

Intro VueJS



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://about.softuni.bg>

Table of Contents

1. Why Vue
2. Project setup
3. Vue Single-File-Component
4. Reactivity Fundamentals
5. Template Syntax
6. CSS Features



Have a Question?



sli.do

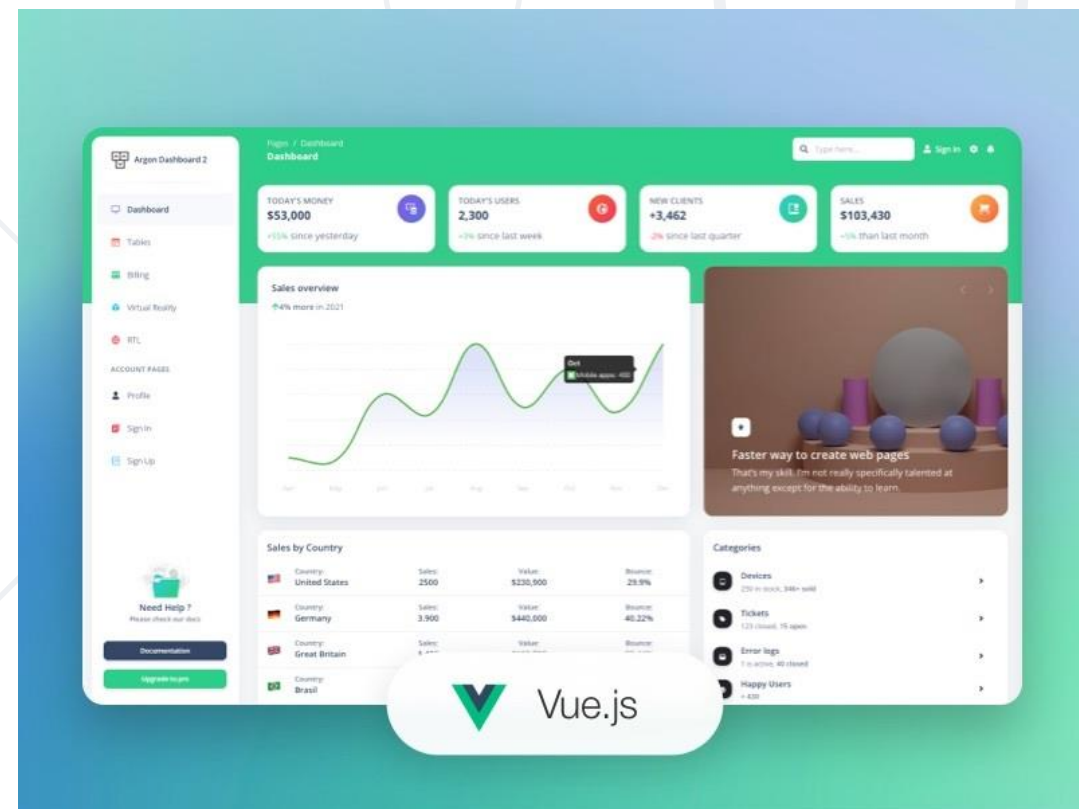
#vue-js



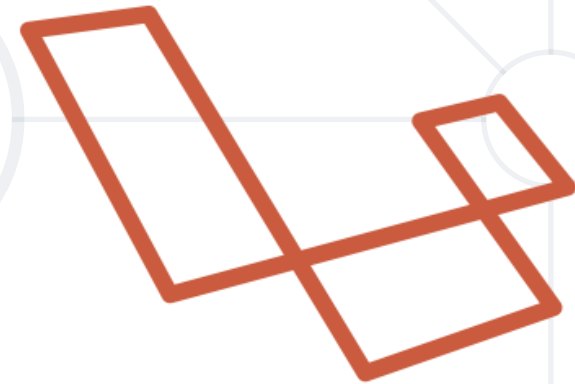
Why Vue?

What is Vue?

- Progressive framework for building user interfaces
- The core library is focused on the view layer only
- Perfectly capable for creating Single-Page Applications
- Created by Evan You (former employee @ Google)
- Combines the best from both React and Angular



- Declarative Rendering: Vue extends standard HTML with a template syntax that allows us to declaratively describe HTML output based on JavaScript state.
- Reactivity: Vue automatically tracks JavaScript state changes and efficiently updates the DOM when changes happen.
- Lightweight – 14kb and very fast performance
- Data Binding – one and two-way
- Components – reusable HTML
- Directives – v-if, v-else, v-for etc.
- Computed Properties & Watchers – listen to changes out-of-the-box



laravel

VueJS vs React

- **Vue**
- Has Virtual DOM
- Template based approach
- Less Popular

- **React**
- Also has Virtual DOM
- JSX approach
- More Popular



VueJS vs Angular

■ Vue

- Has Directives & CLI
- Natively uses JavaScript
- Smaller file size
- Has less build-in features

■ Angular

- Also has Directives & CLI
- Natively uses TypeScript
- Bigger file size
- Has a lot of build-in features





Project Setup

VS Code and Project scaffolding

Vite & creating a new project

- Step 1: Install Node.js and npm <https://nodejs.org/>
- Step 2: Install Vite <https://vitejs.dev/>

```
npm create vite@latest
✓ Project name: ... vite-project
✓ Select a framework: » Vue
✓ Select a variant: » JavaScript

Scaffolding project in ...\\vite-project
```

```
Done. Now run:
cd vite-project
npm install
npm run dev
```

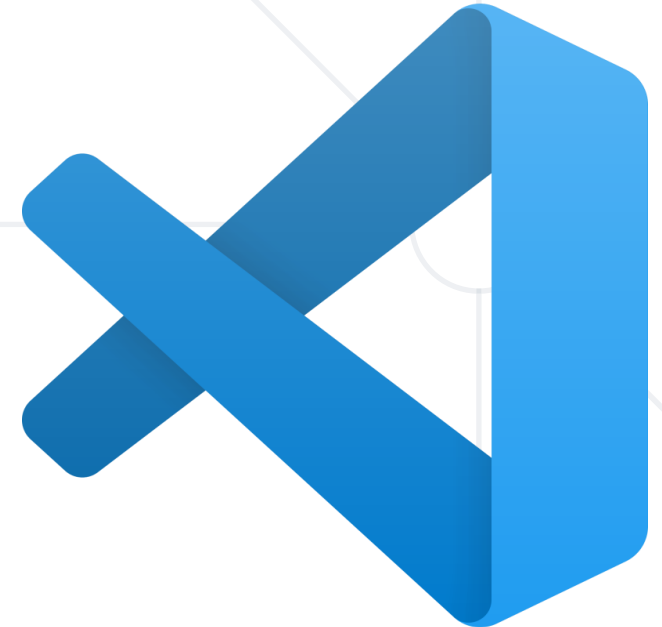
```
npm create vite@latest
```

Need to install the following packages:

[create-vite@4.4.1](#)

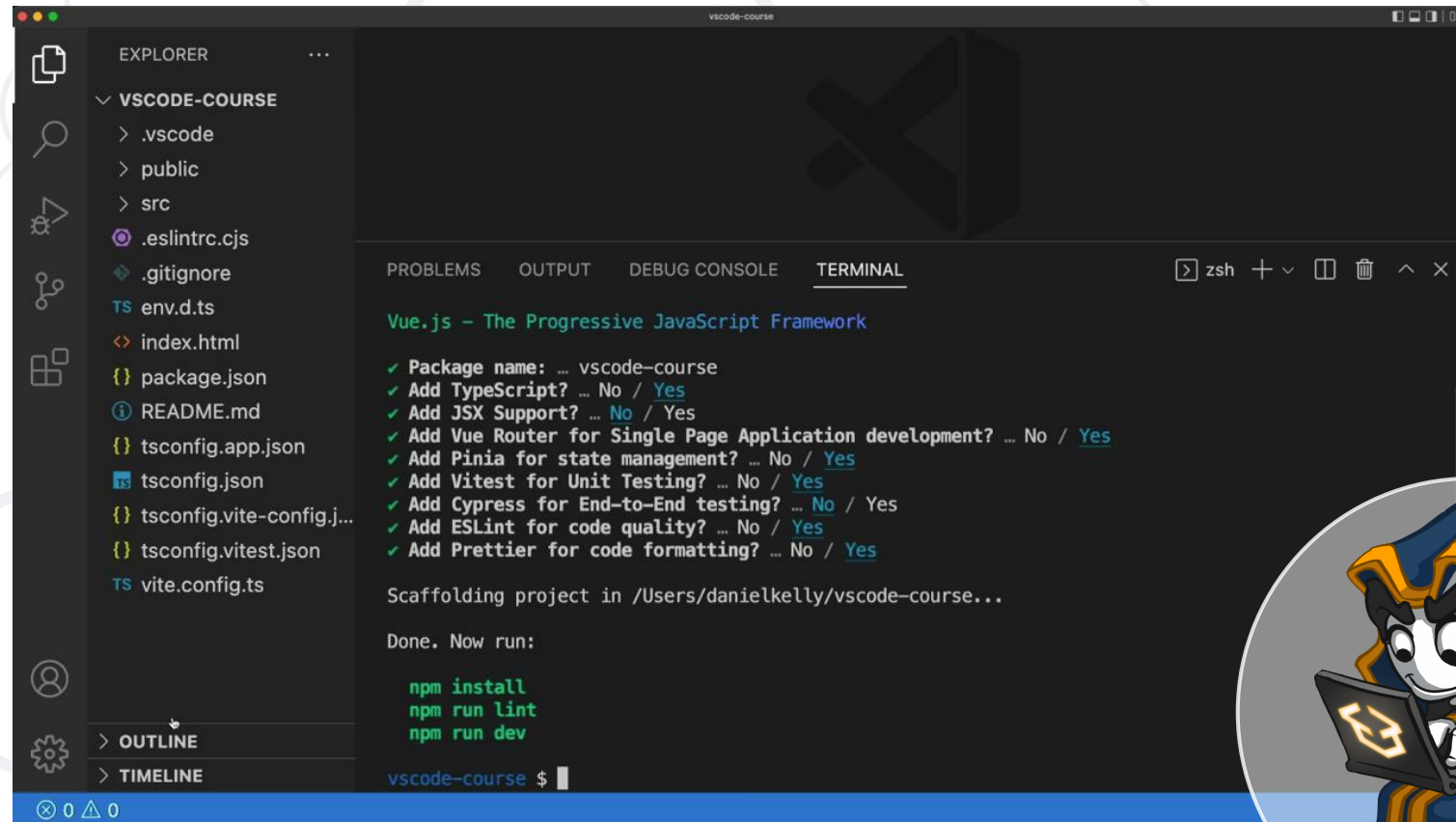
Ok to proceed? (y)

- Vue Language Features
- Optional
 - Vue VSCode Snippets
 - ESLint
 - Prettier *



Practice – Project overview

- Create a new Vue project with Vite and analyze all files and



The screenshot shows the VS Code interface with a terminal window open. The Explorer sidebar on the left shows the project structure for 'VSCODE-COURSE', including files like .vscode, public, src, .eslintrc.cjs, .gitignore, env.d.ts, index.html, package.json, README.md, tsconfig.app.json, tsconfig.json, tsconfig.vite-config.j..., tsconfig.vitest.json, and vite.config.ts. The terminal window displays the output of the 'vue create' command, showing a list of options to be selected (all 'Yes') and the scaffolding process. The prompt 'vscode-course \$' is visible at the bottom of the terminal.

```
Vue.js - The Progressive JavaScript Framework

✓ Package name: ... vscode-course
✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit Testing? ... No / Yes
✓ Add Cypress for End-to-End testing? ... No / Yes
✓ Add ESLint for code quality? ... No / Yes
✓ Add Prettier for code formatting? ... No / Yes

Scaffolding project in /Users/danielkelly/vscode-course...

Done. Now run:

  npm install
  npm run lint
  npm run dev

vscode-course $
```

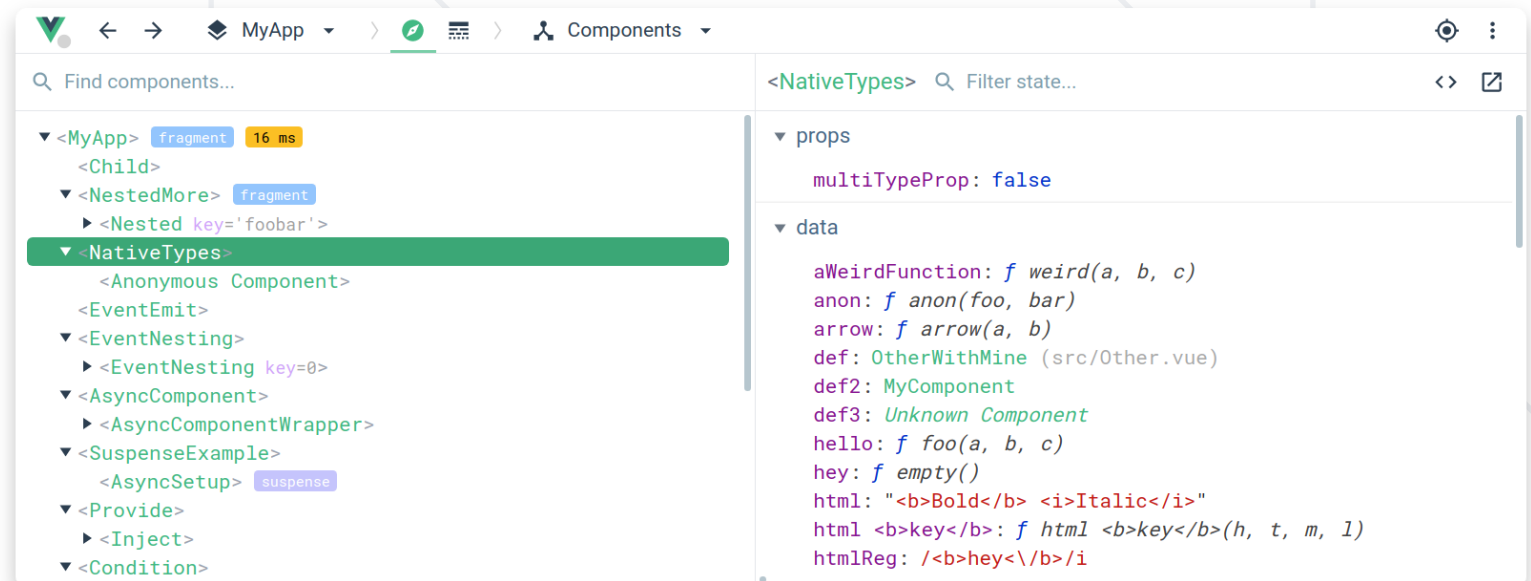


- Installation Guide -

<https://devtools.vuejs.org/guide/installation.html>

- Install on Chrome

- Install on Firefox





SFC File Explained

Template, Script, Styles

- A **special file format** that allows us to encapsulate the template, logic, and styling of a Vue component in a single file

```
vue
<script>
export default {
  data() {
    return {
      greeting: 'Hello World!'
    }
  }
}
</script>

<template>
  <p class="greeting">{{ greeting }}</p>
</template>

<style>
.greeting {
  color: red;
  font-weight: bold;
}
</style>
```


- Author modularized components using familiar HTML, CSS, and JavaScript syntax
- Colocation of inherently coupled concerns
- Component-scoped CSS
- More ergonomic syntax when working with Composition API
- More compile-time optimizations by cross-analyzing template and script
- IDE support with auto-completion and type-checking for template expressions
- Out-of-the-box Hot-Module Replacement (HMR) support



Reactivity

Declaring State and Methods

Declaring State

- We use the **data** option to declare reactive state of a component
- The option value should be a function that returns an object
- Any properties of this object are given access to from the component instance (**this** in methods and lifecycle hooks)

```
<script>
export default {
  data() {
    return {
      count: 1
    }
  },
  // `mounted` is a lifecycle hook which we will
  // explain later
  mounted() {
    // `this` refers to the component instance.
    console.log(this.count) // => 1
    // data can be mutated as well
    this.count = 2
  }
}
</script>
```

Declaring Methods

- We use the **methods** option
- This should be an object containing the desired methods

```
<script>
export default {
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      this.count++
    }
  },
  mounted() {
    // methods can be called in lifecycle
    hooks, or other methods!
    this.increment()
  }
}
</script>
```

Declaring Methods with arrow function

- Vue automatically binds the **this** value for **methods**
- Avoid using arrow functions when defining **methods**, as that prevents Vue from binding the appropriate **this** value



```
methods: {  
  increment() {  
    this.count++  
  },  
}
```



```
methods: {  
  increment: () => {  
    // BAD  
    // no `this` access here!  
  },  
}
```

- In Vue state is deeply reactive by default
- You can expect changes to be detected even when you mutate nested objects or arrays

```
export default {  
  data() {  
    return {  
      obj: {  
        nested: { count: 0 },  
        arr: ['foo', 'bar']  
      }  
    },  
    methods: {  
      mutateDeeply() {  
        // these will work as expected.  
        this.obj.nested.count++  
        this.obj.arr.push('baz')  
      }  
    }  
  }  
}
```



Template Syntax

Inserting text & HTML, directives, attribute bindings

- Text Interpolation

- Most basic form of data binding is text interpolation
- Using the "Mustache" syntax (double curly braces)

- Raw HTML

```
<span>Message: {{ msg }}</span>
```

- Double mustaches interpret the data as plain text, not HTML
- In order to output real HTML, you will need to use the **v-html** directive

```
<p>Using text interpolation: {{ rawHtml }}</p>  
<p>Using v-html directive: <span v-html="rawHtml"></span></p>
```


- Special attributes
- Expected to be single JavaScript expressions
- A directive's job is to reactively apply updates to the DOM when the value of its expression changes

```
<p v-if="isVisible">Now you see me</p>
```

- Mustaches cannot be used inside HTML attributes
- Instructs Vue to keep the element's **id** attribute in sync with the component's **dynamicId** property
- If the value is **null** or **undefined**, then the attribute will be removed from the rendered element

```
<div v-bind:id="dynamicId"></div>
```

```
// Shorthand
```

```
<div :id="dynamicId"></div>
```



CSS Features

Scoped CSS, Style bindings, Inline-Styles

- When a `<style>` tag has the `scoped` attribute, its CSS will apply to elements of the current component only

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

```
<template>
  <div class="example">hi</div>
</template>
```



```
.example[data-v-f3f3eg9] {
  color: red;
}
```

```
<div class="example" data-v-
f3f3eg9>hi</div>
```

Class Bindings – JS Objects

```
// "Inline" - in the attribute  
  
<div :class="{ active: isActive  
}"></div>  
  
<div  
  class="static"  
  :class="{ active: isActive, 'text-  
danger': hasError }"  
></div>
```

```
// As a state object  
  
<div :class="stateStyles"></div>  
  
...  
data() {  
  return {  
    stateStyles : {  
      active: true,  
      'text-danger': false  
    }  
  }  
}
```

Class Arrays & Inline Styles

// As an Array

```
<div :class="['active', 'text-danger']"></div>
```

```
<div :class="[isValid ? 'text-green' : 'text-red',  
'someOtherClass']"></div>
```

// Inline Styles

```
<p  
  style="color: black; font-size: 16px;"  
  :style="{ color: textColor, fontSize: fontSize + 'px' }"  
>  
  This is a paragraph with inline styles.  
</p>
```

Exercise – Magic 8 Ball

- Create a simple Magic 8 Ball app
 - Find and analyze the possible answers of an 8 ball
 - A button to click and start the "thinking"
 - You can just toggle a "8" and "...answer" texts
 - Think about handling the "thinking" UI – showing an icon or text
- Bonus
 - Use images for the ball or create your own with HTML and CSS
 - Add animation for shaking





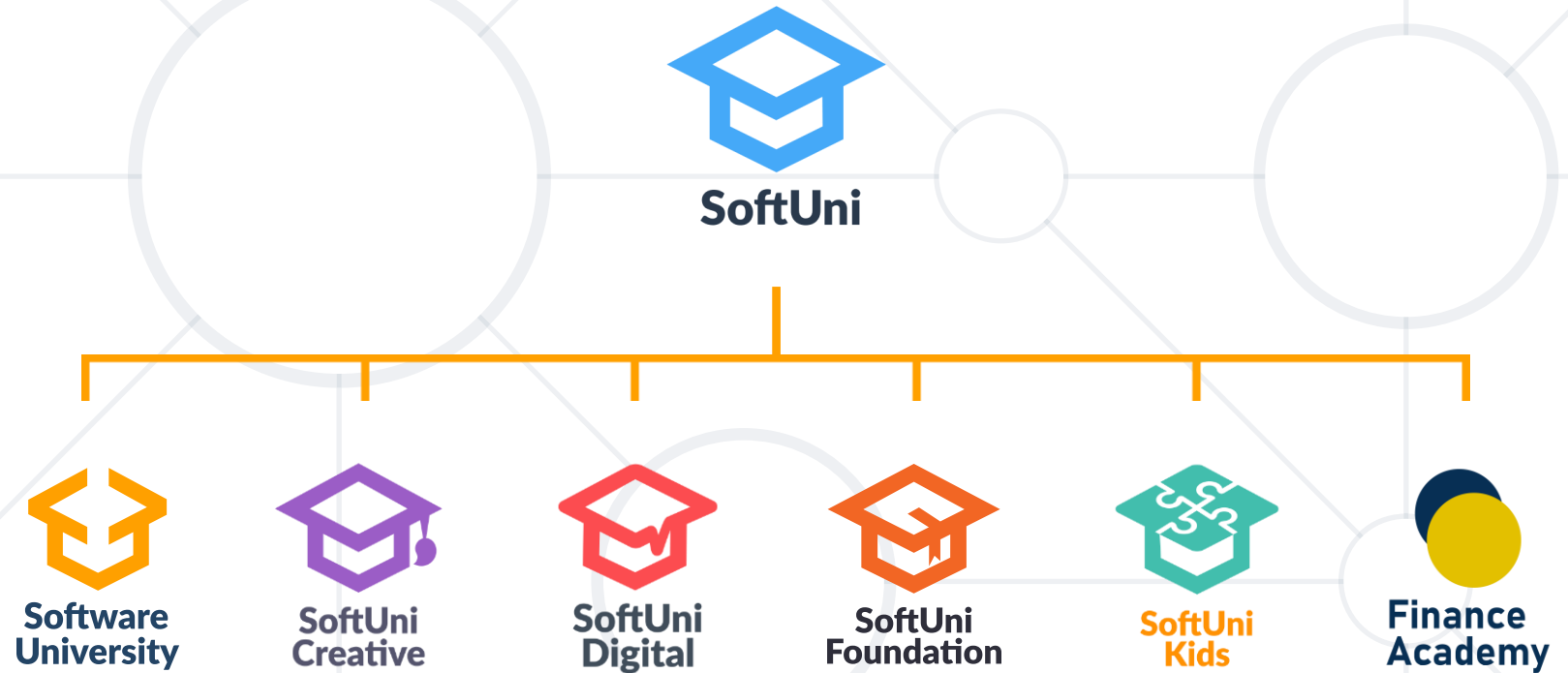
Practice

Live Exercise in Class (Lab)

- Use SFC files to create Vue instances and use it's features
- Use the **data** option to declare reactive state of a component
- Use the **methods** option to declare methods to change the reactive state



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**



**Coca-Cola HBC
Bulgaria**

 **Flutter**TM
International

INDEAVR
Serving the high achievers



AMBITIONED

 **DRAFT
KINGS**



BOSCH

 **Postbank**
Решения за твоето утре

 **PHAR
VISION**



SmartIT

DXC
TECHNOLOGY

createX

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg

