CIS 235 Rich Internet Applications - Winter 2025

W04 Lab: Conditionals, Loops, Components

(For the due date, please refer to this lab’s posting on Canvas)

Please leave the lab questions/instructions/rubrics/etc. in place. Just paste your screenshots and code below my instructions.

Purpose/knowledge/skills: Vue conditionals and loops let you customize the display of your web page based on current information. Components let you define your own vue.js elements using reusable chunks of code.

Task 1: Practice using vue.js conditional rendering (10 points)

Steps to complete the task:

* 1. Please click the link to go to [the vue.js documentation on conditional rendering](https://vuejs.org/guide/essentials/conditional.html)
  2. Click “Try it in the playground” to access your own private vue.js IDE online.
  3. Replace the code in the left pane with the following code. In this instance, feel free to copy-and-paste.

|  |
| --- |
| <script>  // [Student Name] [today’s date]  export default {  data() {  return {  awesome: true  }  }  }  </script>  <template>  <button @click="awesome = !awesome">toggle</button>  <h1 v-if="awesome">Vue is awesome!</h1>  <h1 v-else>Oh no 😢</h1>  </template> |

* 1. In the comment, replace [Student Name] and [today’s date] with your name and the date you started this lab.
  2. Click the buttons to get a feel for how this small application works.
  3. Change the default value for awesome from true to 1.
  4. Change the button action to add 1 to awesome instead of toggling it with !awesome.
  5. Change the v-if condition to display "Vue is awesome!" if awesome > 5.
  6. Add a v-else-if condition: if awesome >= 3, display "Vue is good." on the main page.
     1. The v-else condition must still work: if awesome < 3, "Oh no" should be displayed automatically.

|  |  |
| --- | --- |
| Value of awesome | Desired output |
| 6, 7, 8, 9 . . . | Vue is awesome! |
| 3, 4, 5 | Vue is good. |
| 1, 2 | Oh no 😢 |

* 1. When this works, take screenshots showing the three messages and copy-and-paste your code below the rubric.

Rubric:

* Three screenshots: 6 points total, 2 points each
* Source code logic correct: 4 points

|  |
| --- |
| Screenshot showing "Oh no": |
|  |
| Screenshot showing "Vue is good": |
|  |
| Screenshot showing "Vue is awesome!": |
|  |
| Please copy-and-paste the text of your source code here: |
| <script>      // [Praveena] [Februaru 1, 2025]      export default {          data() {              return {                  awesome: 1              }          }      }  </script>  <template>      <button @click="awesome += 1">add 1</button>      <h1 v-if="awesome > 5">Vue is awesome!</h1>      <h1 v-else-if="awesome >= 3">Vue is good.</h1>      <h1 v-else>Oh no 😢</h1>  </template> |

Task 2: Work with a component (10 points)

Steps to complete the task:

* 1. In a web browser, go to <https://www.w3schools.com/whatis/tryit.asp?filename=trywhatis_vue>
  2. Replace the source code in the left pane with the following vue 3 code. It is okay to copy-and-paste.

|  |
| --- |
| <!DOCTYPE html>  <!-- [Student Name] [today’s date] -->  <html>  <script src="https://unpkg.com/vue@3"></script>  <body>  <div id="app">  <h1> Hi! </h1>  <my-component></my-component>  <my-component></my-component>  </div>  <script>  const { createApp } = Vue  a = createApp();  a.component('my-component', {  template:  `<button @click="count++">  You clicked me {{ count }} times.  </button>` ,  data() {  return {  count: 0  }  }  });  a.mount('#app');  </script>  </body>  </html> |

* 1. Modify the page to do any two of the following:
     1. Display a message when count is bigger than 25.
     2. Use v-for to create 100 buttons, each tracking their own count.
     3. Create/increment/display a different variable, named y.
     4. Rename the component to your first name and last name in kebab-case.
  2. When complete, please take a screenshot, and copy-and-paste your code below the rubric.

Rubric:

* Two changes: 10 points total, 5 points each

Please paste a screenshot of a successful program run and copy-and-paste the text of your source code here:

|  |
| --- |
| Screenshot |
|  |
|  |
| Code text |
| <!DOCTYPE html>  <!-- [Praveena] [1 February, 2025] -->  <html>  <script src="https://unpkg.com/vue@3"></script>  <body>  <div id="app">  <h1> Hi! </h1>  <prave-ezek></prave-ezek>  <prave-ezek></prave-ezek>  </div>  <script>  const { createApp } = Vue;  const app = createApp();  app.component('prave-ezek', {  template: `  <div>  <button @click="increment">  You clicked me {{ count }} times, and y is {{ y }}.  </button>  <p v-if="count > 25">Wow, you-ve clicked more than 25 times!</p>  </div>  `,  data() {  return {  count: 0,  y: 0  };  },  methods: {  increment() {  this.count++;  this.y++;  }  }  });  app.mount('#app');  </script>  </body>  </html> |

Task 3: Q&A on vue.js props (10 points)

Steps to complete the task:

Please read the assigned reading on Prop Validation in Canvas, then answer the following questions.

* 1. What three types of capitalization does the "Prop Name Casing" section discuss?

|  |  |  |  |
| --- | --- | --- | --- |
| camelCase | kebab-case | UPERCASE | PascalCase |

* 1. Which of the following best describes a prop?

|  |  |
| --- | --- |
|  | An application-level (“global”) variable that passes data down to child components |
|  | A function that returns a String value |
|  | A JavaScript-specific workaround to variable naming issues |
|  | An alias for your component |

* 1. Can a component have more than one prop?

|  |  |
| --- | --- |
| Yes | No |

* 1. Where are props visible? Choose two.

|  |  |
| --- | --- |
|  | The main vue application |
|  | Inside the component that declares them |

* 1. 5. Which is true of types String, Array, and Date?

|  |  |
| --- | --- |
|  | They are in kebab-case |
|  | They need a single byte of storage per element |
|  | They are all "native constructors" |

Rubric:

* Five questions: 10 points total, 2 points each

Task 4: Work with events and props (20 points)

Steps to complete this task:

* 1. Open the Props-and-events.html file from this week’s lab files.   
     You can open it using File -> Open -> Web Site . . . , or open it directly in a web browser.
  2. View the file in a web browser and confirm that things work as expected.
  3. Do any two of the following. Your enhancements must use at least one of the vue.js features we covered this week.
     1. Change the counter buttons to increase the count by 2.
     2. Rename the incrementCounter() method to incC(). Make any other changes necessary so the web page once again works as expected.
     3. Connect the myNumber data object to an HTML input field that you create. Observe how the page responds to input. (V-bind may be helpful here.)
     4. Use a number input field to change the value of counter.
     5. Create an additional set of variables in createApp()/data(). Connect these variables to a new instance of my-component. Your web page will have two <my-component>s: the one in the example, and the one you create.
     6. Another enhancement of your choice, please specify in a comment near the top of the HTML file.
     7. Another enhancement of your choice, please specify in a comment near the top of the HTML file.
  4. 3. Please put a comment near the top of the file listing/describing the two enhancements you implemented.

Rubric:

* Screenshot of web browser showing changes, may require multiple screenshots: 4 points
* Enhancement 1: 8 points
* Enhancement 2: 8 points
* Comment describing enhancements: -15 points if fails

Please paste a screenshot of a successful program run and copy-and-paste the text of your source code here:

|  |
| --- |
| Screenshot |
|  |
| Code text |
| <!DOCTYPE html>  <!-- [Praveena Ezekiel] [1 February 2025] -->  <!-- Thanks to Vue.js in Action (2018) by Hanchett -->  <!-- Enhancement 1: Change the counter buttons to increase the count by 2 -->  <!-- Enhancement 2: Rename the incrementCounter() method to incC() -->  <html>  <head>  <script src="https://unpkg.com/vue@3.2"></script>  </head>  <body>  <div id="app">  <my-component :num="myNumber" :str="passedString"  :even="myNumber" :obj="passedObject"></my-component>    <p></p>  <p>Counter: **{{**counter**}}**<br /></p>  <button v-on:click="incC">Increment Counter by 2</button>  <my-button v-on:increment-me="incC"></my-button>  <input v-model="myNumber" type="number" />  </div>  <script>  const { createApp } = Vue  const app = createApp({  data() {  return {  passedString: 'Hello From Parent!',  myNumber: 43,  passedObject: { message: 'Passed Object' },  counter: 0,  // New varaibles for the second my-component  newApp: {  passedString: 'Another String!',  myNumber: 12,  passedObject: { message: 'New Object' },  counter: 0  }  }  },  methods: {  incC() { this.counter += 2; } // Incrementing by 2 as per enhancement 1  }  });  app.component('my-component', {  template: '<div>Number: {{num}}<br />String: {{str}} \  <br/>Object:{{obj.message}}</div>',  props: {  num: {  type: Number,  required: true  },  str: {  type: String,  default: "Hello World"  },  obj: {  type: Object,  default: () => {  return { message: 'Hello from object' }  }  },  // Creates JavaScript console output if value is odd  even: {  validator: (value) => {  return (value % 2 === 0)  }  }  }  });  app.component('my-button', {  template: `<div>  <button v-on:click="childIncrementCounter">Increment From Child</button>  </div>`,  methods: {  childIncrementCounter() {  this.$emit('increment-me');  }  }  });  app.mount("#app");  </script>  </body>  </html> |