

Web Applications Powered by Vue.js!

From Concept to Launch!

Introductions ...

• Peter Pavlovich

- Front-End Evangelist and Technology Addict
- CTO, Censinet, Inc. (Supply Chain Risk Management for the Healthcare Sector)
 - React/Vue/Rails/Python, Postgres/MongoDB/Redis, AWS/Heroku
- Principal Advisor, Embue, Inc. (Smart Apartment Management Systems)
 - Angular/Vue/Meteor, Node, Scala/Java, Postgres/MongoDB, AWS

Introductions ...

- ⦿ Peter Pavlovich
- ⦿ Entrepreneur, Open Source Contributor
and Community Member
- ⦿ Instructor, Mentor and Speaker.
- ⦿ <http://linkedin.com/in/peterpavlovich>
- ⦿ pavlovich@gmail.com
- ⦿ @ppavlovich

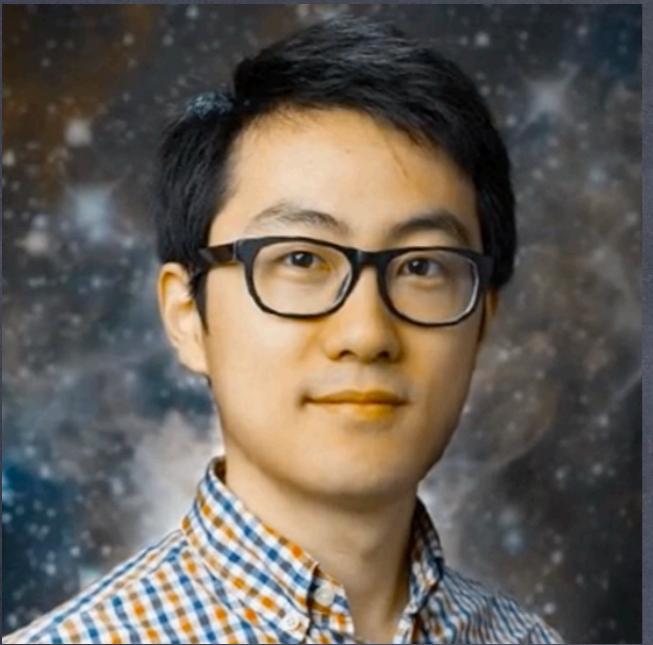
Plan for our session ...

- ⦿ Build and deploy a cool Task Manager App
- ⦿ Along the way we will:
 - ⦿ Explore all the major features of Vue.js
 - ⦿ Understand the add-on technologies
 - ⦿ Build a Web app
 - ⦿ Utilize Vuex, and Vuetify
 - ⦿ Witness amazingly easy application development
 - ⦿ Love how simple, powerful, and fun Vue.js is!

Workshop PDF

🔗 <https://app.box.com/v/vue-workshop-pdf>

What is Vue.js?



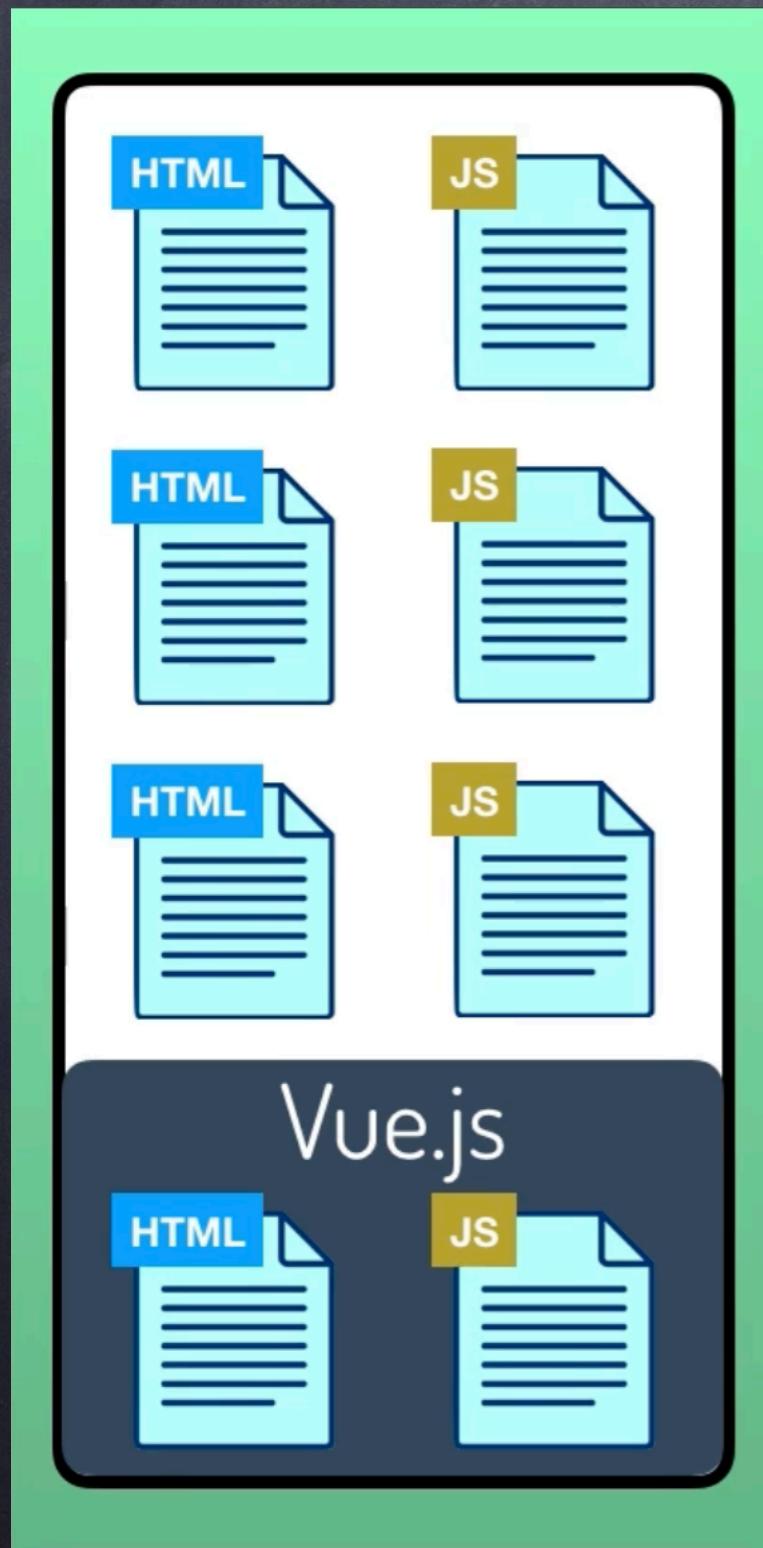
What is Vue.js

- A web application, front-end framework
- Based on work by Evan You, an ex-googler from the Angular world.
- "I figured, what if I could just extract the part that I really liked about Angular and build something really lightweight"

Why Vue.js?

- ⦿ It is:
 - ⦿ Approachable
 - ⦿ Versatile
 - ⦿ Performant
 - ⦿ Maintainable
 - ⦿ Testable

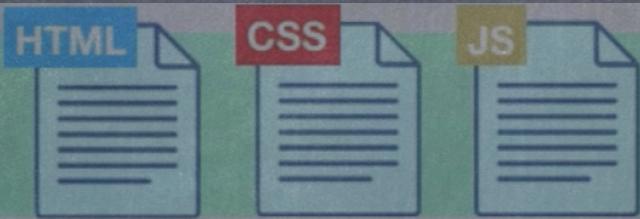
Why Vue.js?



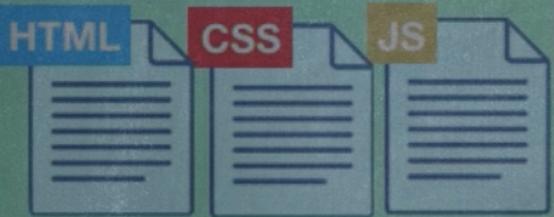
- ➊ Progressive
 - ➋ Convert your existing app over time
 - ➋ Coexists with:
 - ➋ Angular, React, Polymer, etc.
 - ➋ Java/Grails, .net, Ruby/Rails, Meteor

Why Vue.js?

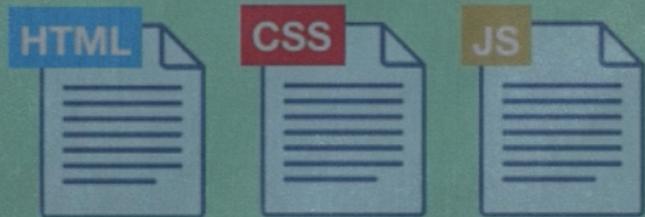
Header & Navigation



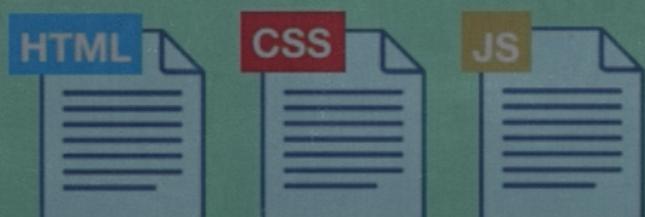
Product Image



Product Description



Similar Products



- Component Based

- Stateless,
functional
components for
performance

- Stateful
components for
flexibility/power

Why Vue.js?

- ⦿ Simultaneous support for:
 - ⦿ Multiple scripting languages
 - ⦿ ES5, ES6, ES7, coffeescript, typescript
 - ⦿ Multiple styling libraries
 - ⦿ Plain CSS, Less, Sass, Stylus
 - ⦿ Multiple templating languages
 - ⦿ HTML, Handlebars, JSX, Pug/Jade

Why Vue.js?

- ☞ Simultaneous support for:
 - ☞ Multiple component patterns:
 - ☞ Stand-alone mini-app ‘components’
 - ☞ Stateful (class based) components
 - ☞ Functional (stateless) components
 - ☞ Manually rendered components: Explicit DOM construction using JavaScript

Vue is Reactive!

- When the data for our app changes the UI will update to reflect those changes.

Reactive demo

- ➊ Build a simple Vue page and *see it react!*

Why yet another
framework?

Vue vs React

- ⦿ Both: Use a virtual DOM
- ⦿ Both: Components are reactive & composable
- ⦿ Both: Provide stateful & stateless components
- ⦿ Both: Well factored:
 - ⦿ Core library
 - ⦿ Routing, global state in separate libraries
- ⦿ Both: Perform well with little differences

Vue vs React

- ⦿ When React re-renders a component, it re-renders the entire component subtree.
- ⦿ Need 'shouldComponentUpdate' a lot = extra work for developers
- ⦿ Vue.js does this for you automatically without extra coding for the developer.

Vue vs React

- In React, everything is just javascript
- JSX templates everywhere
- CSS is also 'javascript'
- Vue.js supports JSX when you need that but also supports plain HTML templates when you don't. Also supports Jade/Pug, handlebars, ...

Vue vs React

- ⦿ Scaling
- ⦿ Big learning curve with React:
 - ⦿ JSX, ES6+, build systems, etc.
- ⦿ Vue is simpler, more flexible, less opinionated.

Vue vs Angular

- ⦿ AngularJS
 - ⦿ Vue.js is simpler, component based, faster.
- ⦿ Angular (2+)
 - ⦿ Vue.js is smaller (50% less), less opinionated/flexible and does not mandate the use of Typescript.
 - ⦿ Vue.js has a much lower learning curve.

Vue.js Ecosystem

- Lots of great plugins:
 - **Vuex**: Redux for vue
 - **Vue-Router**: Full featured Router for vue
 - **Nuxt**: developer framework for vue
 - **vuefire/vuexFire**: Firebase bindings for vue
 - **Vuemotion**: Easy state transitions for vue

Vue.js Ecosystem

- ⦿ More great plugins:
 - ⦿ **axios**: easy REST calls integrated with vue
 - ⦿ **Vuelidate**: model validation framework
 - ⦿ **Vue-apollo**: Apollo/GraphQL integration
 - ⦿ **Vue-i18n**: Internationalization plugin
 - ⦿ **vuetify**: Material component framework

Vue.js Ecosystem

- ⦿ Even more great plugins:
- ⦿ [https://curated.Vue.js.org/module/
github_com:vuetifyjs](https://curated.Vue.js.org/module/github_com:vuetifyjs)

Under the hood

How does Vue Work?

- ⦿ In JS, you create an instance of Vue.js.
- ⦿ In the constructor, you pass an options obj.
- ⦿ You identify the DOM element from your HTML that you wish to be controlled by Vue.
- ⦿ Vue takes the HTML from that element and creates a template function from it.
- ⦿ The template function generates the HTML for the browser using info/data from the vue instance.

Creating Vue instance

```
new Vue({  
  el: '#app'  
}) ;
```

```
<div id="app">  
  <span>I love ice cream!</span>  
</div>
```

Creating Vue instance

```
new Vue({  
  el: '#app',  
  data: {  
    product: 'ice cream'  
  },  
  ...  
}) ;
```

Using dynamic data

```
new Vue({  
  el: '#app',  
  data: {  
    product: 'ice cream'  
  }  
}) ;
```

```
<div id="app">  
  <span>I love {{ product }}!</span>  
</div>
```

Using methods

```
new Vue({  
  el: '#app',  
  methods: {  
    locateMe: function() {  
      return "Denver"  
    }  
  } ) ;  
  
<div id="app">  
  <span>I love it in {{ locateMe() }} !</span>  
</div>
```

Using data & methods

```
new Vue({  
  el: '#app',  
  data: { place: 'Denver' },  
  methods: {  
    locateMe: function() {  
      return this.place;  
    }  
  }  
}) ;
```

```
<div id="app">  
  <span>I love it in {{ locateMe() }} !</span>  
</div>
```

Dynamic HTML attributes

```
new Vue({  
  el: '#app',  
  data: {  
    company: {  
      name: 'Censinet',  
      link: 'http://censinet.com'  
    }  
  }  
}) ;  
  
<div id="app">  
  <span>I love working at  
    <a href="{{ company.link }}>{{ company.name }}</a>  
  </span>  
</div>
```

Vue's v-bind directive

```
new Vue({  
  el: '#app',  
  data: {  
    company: {  
      name: 'Censinet',  
      link: 'http://censinet.com'  
    }  
  }  
}) ;  
  
<div id="app">  
  <span>I love working at  
    <a v-bind:href="company.link">{{ company.name }}</a>  
  </span>  
</div>
```

Vue's binding ":" shortcut

```
new Vue({  
  el: '#app',  
  data: {  
    company: {  
      name: 'Censinet',  
      link: 'http://censinet.com'  
    }  
  }  
}) ;  
  
<div id="app">  
  <span>I love working at  
    <a :href="company.link">{ { company.name } }</a>  
  </span>  
</div>
```

Other directives: v-once

```
new Vue({  
  el: '#app',  
  data: {  
    food: 'cheesecake'  
  },  
  methods: {  
    updateName: function () {  
      this.food = 'peas'  
    }  
  }  
}) ;  
  
<div id="app">  
  <span v-once>I love {{ food }}</span>  
  <span>{{ food }} is my comfort food!</span>  
</div>
```

Other directives: v-html

```
new Vue({  
  el: '#app',  
  data: {  
    favorite: {  
      name: 'Google',  
      link: '<a href="http://google.com">Google</a>'  
    }  
  } );  
  
<div id="app">  
  <p>Try searching at {{favorite.name}}!</p>  
  <p>{{ favorite.link }}</p>  
  <p v-html="favorite.link"></p>  
</div>
```

Exercise

- ⦿ <https://github.com/pavlovich/VueFiles>
- ⦿ Use <http://codepen.io> or <http://jsfiddle.net>
- ⦿ Import <https://unpkg.com/vue/dist/vue.js>
- ⦿ Create a hard-coded task with id (number), name, icon URL (find a suitable one hosted on the web) and owner (email) and store that data in a vue instance's data.
- ⦿ Output the task icon, name and priority information.
- ⦿ Also, show the current date and time retrieved from the vue instance using a dynamic method.

Other directives: v-for (collection)

```
new Vue({  
  el: '#app',  
  data: {  
    foods: [  
      'rice', 'apples', 'steak'  
    ]  
  } );  
  
<div id="app">  
  <ul>  
    <li v-for="food in foods">{{ food }}</li>  
  </ul>  
</div>
```

Other directives: v-for (collection/index)

```
new Vue({  
  el: '#app',  
  data: {  
    foods: [  
      'rice', 'apples', 'steak'  
    ]  
  } );  
  
<div id="app">  
  <ul>  
    <li v-for="(f, i) in foods">{{i}} {{f}}</li>  
  </ul>  
</div>
```

Other directives: v-for (object/with key)

```
new Vue({  
  el: '#app',  
  data: {  
    foods: {  
      rice: {type: 'grains', size: 'cup'},  
      apple: {type: 'fruit', size: 'each'}  
    }  
  }  
});  
  
<div id="app">  
  <ul> <li v-for="(p, name) in foods">  
    {{` ${name} (${p.type}) is sold by the ${p.size}`}}  
  </li> </ul>  
</div>
```

Other directives: v-for (over numbers)

```
new Vue({  
  el: '#app'  
});  
  
<div id="app">  
  <ul>  
    <li v-for="num in 100">  
      The count is now {{ num }}  
    </li>  
  </ul>  
</div>
```

Other directives: v-on

```
new Vue({  
  el: '#app',  
  data: {  
    foods: ['rice', 'apples', 'steak']  
  },  
  methods: {  
    enter: (event)=>{event.target.style.backgroundColor: 'red'}  
    exit: (event)=>{event.target.style.backgroundColor: 'transparent'}  
  }) ;  
  
<div id="app">  
  <ul>  
    <li v-for="food in foods"  
      v-on:mouseenter="enter"  
      v-on:mouseleave="exit">{{ food }}</li>  
  </ul>  
</div>
```

Vue's listen "@" shortcut

```
new Vue({  
  el: '#app',  
  data: {  
    foods: ['rice', 'apples', 'steak']  
  },  
  methods: {  
    enter: (event)=>{event.target.style.backgroundColor = 'red'}  
    exit: (event)=>{event.target.style.backgroundColor =  
      'transparent'}  
  }) ;  
  
<div id="app">  
  <ul>  
    <li v-for="food in foods"  
        @mouseenter="enter"  
        @mouseleave="exit">{{ food }}</li>  
  </ul>  
</div>
```

Passing arguments

```
new Vue({  
  el: '#app',  
  data: {  
    foods: ['rice', 'apples', 'steak']  
  },  
  methods: {  
    enter: (food, event) => {event.target.style.backgroundColor: 'red'}  
    exit: (food, event) => {event.target.style.backgroundColor:  
      'transparent'}  
  }) ;  
  
<div id="app">  
  <ul>  
    <li v-for="food in foods"  
        @mouseenter="enter(food, $event)"  
        @mouseleave="exit(food, $event)">{{ food }}</li>  
  </ul>  
</div>
```

Exercise

- ➊ Hardcode a list of tasks in a vue instance
- ➋ Display that list using an unordered list showing only the name
- ➌ When you click on one of these items, change its background to indicate it is selected and display the details of it below the task list.
- ➍ If you click it a second time, ‘unselect’ it and clear the area below the task list.

Using modifiers

```
const myApp = new Vue({  
  el: '#app',  
  data: { x: 0, y: 0 },  
  methods: {  
    move: function(e){ this.x = e.clientX; this.y = e.clientY; },  
    noMove: function(event){ event.stopPropagation(); }  
  }  
});  
  
<div id="app">  
  <div @mousemove="move">  
    This is a test  
    <span @mousemove="noMove"> ... this is only a test!</span>  
    <span @mousemove.stop> ... this, too, is only a test!</span>  
  </div>  
  <div> {{ x }} / {{ y }}</div>  
</div>
```

Key event modifiers

```
const myApp = new Vue({  
  el: '#app',  
  data: { text: 'nothing yet' },  
  methods: {  
    key: function({target}) { this.text = target.value },  
    clear: function({target}) {  
      target.value = "";  
      this.text = 'cleared';  
    }  
  }  
});  
  
<div id="app">  
  <div>  
    <input type="text" @focus="clear" @keyup.enter.space="key" />  
  </div>  
  <div> {{ text }} </div>  
</div>
```

Exercise

- Change the previous exercise so that when you select a task, an input field containing its name appears. The input field is contained in a form and is displayed below the list.
- When you change the text in the input field and press enter, the form is submitted and the edited text becomes the new name for the selected task and the task is then unselected.
- If you click a selected task before pressing enter, it simply is unselected with no update and the input field is hidden.

2-way binding: v-model

```
const myApp = new Vue({  
  el: '#app',  
  data: { text: 'Enter something ...' }  
}) ;  
  
<div id="app">  
  <div>  
    <input type="text" v-model="text"/>  
  </div>  
  <div> {{ text }} </div>  
</div>
```

2-way binding: v-model

```
const myApp = new Vue({
  el: '#app',
  data: { text: 'Type something ... press space or enter to update' },
  methods: {
    key: function({target}){ this.text = target.value },
    clear: function({target}){
      target.value = "";
      this.text = 'cleared';
    }
  }
});

<div id="app">
  <div>
    <input type="text" @focus="clear" @keyup.enter.space="key" />
  </div>
  <div> {{ text }} </div>
</div>
```

Computed properties

```
const myApp = new Vue({
  el: '#app',
  data: { msg: 'Greetings,' , name: 'Peter Pavlovich' },
  methods: {
    key: function({target}){ this.text = target.value },
    clear: function({target}) {
      target.value = "";
      this.text = 'No message for';
    }
  },
  computed: { greeting: function() { return `#${msg} ${name}.` } }
});

<div id="app">
  <div>
    <input type="text" @focus="clear" @keyup.enter.space="key" />
  </div>
  <div> {{ greeting }} </div>
</div>
```

Watching properties

```
const myApp = new Vue({
  el: '#app',
  data: { text: 'initial' },
  watch: {
    text: function(value, original) {
      console.log(`My text changed from ${original} to ${value}`)
    }
  },
  methods: { changeText: function() {this.text = 'updated'} }
});
```

```
<div id="app">
  <div>
    <span>My text is {{ text }}</span>
  </div>
  <button @click="changeText">Update text</button>
</div>
```

Exercise

- Convert the previous example to use v-model to update a temporary copy of the selected task prior to updating.
- Display a confirmation dialog to the user if they attempt to unselect or select a different task when editing a task and unsaved changes are present.

Dynamic Styling (via classes)

```
const myApp = new Vue({
  el: '#app',
  data: { isRed: false },
  methods: {
    toggleColor: function() { this.isRed = !this.isRed}
  }
});
```



```
<div id="app">
  <div class="demo-text"
    @click="toggleColor"
    :class="{ 'make-it-red': isRed, 'make-it-blue': !isRed }" >
    Click to change colors
  </div>
</div>
```

css is as you would expect it to be.

Dynamic Styling (via code)

```
const myApp = new Vue({  
  el: '#app',  
  data: { isRed: false },  
  methods: { toggleColor: function() { this.isRed = !this.isRed} },  
  computed: { myClasses: function(){  
    return { 'make-it-red': this.isRed, 'make-it-blue': !this.isRed}  
  }  
}) ;
```

```
<div id="app">  
  <div class="demo-text"  
    @click="toggleColor"  
    :class="myClasses" >  
    Click to change colors  
  </div>  
</div>
```

css is as you would expect it to be.

Dynamic CSS (via data props)

```
const myApp = new Vue({  
  el: '#app',  
  data: { isRed: false, dynClass: 'super-bold' },  
  methods: { toggleColor: function() { this.isRed = !this.isRed} }  
});
```

```
<div id="app">  
  <div class="demo-text"  
    @click="toggleColor"  
    :class="[dynClass, { 'make-it-red': isRed}]" >  
    Click to change colors  
  </div>  
</div>
```

css is as you would expect it to be.

Dynamic CSS (via computed styles)

```
const myApp = new Vue({  
  el: '#app',  
  data: { bgColor: 'green', bdrStyle: '1px solid blue' }  
});  
  
<div id="app">  
  <div class="demo-text"  
    :style="{ 'background-color': bgColor, borderTop: bdrStyle }">  
    Sample text  
  </div>  
</div>  
  
css is as you would expect it to be.
```

Conditional Rendering: v-if / v-else-if / v-else

```
const myApp = new Vue({  
  el: '#app',  
  data: { showFormal: false, showCasual: false }  
});  
  
<div id="app">  
  <div v-if="showFormal">  
    Greetings!  
  </div>  
  <div v-else-if="showCasual">  
    Hi there!  
  </div>  
  <div v-else>  
    There you are!  
  </div>  
</div>
```

Conditional Rendering: v-show

```
const myApp = new Vue({  
  el: '#app',  
  data: { showFormal: false, showCasual: false }  
});  
  
<div id="app">  
  <div v-show="showFormal">  
    Greetings!  
  </div>  
  <div v-show="showCasual">  
    Hi there!  
  </div>  
</div>
```

Rendering multiple top-level elements

```
const myApp = new Vue({  
  el: '#app',  
  data: { showMe: true }  
});
```

```
<div id="app">  
  <template v-if="showMe">  
    <div> One </div>  
    <div> Two </div>  
  </template>  
</div>
```

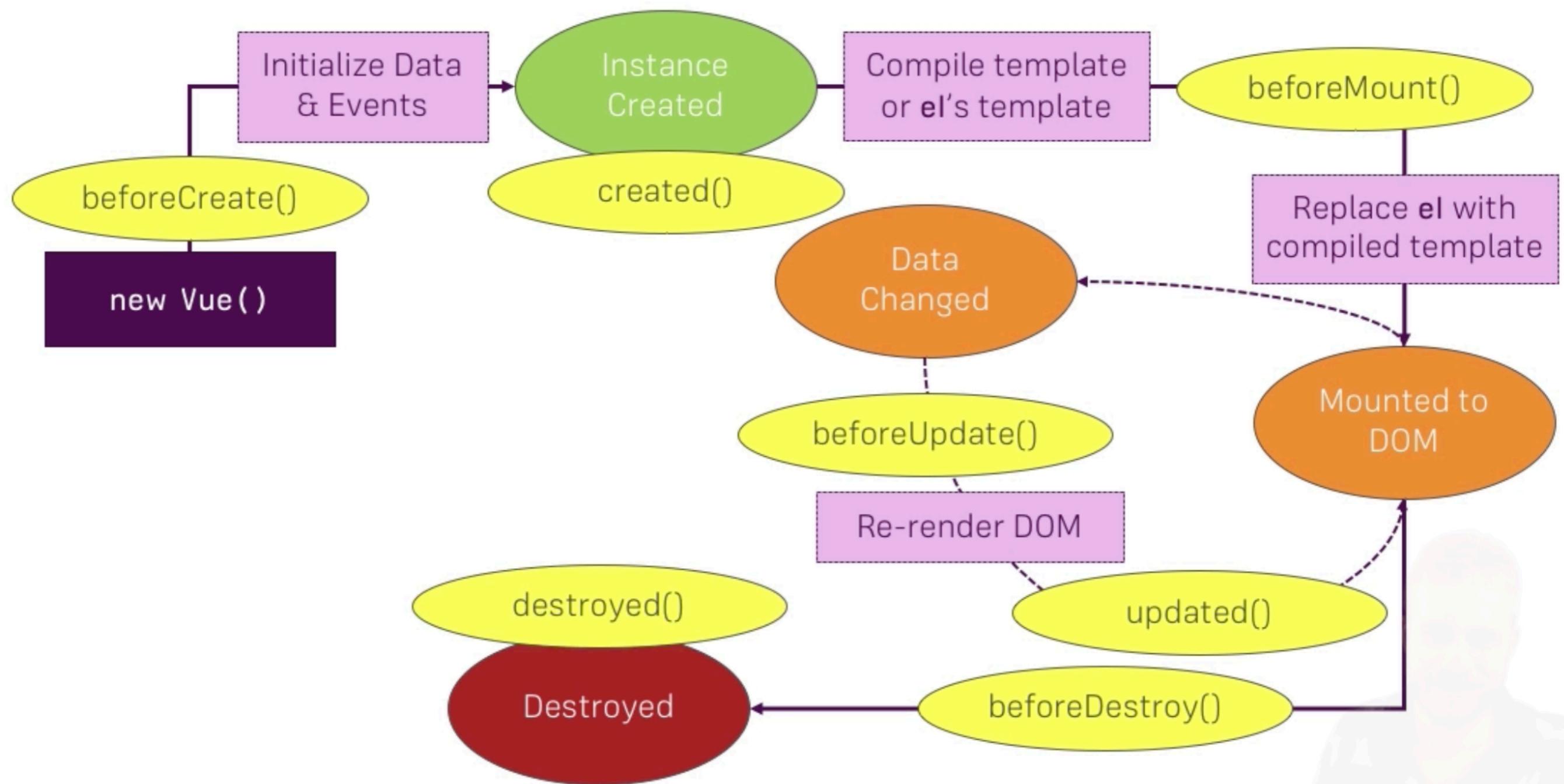
Referencing HTML elements

```
const myApp = new Vue({  
  el: '#app',  
  data: { showMe: true },  
  methods: {  
    updateText: function() {  
      this.$refs.one.innerText = 'New One'  
    }  
  } );  
  
<div id="app">  
  <template v-if="showMe">  
    <div ref="one" @click="updateText"> One </div>  
    <div> Two </div>  
  </template>  
</div>
```

Manually Mounting Vue

```
const myApp = new Vue({  
  data: { showMe: true },  
  methods: {  
    updateText: function() {  
      this.$refs.one.innerText = 'New One'  
    }  
  } );  
  
myApp.$mount('#app');  
  
<div id="app">  
  <template v-if="showMe">  
    <div ref="one" @click="updateText"> One </div>  
    <div> Two </div>  
  </template>  
</div>
```

Vue instance lifecycle



Project Setup

Install WebStorm

☞ <https://www.jetbrains.com/webstorm/download/>

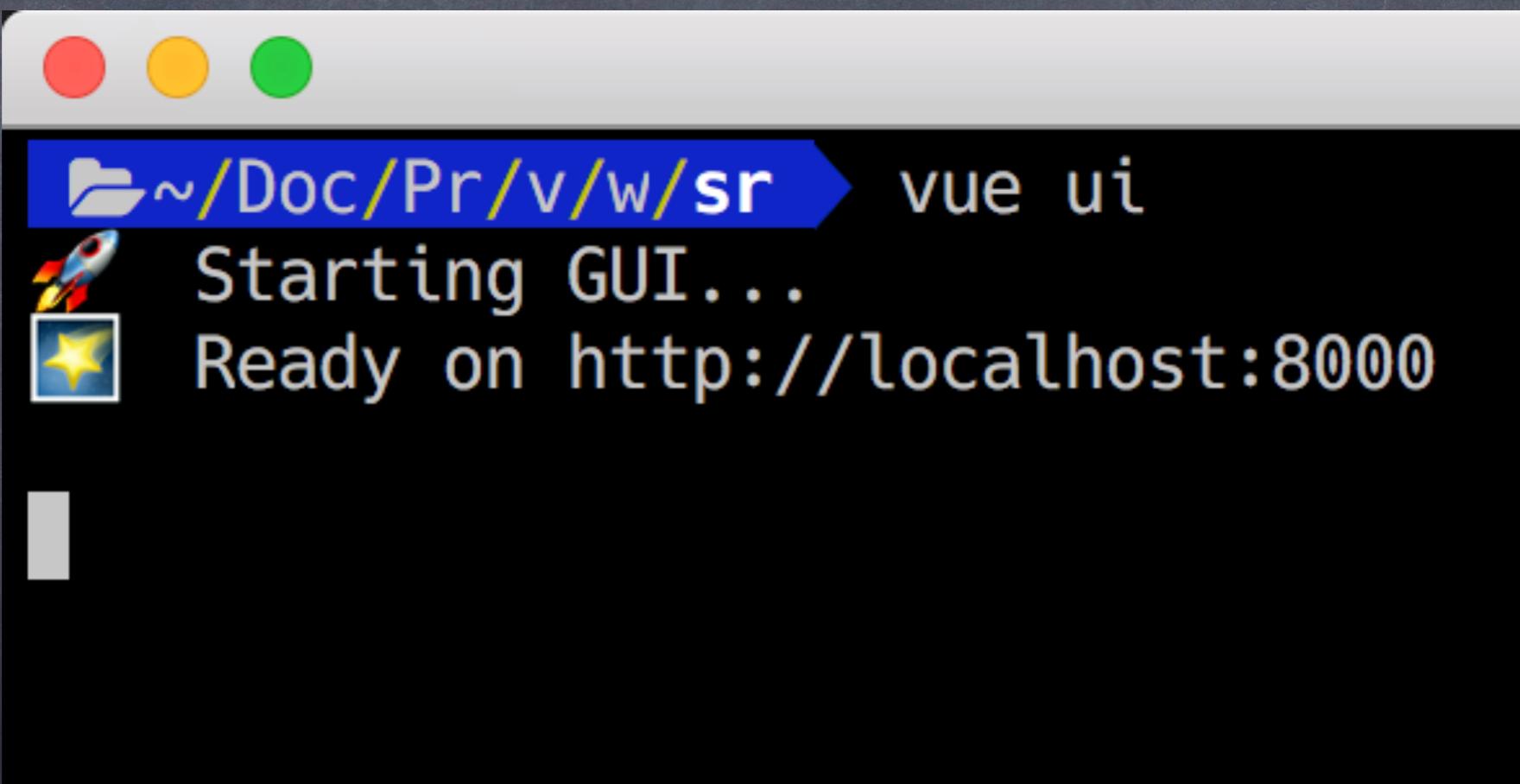
Install Sourcetree

☞ <https://www.atlassian.com/software/sourcetree>

Install the vue-cli

```
npm install -g yarn  
npm install -g @vue/cli
```

Using the all-new GUI Project Manager



Using the all-new GUI Project Manager

The screenshot shows a web browser window titled "[Beta] Vue Project Manager". The address bar indicates the URL is "localhost:8000/project/select". The browser interface includes standard navigation buttons (back, forward, search, etc.) and a toolbar with various icons. Below the toolbar, there are several tabs or links: "Apps", "Meteor 1.8 erases ...", "MongoDB Free M...", "Log In - Stack Ov...", "aurelia-contrib/aw...", "Spark Chocolate", and "Thank you for you...". The main content area is titled "Vue Project Manager" and features a large purple button labeled "Projects". To the right of the button are "Create" and "Import" buttons. A large paperclip icon is centered below the "Projects" button, accompanied by the text "No existing projects".

Using the all-new GUI Project Manager

The screenshot shows a web browser window titled "[Beta] Vue Project Manager". The address bar indicates the URL is `localhost:8000/project/select`. The main content area is titled "Vue Project Manager" and features a navigation bar with "Projects", "Create" (which is highlighted), and "Import". Below this is a toolbar with icons for "Users", "Documents", "Presentations", and file names "vue", "workshop", and "src". A green button at the bottom left says "+ Create a new project here". The status bar at the bottom shows "No project open", the current directory as "/Users/pavlovich/Documents/Presentations/vue/workshop/src", and a message "Ready on http://localhost:8000". The date and time "12/2/2018, 5:55:22 AM" are also displayed.

Using Vue ui

Create a new project

Details Presets Features Configuration

Project folder

Type a name

/Users/pavl...entations/vue/workshop/src/ 

Package manager

Default

Additional options

Overwrite target folder if it exists 

Git repository

Initialize git repository (recommended) 

Initial commit message (optional)

 Cancel  Next →

Using Vue ui

Create a new project

☰ Details ✅ Presets 🧑 Features ⚙ Configuration

Project folder

taskmaster

/...entations/vue/workshop/src/**taskmaster**

Package manager

yarn

Additional options

Overwrite target folder if it exists

Git repository

Initialize git repository (recommended)

From [Vue UI Generator](#)

×

Cancel

Next →

Using Vue ui

Create a new project

≡ Details Presets Features Configuration

i A preset is an association of plugins and configurations. After you've selected features, you can optionally save it as a preset so that you can reuse it for future projects, without having to reconfigure everything again.

Select a preset

- censinet_standard
vue-router, vuex, sass, babel, typescript, pwa, eslint, unit-mocha, e2e-cypress (Use config files)
- Standard_presentation_config
babel, eslint (Use config files)
- NFJS_Workshop
vue-router, vuex, less, babel, eslint (Use config files)
- test preset
vue-router, vuex, less, babel, eslint
- Default preset
babel, eslint
- Manual
Manually select features
- Remote preset

[← Previous](#) [Next →](#)

Using Vue ui

Create a new project

≡ Details ✅ Presets  Features ⚙ Configuration

Babel Transpile modern JavaScript to older versions (for compatibility) 	<input checked="" type="checkbox"/>
TypeScript Add support for the TypeScript language 	<input checked="" type="checkbox"/>
Progressive Web App (PWA) Support Improve performances with features like Web manifest and Service workers 	<input checked="" type="checkbox"/>
Router Structure the app with dynamic pages 	<input checked="" type="checkbox"/>
Vuex Manage the app state with a centralized store 	<input checked="" type="checkbox"/>
CSS Pre-processors Add support for CSS pre-processors like Sass, Less or Stylus 	<input checked="" type="checkbox"/>
Linter / Formatter Check and enforce code quality with ESLint or Prettier 	<input checked="" type="checkbox"/>
Unit Testing Add a Unit Testing solution like Jest or Mocha 	<input checked="" type="checkbox"/>
E2E Testing Add an End-to-End testing solution to the app like Cypress or Nightwatch 	<input checked="" type="checkbox"/>
Use config files Use specific configuration files (like '.babelrc') instead of using 'package.json'.	<input checked="" type="checkbox"/>

 Previous Next 

Using Vue ui

Create a new project

≡ Details

✓ Presets

👤 Features

⚙ Configuration

Use history mode for router? (Requires proper server setup for index fallback in production)

By using the HTML5 History API, the URLs don't need the '#' character anymore. [More Info](#)



Pick a CSS pre-processor:

PostCSS, Autoprefixer and CSS Modules are supported by default.

Less



Pick a linter / formatter config:

Checking code errors and enforcing an homogeneous code style is recommended.

ESLint + Prettier



Pick additional lint features:

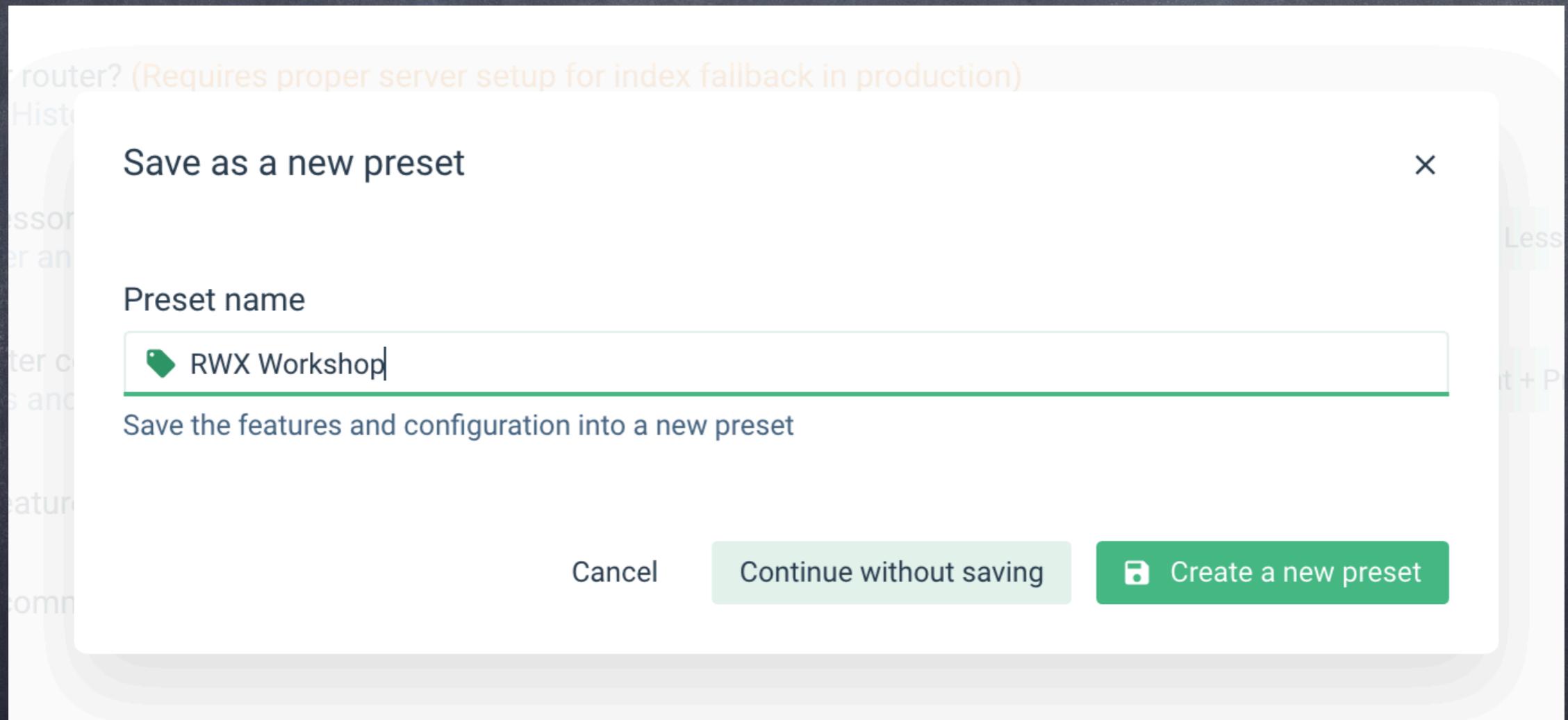
Lint on save

Lint and fix on commit

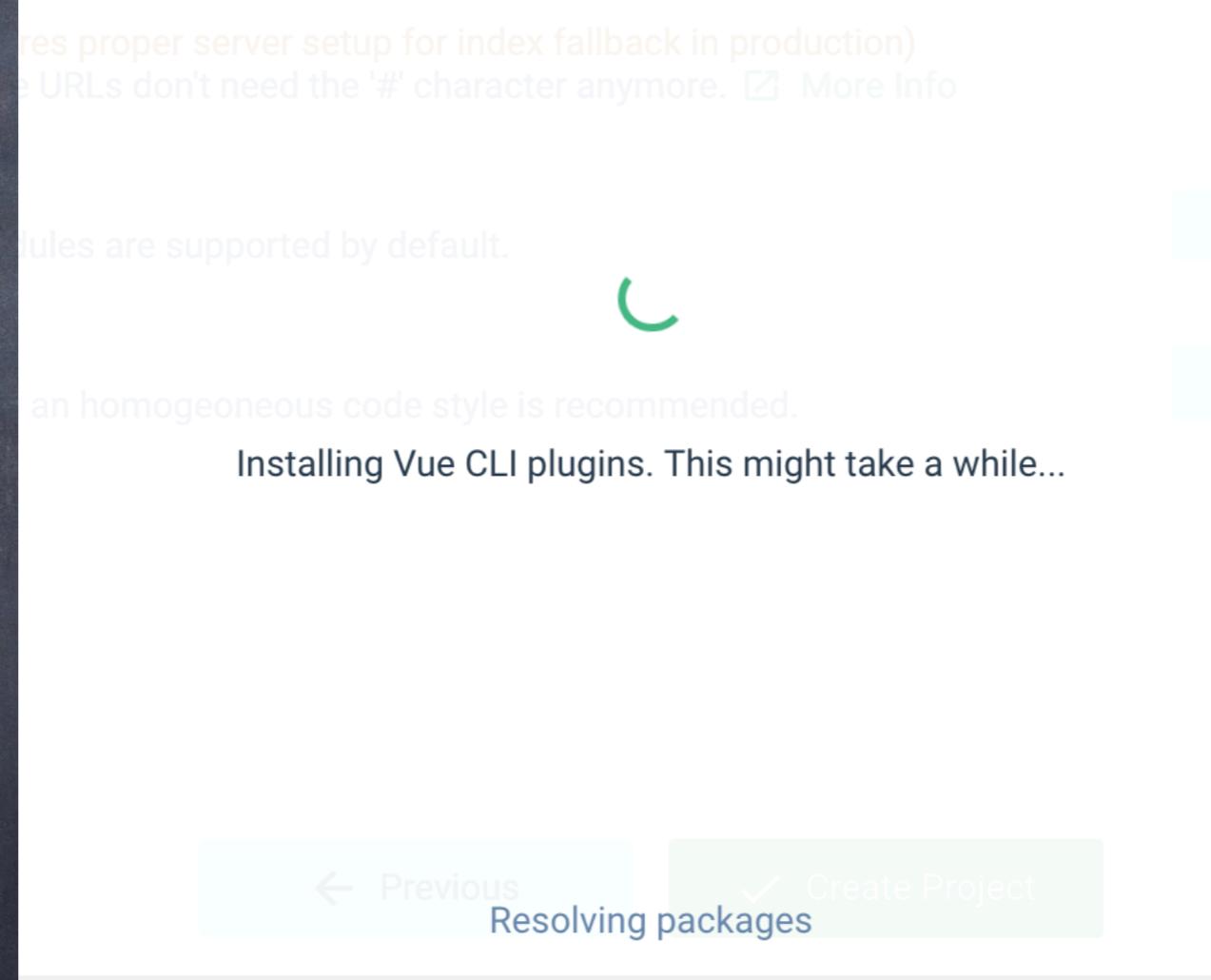
← Previous

✓ Create Project

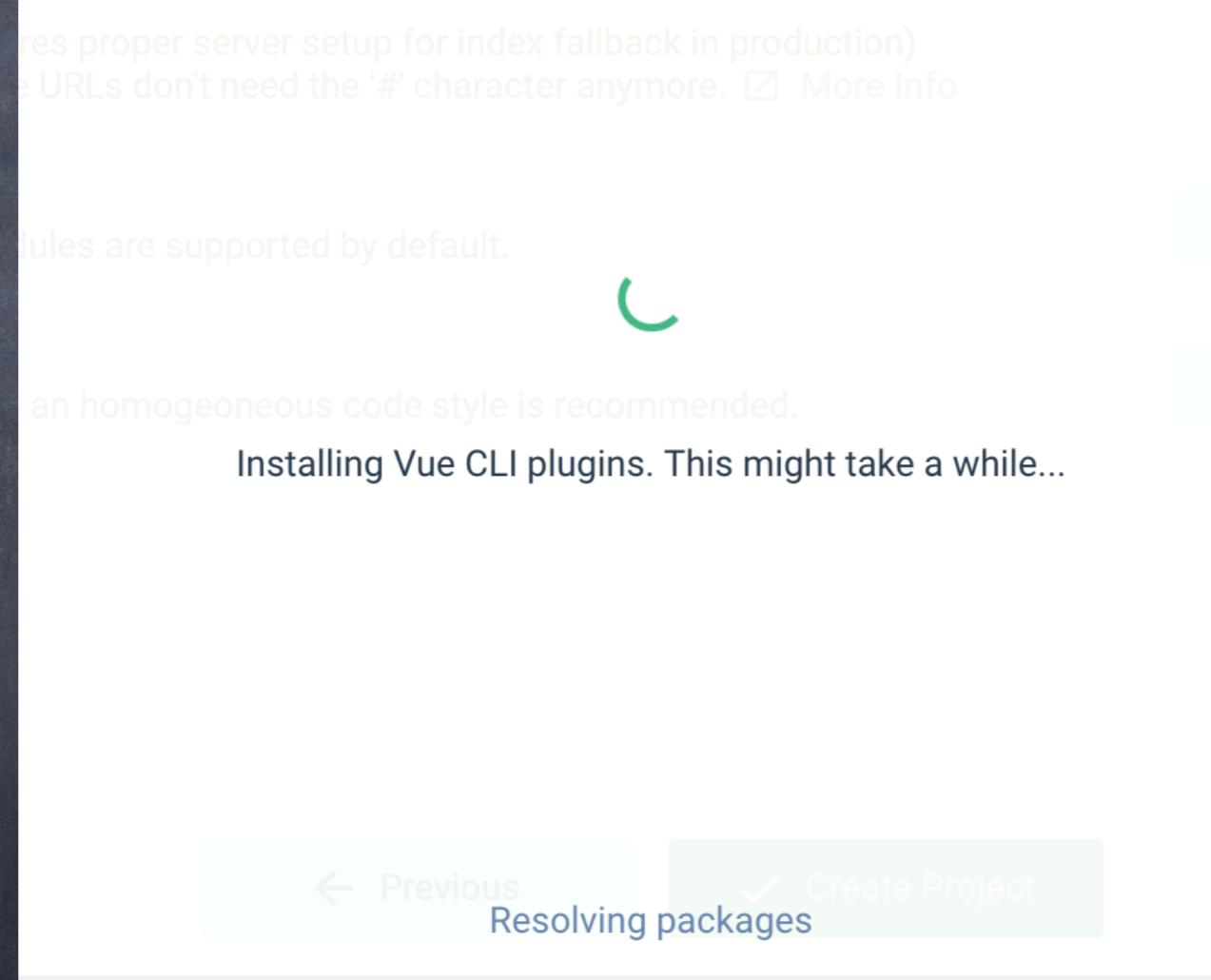
Using Vue ui



Using Vue ui



Using Vue ui



Using Vue ui

Project dashboard

Welcome tips



Welcome to your new project!

You are looking at the project dashboard where you can put widgets. Use the 'Customize' button to add more! Everything is automatically saved.

On the left are the different available pages. 'Plugins' let you add new Vue CLI plugins, 'Dependencies' for managing the packages, 'Configuration' to configure the tools and 'Tasks' to run scripts (for example webpack).

Return to the project manager with the dropdown at the top left of the screen or the home button in the status bar at the bottom.

Customize

Kill port

Ready to kill

Enter a network port

Kill

Understood

...

Using Vue ui

Project dashboard

Customize

Kill port

Ready to kill

Enter a network port

Kill

A screenshot of a Vue.js project dashboard. The main header says "Project dashboard". On the right, there's a green button labeled "Customize". Below the header, there's a sidebar with five icons: a green square with a dot, a puzzle piece, a document, a gear, and a clipboard. The main area features a component titled "Kill port" with a sub-section "Ready to kill". It includes a text input field "Enter a network port" and a green button "Kill" with a lightning bolt icon.

Using Vue ui

Project plugins

[+ Add plugin](#) [⋮](#)

Installed plugins

	<code>@vue/cli-service</code>	version 3.2.0	latest 3.2.0	 Official	 Installed	local service for vue-cli projects		More Info
	<code>@vue/cli-plugin-babel</code>	version 3.2.0	latest 3.2.0	 Official	 Installed	babel plugin for vue-cli		More Info
	<code>@vue/cli-plugin-eslint</code>	version 3.2.1	latest 3.2.1	 Official	 Installed	eslint plugin for vue-cli		More Info

Using Vue ui

Add a plugin

Search Configuration Files changed

vuetify

-  **vue-cli-plugin-vuetify** 0.4.6
Vuetify Framework Plugin for Vue CLI 3  57.5K  vuetifyjs 
-  **vue-cli-plugin-form-generator-vuetify** 0.2.3
Plugin for Vue CLI 3  2.8K  Gripon-web 
-  **vue-cli-plugin-vuetify-cordova** 0.0.13
Vuetify Cordova plugin. Make the world easy  7.6K  canhkieu 
-  **vue-cli-plugin-vuetify-electron** 0.0.7
Electron plugin for vue-cli-3  5.4K  canhkieu 

1

...

Search by algolia

Browse local plugin

Cancel

Install vue-cli-plugin-vuetify

Using Vue ui

Add a plugin

Search Configuration Files changed

Installation of vue-cli-plugin-vuetify

Choose a preset: Default (recommended) (Default) ▾

Cancel Finish installation

This screenshot shows the 'Add a plugin' dialog from the Vue UI application. The 'Configuration' tab is active. The main area displays the message 'Installation of vue-cli-plugin-vuetify'. Below it, a 'Choose a preset:' dropdown is set to 'Default (recommended) (Default)'. At the bottom, there are 'Cancel' and 'Finish installation' buttons.

Using Vue ui

Add a plugin

The screenshot shows a GitHub commit interface. The top navigation bar includes 'Search', 'Configuration', and 'Files changed' (which is underlined). On the left, there's a sidebar with icons for Home, Pull Requests, Issues, and Repository. The main area shows a diff for the file `package.json`. The changes are as follows:

```
diff --git a/package.json b/package.json
index 1234567..8901234 100644
--- a/package.json
+++ b/package.json
@@ -10,16 +10,26 @@
   "dependencies": {
     "vue": "^2.5.17",
     "vue-router": "^3.0.1",
+    "vuetify": "^1.3.0",
     "vuex": "^3.0.1"
   },
   "devDependencies": {
@@ -23,16 +24,26 @@
     "less": "^3.0.4",
     "less-loader": "^4.1.0",
     "lint-staged": "^7.2.2",
-    "vue-template-compiler": "^2.5.17"
+    "stylus": "^0.54.5",
+    "stylus-loader": "^3.0.1",
+    "vue-cli-plugin-vuetify": "^0.4.6",
+    "vue-template-compiler": "^2.5.17",
+    "vuetify-loader": "^1.0.5"
   },
   "gitHooks": {
     "pre-commit": "lint-staged"
   }
}
```

At the bottom, there are two buttons: 'Commit changes' with a downward arrow icon and 'Skip'.

Using Vue ui

Project dashboard

Run task

Run task

Configure widget

Kill port

Rea Kill port

Enter a network port

Kill

Add widgets

Done

Welcome tips

Some tips to help you get sta...

+

Plugin updates

Monitor plugin updates

+

Dependency updates

Monitor dependencies updat...

+

Vulnerability check

Check for known vulnerabiliti...

+

Run task

Shortcut to run a task

+

News feed

Read news feed

+

...

Using Vue ui

Project dashboard

Customize

Run task

Configure widget

Kill port

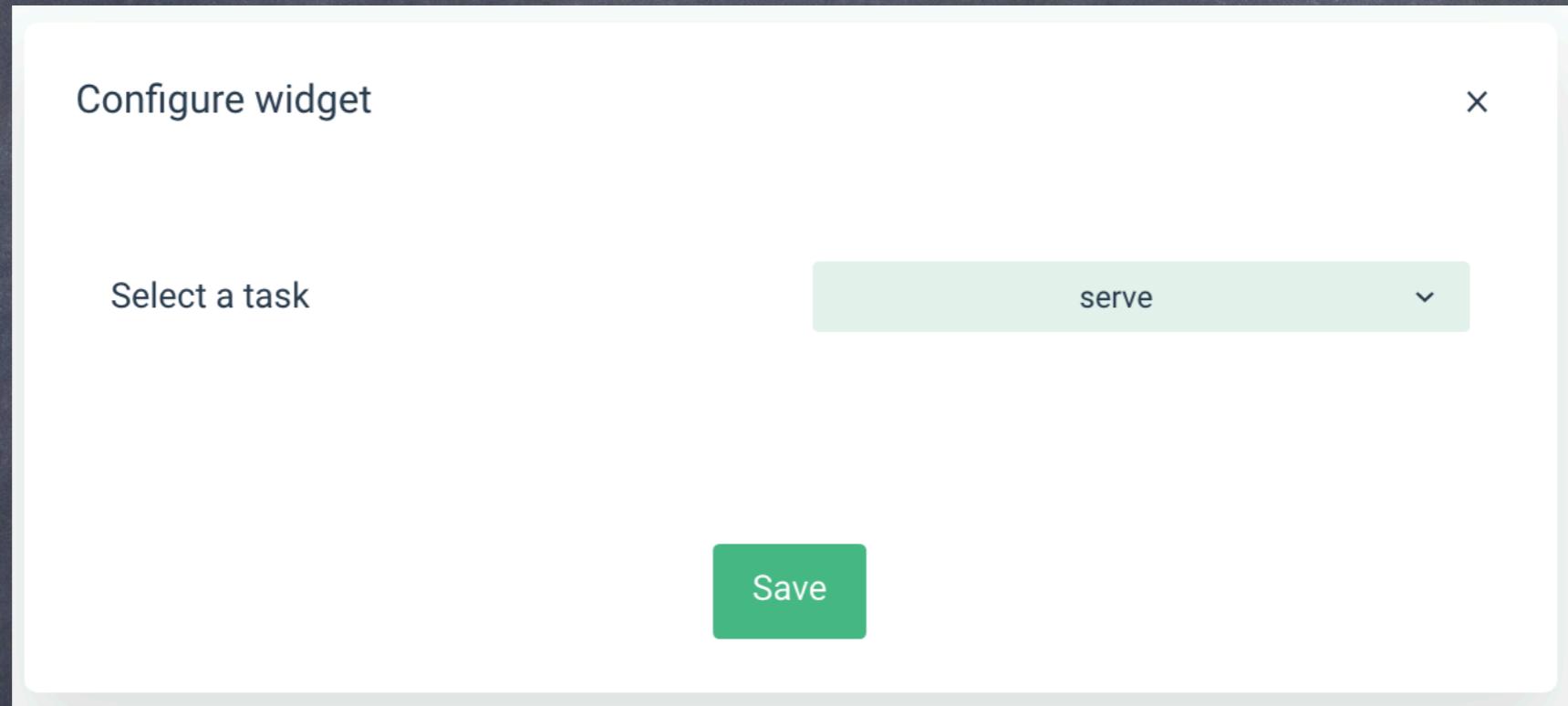
Ready to kill

Enter a network port

Kill

The screenshot displays a 'Project dashboard' interface with a light green header bar. On the far left, there is a vertical sidebar with five icons: a grid, a puzzle piece, a document, a gear, and a clipboard. The main area features two large, rounded rectangular cards. The first card, titled 'Run task', contains a blue gear icon and a green 'Configure widget' button. The second card, titled 'Kill port', includes a lightning bolt icon, a message 'Ready to kill', an input field for 'Enter a network port', and a green 'Kill' button.

Using Vue ui



Using Vue ui

Project dashboard

Customize

Run task

serve
Compiles and hot-reloads for development...

Run task Go to Task

Kill port

Ready to kill

Enter a network port Kill

The screenshot shows the Vue UI Project Dashboard. On the left, there's a vertical sidebar with icons for file, puzzle, document, gear, and list. The main area has a light green header with the title 'Project dashboard' and a 'Customize' button. Below the header, there are two main sections. The first section, 'Run task', contains a card for 'serve'. It includes a blue hexagon icon, the text 'serve Compiles and hot-reloads for development...', a green 'Run task' button with a play icon, and a grey 'Go to Task' button with a list icon. The second section, 'Kill port', has a grey background. It features a lightning bolt icon, the text 'Ready to kill', a text input field with 'Enter a network port', and a green 'Kill' button with a lightning bolt icon.

Using Vue ui

Project dashboard

Customize

Run task

serve
Running

Stop task Go to Task

Kill port

Ready to kill

Enter a network port Kill

The screenshot shows the Vue UI Project Dashboard. On the left, there's a vertical sidebar with icons for file, puzzle, document, gear, and clipboard. The main area has a light green header with the title 'Project dashboard' and a 'Customize' button. Below the header, there are two main sections. The first section, 'Run task', contains a blue cube icon with a green dot, the text 'serve' and 'Running', and two buttons: 'Stop task' (purple) and 'Go to Task' (light green). The second section, 'Kill port', has a lightning bolt icon, the text 'Ready to kill', a text input field 'Enter a network port', and a green 'Kill' button.

Using Vue ui

Project dashboard

Customize

The screenshot shows a project dashboard interface with a sidebar containing icons for file, settings, and other project management tasks. The main area displays two task cards and a kill port section.

Task Card 1: Run task `serve` (Running). Includes a Stop task button and a Go to Task button.

Task Card 2: Run task `inspect` (Done). Includes a Run task button and a Go to Task button.

Kill Port Section: Shows a lightning bolt icon and the text "Ready to kill". A text input field says "Enter a network port" and a Kill button is present.

Using Vue ui

Project tasks

- `serve` Running
- `build` Compiles and minifies for p...
- `lint` Lints and fixes files
- `inspect` Done

`serve` Compiles and hot-reloads for development

vue-cli-service serve

Status Success Errors 0 Warnings 0

Assets 5.2MB (Parsed) Modules 1.7MB (Parsed) Dependencies 1.6MB 97.96%

Idle (12s)

Speed stats

Global	Average	6s	Mobile	Edge	175.1s	2G	150.17s	
3G	Slow	104.96s	3G	Basic	26.44s	3G	Fast	26.29s
4G		4.82s	LTE		3.56s	Dial Up		836.59s
DSL		27.93s	Cable		8.39s	FIOS		2.1s

Assets

	Parsed	Global	3G Slow	3G Fast
<code>app.js</code>	5.2MB	5.99s	104.66s	26.22s
<code>about.js</code>	12.7kB	0.04s	0.65s	0.21s
<code>favicon.ico</code>	1.1kB	0.03s	0.42s	0.16s
<code>index.html</code>	0.9kB	0.03s	0.42s	0.15s

Dependencies

vuetify	1.0MB	
vue	207.0kB	
sockjs-client	180.5kB	
vue-router	64.4kB	
html-entities	57.4kB	
core-js	39.3kB	

Using Vue ui

Project tasks

- `serve` Running
- `build` Compiles and minifies for p...
- `lint` Lints and fixes files
- `inspect` Done

`serve` Compiles and hot-reloads for development

`vue-cli-service serve`

Stop task Parameters

Output Dashboard Analyzer

Analyzer

Go up Go to home Chunk app (app) Parsed ?

vue-router
Stats: 64.4kB
Parsed: 64.4kB
Gzip: 0.0kB

Using Vue ui

Project tasks

The screenshot shows the Vue UI TaskMaster interface. On the left, there's a sidebar with icons for file management, project settings, and more. Below it is a list of project tasks:

- serve** Running
- build** Compiles and minifies for p...
- lint** Lints and fixes files
- inspect** Done

The main area is focused on the **serve** task, which is currently running. It has a status bar at the top with the command `vue-cli-service serve`. Below that are buttons for **Stop task**, **Parameters**, and a copy icon. A tab bar at the bottom includes **Output** (which is selected), **Dashboard**, and **Analyzer**.

The **Output** panel displays the build logs:

```
67% building 493/513 modules 20 active ...ylus/components/_date-picker-table
67% building 494/513 modules 19 active ...ylus/components/_date-picker-table
67% building 495/513 modules 18 active ...ylus/components/_date-picker-table
68% building 496/513 modules 17 active ...ylus/components/_date-picker-table
68% building 497/513 modules 16 active ...ylus/components/_date-picker-table
68% building 498/513 modules 15 active ...ylus/components/_date-picker-table
68% building 499/513 modules 14 active ...ylus/components/_date-picker-table
68% building 500/513 modules 13 active ...ylus/components/_date-picker-table
68% building 501/513 modules 12 active ...ylus/components/_date-picker-table
68% building 502/513 modules 11 active ...ylus/components/_date-picker-table
68% building 503/513 modules 10 active ...ylus/components/_date-picker-table
68% building 504/513 modules 9 active ...ylus/components/_date-picker-table.
69% building 505/513 modules 8 active ...ylus/components/_date-picker-table.
69% building 506/513 modules 7 active ...ylus/components/_date-picker-table.
69% building 507/513 modules 6 active ...ylus/components/_date-picker-table.
69% building 508/513 modules 5 active ...ylus/components/_date-picker-table.
69% building 509/513 modules 4 active ...ylus/components/_date-picker-table.
69% building 510/513 modules 3 active ...ylus/components/_date-picker-table.
69% building 511/513 modules 2 active ...ylus/components/_date-picker-table.
69% building 512/513 modules 1 active ...ylus/components/_date-picker-table.
98% after emitting DONE Compiled successfully in 11908ms 06:27:32
```

A **WARN** message follows:

Couldn't parse bundle asset "/Users/pavlovich/Documents/Presentations/vue/wor
kshop/src/taskmaster/dist/about.js".
Analyzer will use module sizes from stats file.

At the bottom, it shows the app is running at:

- Local: <http://localhost:8080/>
- Network: <http://172.20.2.161:8080>

Note that the development build is not optimized.
To create a production build, run `yarn build`.

Using Vue ui

```
~/Doc/Pr/v/w/src ➜ vue ui
Starting GUI...
Ready on http://localhost:8000
{"type": "warning", "data": "\">@vue/cli-plugin-babel > babel-loader@8.0.4\" has unmet peer dependency \"webpack@>=2\"
"} {"type": "warning", "data": "\">@vue/cli-plugin-eslint > eslint-loader@2.1.1\" has unmet peer dependency \"webpack@>=2.0.0 <5.0.0\"
"} {"type": "warning", "data": "\" > less-loader@4.1.0\" has unmet peer dependency \"webpack@^2.0.0 || ^3.0.0 || ^4.0.0\"
"} Invoking generator for vue-cli-plugin-vuetify...
Installing additional dependencies...

{"type": "warning", "data": "\">@vue/cli-plugin-babel > babel-loader@8.0.4\" has unmet peer dependency \"webpack@>=2\"
"} {"type": "warning", "data": "\">@vue/cli-plugin-eslint > eslint-loader@2.1.1\" has unmet peer dependency \"webpack@>=2.0.0 <5.0.0\"
"} {"type": "warning", "data": "\" > less-loader@4.1.0\" has unmet peer dependency \"webpack@^2.0.0 || ^3.0.0 || ^4.0.0\"
"} {"type": "warning", "data": "\" > vuetify-loader@1.0.8\" has unmet peer dependency \"webpack@^4.0.0\"
"} 
Running completion hooks...
✓ Successfully invoked generator for plugin: vue-cli-plugin-vuetify
The following files have been updated / added:

src/assets/logo.svg
src/plugins/vuetify.js
package.json
public/index.html
src/App.vue
src/components/Helloworld.vue
src/main.js
src/views/Home.vue
yarn.lock
```

You should review these changes with `git diff` and commit them.

Using Vue ui

The screenshot shows a web browser window with three tabs open: '[Beta] serve - Project tasks -' (closed), 'CLI Service | Vue CLI 3' (closed), and 'taskmaster'. The main content area displays the Vuetify Material Design landing page at localhost:8080. The page features a large blue 'V' logo, the text 'Welcome to Vuetify', and a call to action to join the [Discord Community](#). It also includes sections for 'What's next?' with links to 'Explore components', 'Select a layout', and 'Frequently Asked Questions', and 'Important Links' with links to 'Documentation', 'Chat', 'Made with Vuetify', 'Twitter', and 'Articles'. At the bottom, there is an 'Ecosystem' section with links to 'vuetify-loader', 'github', and 'awesome-vuetify'.

VUETIFY MATERIAL DESIGN LATEST RELEASE

Welcome to Vuetify

For help and collaboration with other Vuetify developers,
please join our online [Discord Community](#).

What's next?

[Explore components](#) [Select a layout](#) [Frequently Asked Questions](#)

Important Links

[Documentation](#) [Chat](#) [Made with Vuetify](#) [Twitter](#) [Articles](#)

Ecosystem

[vuetify-loader](#) [github](#) [awesome-vuetify](#)

Using the vue-cli

```
> cd ~/projects  
projects> vue create taskmaster
```

Vue CLI v3.0.0-rc.4

? Please pick a preset:

censinet_standard (vue-router, vuex, sass, babel, typescript, pwa, eslint, unit-mocha, e2e-cypress)

Standard_presentation_config (babel, eslint)

NFJS_Workshop (vue-router, vuex, less, babel, eslint)

default (babel, eslint)

> Manually select features

Vue CLI v3.0.0-rc.4

? Please pick a preset: Manually select features

? Check the features needed for your project:

> Babel

TypeScript

Progressive Web App (PWA) Support

Router

Vuex

CSS Pre-processors

Linter / Formatter

Unit Testing

E2E Testing

Using the vue-cli

Vue CLI v3.0.0-rc.4

- ? Please pick a preset: Manually select features
- ? Check the features needed for your project: Babel, CSS Pre-processors, Linter
- ? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): SCSS/SASS
- › LESS
- Stylus

Vue CLI v3.0.0-rc.4

- ? Please pick a preset: Manually select features
- ? Check the features needed for your project: Babel, CSS Pre-processors, Linter
- ? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): LESS
- ? Pick a linter / formatter config:
 - ESLint with error prevention only
 - ESLint + Airbnb config
 - ESLint + Standard config
- › ESLint + Prettier

Using the vue-cli

Vue CLI v3.0.0-rc.4

- ? Please pick a preset: Manually select features
- ? Check the features needed for your project: Babel, CSS Pre-processors, Linter
- ? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): LESS
- ? Pick a linter / formatter config: Prettier
- ? Pick additional lint features: (Press `<space>` to select, `<a>` to toggle all, `<i>` to invert selection)
 - > Lint on save
 - Lint and fix on commit

Vue CLI v3.0.0-rc.4

- ? Please pick a preset: Manually select features
- ? Check the features needed for your project: Babel, CSS Pre-processors, Linter
- ? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): LESS
- ? Pick a linter / formatter config: Prettier
- ? Pick additional lint features: Lint on save
- ? Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? (Use arrow keys)
 - > In dedicated config files
 - In package.json

Using the vue-cli

Vue CLI v3.0.0-rc.4

- ? Please pick a preset: Manually select features
- ? Check the features needed for your project: Babel, CSS Pre-processors, Linter
- ? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): LESS
- ? Pick a linter / formatter config: Prettier
- ? Pick additional lint features: Lint on save
- ? Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? In dedicated config files
- ? Save this as a preset for future projects? (y/N) |

Using the vue-cli

Vue CLI v3.2.1

```
✨ Creating project in /Users/pavlovich/Documents/Presentations/vue/workshop/src/taskmaster.  
🗃️ Initializing git repository...  
⚙️ Installing CLI plugins. This might take a while...
```

yarn install v1.7.0

```
info No lockfile found.  
[1/4] 🔎 Resolving packages...  
[2/4] 🚚 Fetching packages...  
[3/4] 🛡️ Linking dependencies...  
[4/4] 📄 Building fresh packages...  
success Saved lockfile.  
info To upgrade, run the following command:  
$ curl --compressed -o- -L https://yarnpkg.com/install.sh | bash  
✨ Done in 19.99s.  
🚀 Invoking generators...  
📦 Installing additional dependencies...
```

yarn install v1.7.0

```
[1/4] 🔎 Resolving packages...  
[2/4] 🚚 Fetching packages...  
[3/4] 🛡️ Linking dependencies...  
[4/4] 📄 Building fresh packages...
```

success Saved lockfile.

```
✨ Done in 7.06s.  
⚓ Running completion hooks...
```

```
📄 Generating README.md...
```

```
🎉 Successfully created project taskmaster.  
👉 Get started with the following commands:
```

```
$ cd taskmaster  
$ yarn serve
```

Using the vue-cli

```
$ cd taskmaster
$ yarn serve

~/Doc/Pr/v/w/src ➔ yarn serve
yarn run v1.7.0
error Couldn't find a package.json file in "/Users/pavlovich/Documents/Presentations/vue/workshop/src"
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.

~/Doc/Pr/v/w/src ➔ cd taskmaster
~/Doc/Pr/v/w/src/t ➔ git ⌘ b7fe5 ↵ master ➔ clear
~/Doc/Pr/v/w/src/t ➔ git ⌘ b7fe5 ↵ master ➔ yarn serve
yarn run v1.7.0
$ vue-cli-service serve
INFO Starting development server...
98% after emitting CopyPlugin

DONE Compiled successfully in 4150ms
```

App running at:
- Local: <http://localhost:8080/>
- Network: <http://172.20.2.161:8080/>

Note that the development build is not optimized.
To create a production build, run `yarn build`.

A screenshot of a web browser window displaying the "Welcome to Your Vue.js App" page. The browser has three tabs open: "[Beta] serve - Project t", "CLI Service | Vue CLI 3", and "taskmaster". The address bar shows "localhost:8080". The main content features a large green and dark blue 'V' logo, followed by the text "Welcome to Your Vue.js App". Below this, there's a link to the "vue-cli documentation". A section titled "Installed CLI Plugins" lists "babel" and "eslint". Another section titled "Essential Links" includes links to "Core Docs", "Forum", "Community Chat", "Twitter", and "News". A final section titled "Ecosystem" includes links to "vue-router", "vuex", "vue-devtools", "vue-loader", and "awesome-vue".

[Beta] serve - Project t | CLI Service | Vue CLI 3 | taskmaster

localhost:8080

Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [eslint](#)

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

Exercise

- ➊ Create a new Vue project named 'taskmaster'
 - ➋ Use the CLI
 - ➌ Follow the selections as on the slides.

Review of base template

⌚ Demo ...

Vue's ecosystem

- ⦿ Vue has a number of 'plug-ins'
 - ⦿ Official
 - ⦿ **Vue-router**
 - ⦿ **Vuex** (centralized store - like Redux)
 - ⦿ Community
 - ⦿ **Vuetify** (Material Design Inspired UI)

Let's add our UI widgets

```
~/.Doc/Pr/v/w/src/t ➜ git ⌘ b7fe5 ↵ master ➜ vue add vuetify  
📦 Installing vue-cli-plugin-vuetify...  
  
yarn add v1.7.0  
[1/4] 🔎 Resolving packages...  
[2/4] 🚚 Fetching packages...  
[3/4] 🔗 Linking dependencies...  
[4/4] 💾 Building fresh packages...  
  
success Saved lockfile.  
success Saved 1 new dependency.  
info Direct dependencies  
└ vue-cli-plugin-vuetify@0.4.6  
info All dependencies  
└ vue-cli-plugin-vuetify@0.4.6  
✨ Done in 4.31s.  
✓ Successfully installed plugin: vue-cli-plugin-vuetify  
  
? Choose a preset: (Use arrow keys)  
❯ Default (recommended)  
  Prototype (rapid development)  
  Configure (advanced)
```

Let's add our UI widgets

```
? Choose a preset: Default (recommended)

🚀 Invoking generator for vue-cli-plugin-vuetify...
📦 Installing additional dependencies...

yarn install v1.7.0
[1/4] 🔎 Resolving packages...
[2/4] 🚚 Fetching packages...
[3/4] 🔗 Linking dependencies...
[4/4] 🏺 Building fresh packages...

success Saved lockfile.
✨ Done in 5.40s.
⚓ Running completion hooks...

✓ Successfully invoked generator for plugin: vue-cli-plugin-vuetify
The following files have been updated / added:

src/assets/logo.svg
src/plugins/vuetify.js
package.json
public/index.html
src/App.vue
src/components/Helloworld.vue
src/main.js
yarn.lock

You should review these changes with git diff and commit them.
```

Let's add our UI widgets

```
147 | </style>
148 |
```

62 warnings found.

62 warnings potentially fixable with the `--fix` option.

You may use special comments to disable some warnings.

Use `// eslint-disable-next-line` to ignore the next line.

Use `/* eslint-disable */` to ignore all warnings in a file.

App running at:

- Local: <http://localhost:8080>/
- Network: <http://172.20.2.161:8080>/

Note that the development build is not optimized.

To create a production build, run `yarn build`.

^C

```
~/.Doc/Pr/v/w/src/t ➤ git ⚡ b7fe5 ↵ master ⚡? yarn lint --fix
```

```
yarn run v1.7.0
```

```
$ vue-cli-service lint --fix
```

```
The following files have been auto-fixed:
```

```
src/App.vue
```

```
src/components/HelloWorld.vue
```

```
src/main.js
```

```
src/plugins/vuetify.js
```

DONE All lint errors auto-fixed.

✨ Done in 2.00s.

Let's add our UI widgets

```
~ Doc/Pr/v/w/src/t git ~ b7fe5 master ! ? yarn serve
yarn run v1.7.0
$ vue-cli-service serve
INFO Starting development server...
98% after emitting CopyPlugin

DONE Compiled successfully in 13512ms
```

App running at:

- Local: <http://localhost:8080/>
- Network: <http://172.20.2.161:8080/>

Note that the development build is not optimized.
To create a production build, run [yarn build](#).

Let's add our UI widgets

The screenshot shows a web browser window with the URL `localhost:8080` in the address bar. The page title is "VUETIFY MATERIAL DESIGN". On the right, there is a "LATEST RELEASE" button. The main content features the Vuetify logo (a stylized blue 'V') and the heading "Welcome to Vuetify". Below this, a call to action encourages users to join the [Discord Community](#). A section titled "What's next?" includes links to "Explore components", "Select a layout", and "Frequently Asked Questions". Another section titled "Important Links" includes links to "Documentation", "Chat", "Made with Vuetify", "Twitter", and "Articles". At the bottom, a section titled "Ecosystem" includes links to "vuetify-loader", "github", and "awesome-vuetify". The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with various icons.

VUETIFY MATERIAL DESIGN

LATEST RELEASE

Welcome to Vuetify

For help and collaboration with other Vuetify developers,
please join our online [Discord Community](#).

What's next?

[Explore components](#) [Select a layout](#) [Frequently Asked Questions](#)

Important Links

[Documentation](#) [Chat](#) [Made with Vuetify](#) [Twitter](#) [Articles](#)

Ecosystem

[vuetify-loader](#) [github](#) [awesome-vuetify](#)

Review of Vuetify App

⌚ Demo ...

Basic *.vue Component

```
<template lang="... | default is the Vue template syntax">
  <SomeComponentINeed></SomeComponentINeed>
</template>

<script lang="typescript | coffee | default is ES6">
  import <SomeComponentINeed>;

  export default {
    name: 'MyComponentName',
    props: [propName1, propName2, ...],
    data: () => { return {stateVar1: initialValue1, ...} },
    computed: {propName: ()=> {return value}, ...},
    methods: {methodName: () => {... do something ...}},
    watch: {propName: (newValue, oldValue){... do something ...}, ...}
    lifecycleMethod1() { ... do something ...}, ...,
    components: {LocalCmpName: SomeComponentINeed, ...}
  }
</script>

<style [scoped] lang="scss | stylus | less | default = plain css">
...
</style>
```

A sneak peak!

- ⦿ Our initial setup is now complete.
- ⦿ Let's explore what we will ultimately build together!
- ⦿ Demo