** to do in-line styling in Vue

```
<input type="range" v-model="opacityVal">
<div v-bind:style="{ backgroundColor: 'rgba(155,20,20,'+opacityVal+')' }">
  Drag the range input above to change opacity here.
</div>
```

CSS_inline.html


0.4 (opacity value)

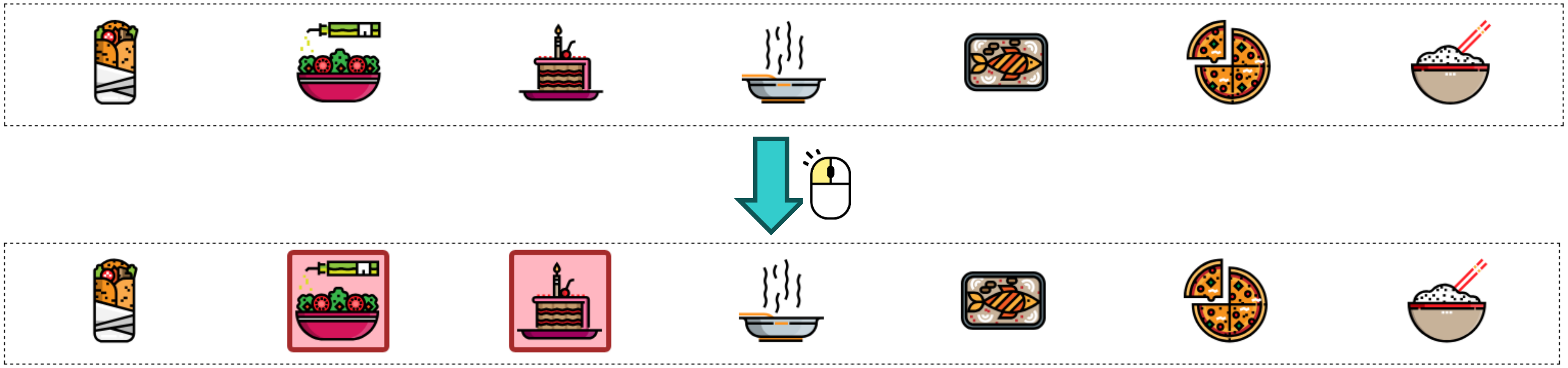
Drag the range input to change opacity here.

CSS Binding

- **Assign a Class**
  - Can be use **v-bind:class** to assign a class to an HTML tag

```
<div v-for="(img, index) in images">
 <img v-bind:src="img.url"
     v-on:click="select(index)"
     v-bind:class="{ selClass: img.sel }">
</div>
```

CSS_assign_a_class.html

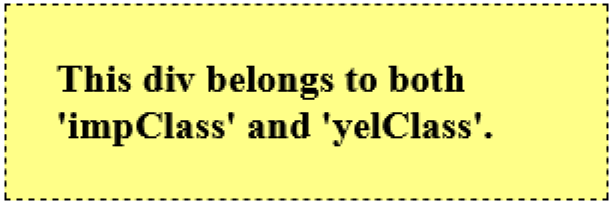


CSS Binding

- **Merges 'class' And 'v-bind:class'**
  - Can be use **v-bind:class** to assign a class to an HTML tag
  - **A <div> element belongs to two classes: 'impClass' and 'yelClass’**
  - The 'important' class is set the normal way with the class attribute, and 'yellow' class is set with v-bind:class

```
<div class="impClass" v-bind:class="{yelClass: isYellow}">
 This div belongs to both 'impClass' and 'yelClass'.
</div>
```

CSS_merge_class.html



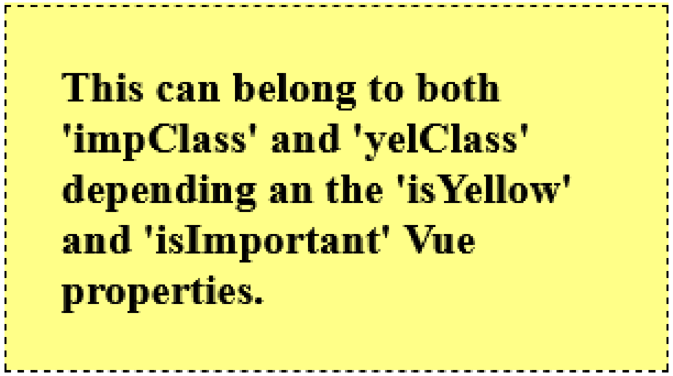
This div belongs to both 'impClass' and 'yelClass'.

CSS Binding

- **Assign More Than One Class With 'v-bind:class'**
  - A <div> element can belong to both 'impClass' and 'yelClass' classes, depending on the boolean Vue data properties 'isYellow' and 'isImportant'.

```
<div v-bind:class="{yelClass: isYellow, impClass: isImportant}">
  This tag can belong to both the 'impClass' and 'yelClass' classes.
</div>
```

CSS_more_than_one_class.html



This can belong to both 'impClass' and 'yelClass' depending an the 'isYellow' and 'isImportant' Vue properties.

Computed Properties

- **Computed properties are**
  - like **data properties**, except they depend on other properties
  - are written like **methods**, but they do not accept any input arguments
  - are **updated automatically when a dependency changes**, while methods are called on when something happens, like with event handling for example
  - are **used when outputting something** that depends on something else

Computed Properties

- **Computed properties syntax**

```
const app = Vue.createApp({
  data() {
    ...
  },
  computed: {
    ...
  },
  methods: {
    ...
  }
})
```

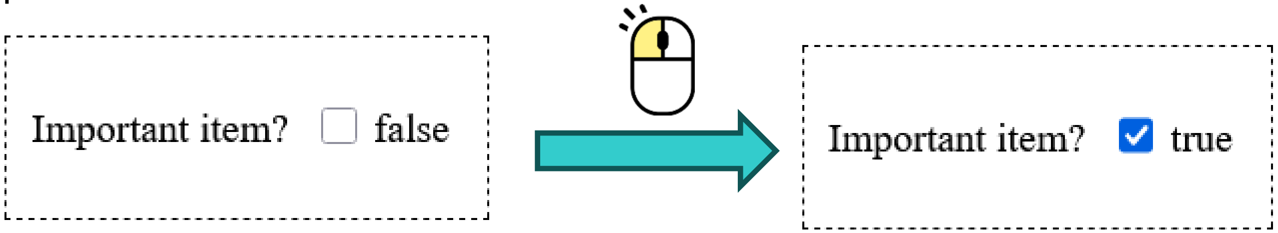
Computed Properties

- **Computed properties example**

```
<input type="checkbox" v-model="chbxVal"> {{ chbxVal }}
```

```
data() {
  return {
    chbxVal: false
  }
}
```

Computed_properties.html



Important item? ☐ false

Important item? ☑ true

Watchers

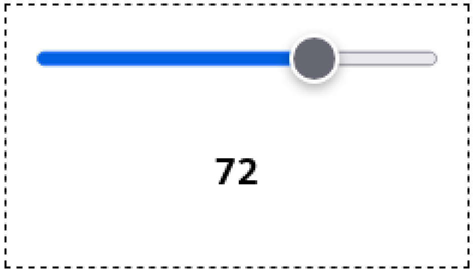
- A watcher is **a method that watches a data property** with the same name
- It **runs every time the data property value changes**
- **Use it if a certain data property value requires an action**

```
const app = Vue.createApp({
  data() {
    ...
  },
  watch: {
    ...
  },
  computed: {
    ...
  },
  methods: {
    ...
  }
})
```

Watchers

```
<input type="range" v-model="rangeVal">
<p>{{ rangeVal }}</p>


const app = Vue.createApp({
  data() {
    rangeVal: 70
  },
  watch: {
    rangeVal(val){
      if( val>20 && val<60) {
        if(val<40){
          this.rangeVal = 20;
        }
        else {
          this.rangeVal = 60;
        }
      }
    }
  }
})
```

Watcher.html


72

Watchers

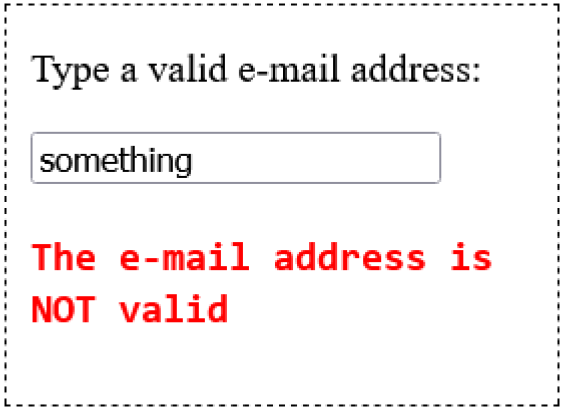
- The value from an <input> element is connected to a watcher
- If the value includes a '@' it is considered a valid e-mail address
- The user gets a feedback text to inform if the input is valid, invalid, or if it just got valid with the last keystroke

```
<input v-type="email" v-model="inpAddress">
<p v-bind:class="myClass">{{ feedbackText }}</p>
```

```
watch: {
    inpAddress(newVal,oldVal) {
      if( !newVal.includes('@') ) {
        this.feedbackText = 'The e-mail address is NOT valid';
        this.myClass = 'invalid';
      }
      else if( !oldVal.includes('@') && newVal.includes('@') ) {
        this.feedbackText = 'Perfect! You fixed it!';
        this.myClass = 'valid';
      }
      else {
        this.feedbackText = 'The e-mail address is valid :)';
      }
    }
  }
```

Watchers

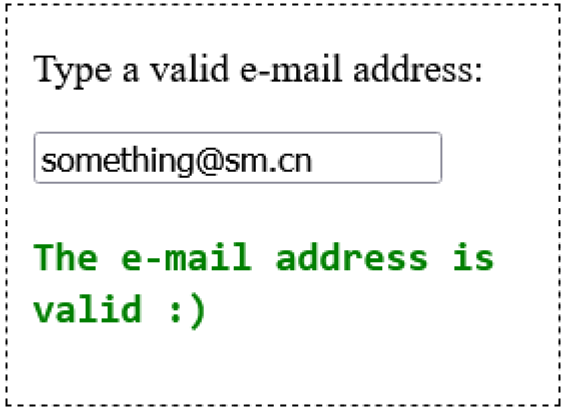
Watcher_email.html



Type a valid e-mail address:

something

The e-mail address is NOT valid



Type a valid e-mail address:

something@sm.cn

The e-mail address is valid :)

Templates

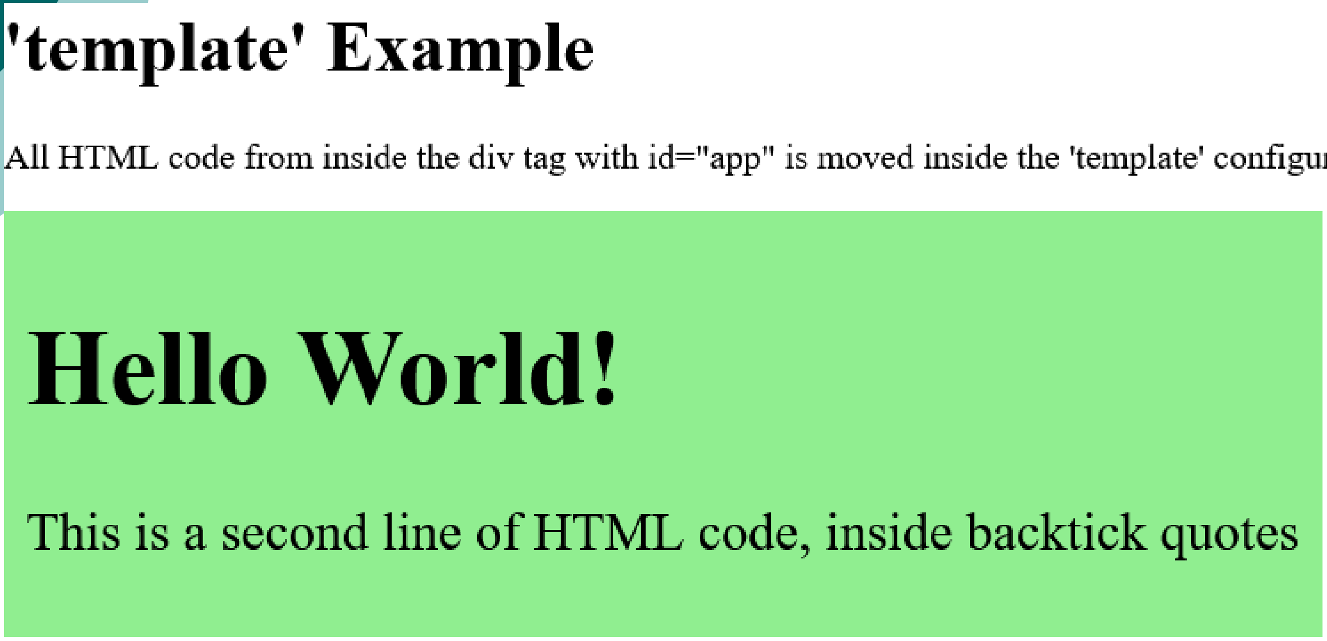
- **the HTML part of the a Vue application**
- The **<template> tag** will later be used in *.vue files to structure our code in a better way
- It is **possible to use template as a configuration** option in the Vue instance, and put the HTML code inside

Templates

```
<div id="app"></div>

<script
src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
  const app = Vue.createApp({
    template:
      `<h1>{{ message }}</h1>
      <p>This is a second line of HTML code, inside back tick
quotes</p>`,
    data() {
      return {
        message: "Hello World!"
      }
    }
  })
app.mount('#app')
</script>
```

Templates

Template.html


'template' Example

All HTML code from inside the div tag with id="app" is moved inside the 'template' configuration option, in backtick quotes.

Hello World!

This is a second line of HTML code, inside backtick quotes

Thank you for your attention!