Module 3

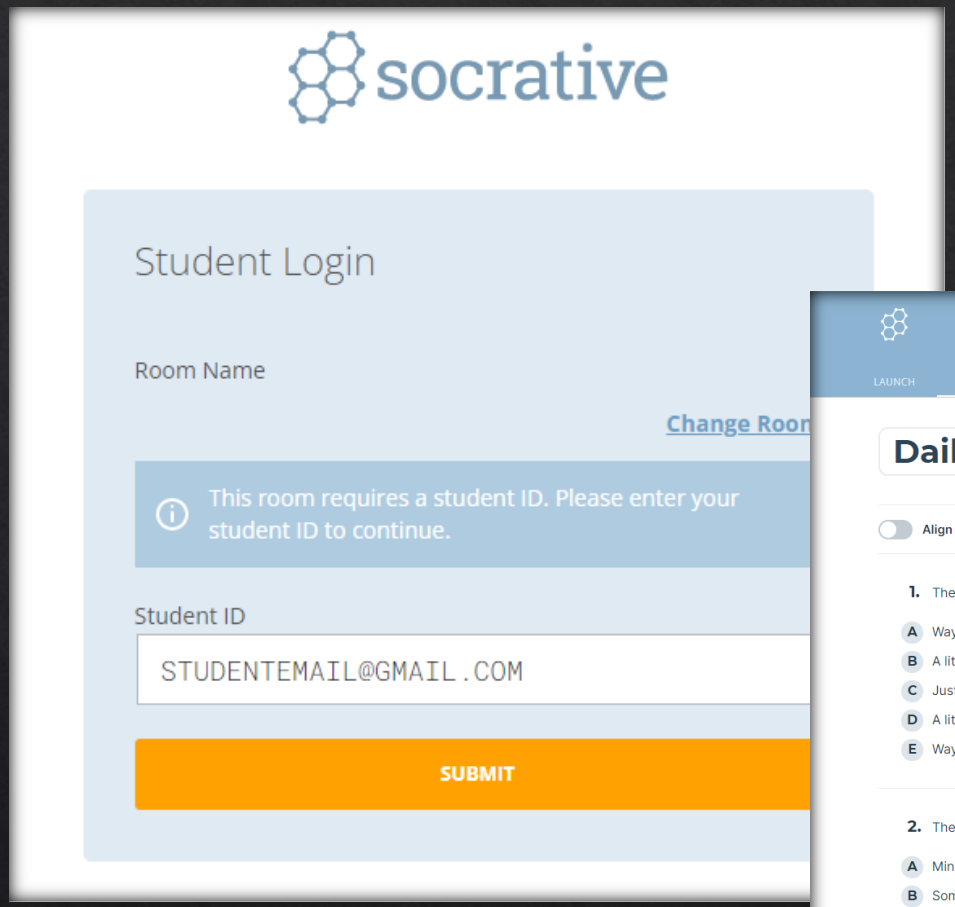
INTRO TO VUE.JS

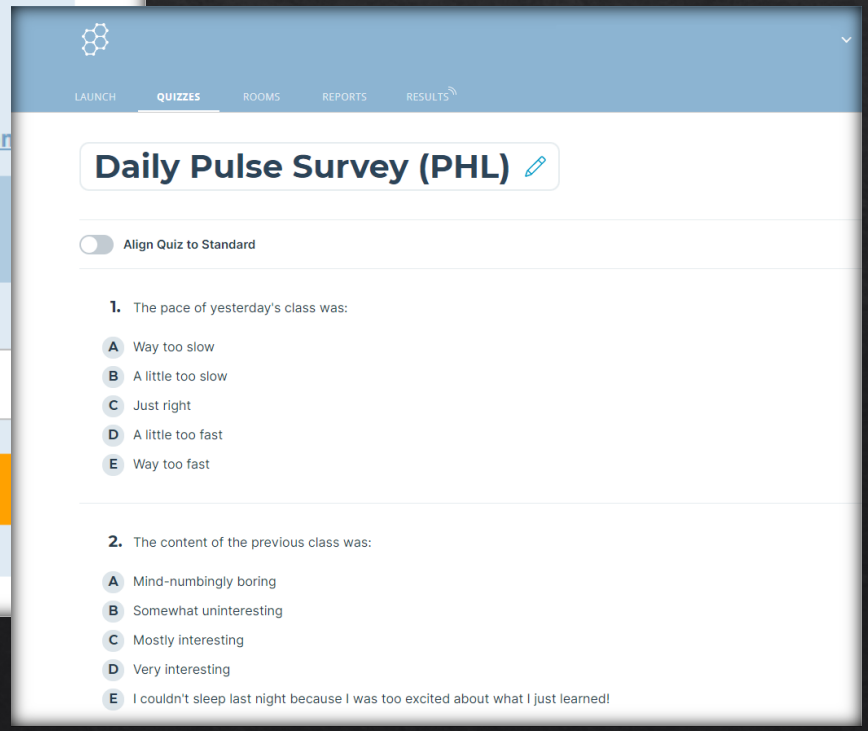
JAVA – MODULE 3, DAY 11

ELEVATE YOURSELF

PULSE SURVEY

socrative.com – Rooms PHL7JAVAPULSESURVEY

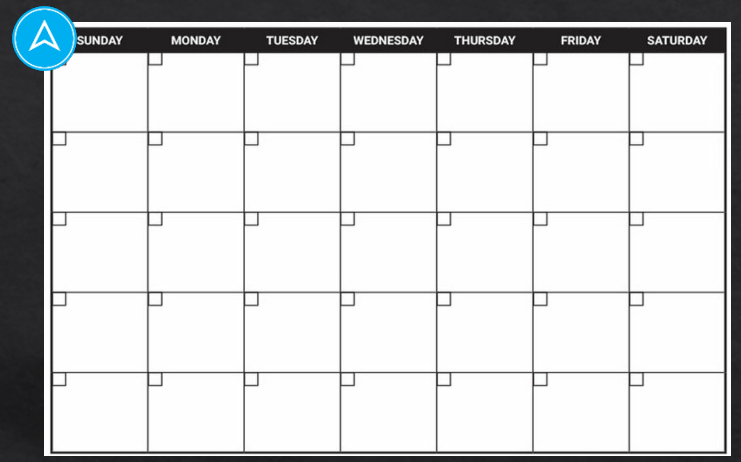




TECH ELEVATOR

Announcements

CAMPUS CALENDAR

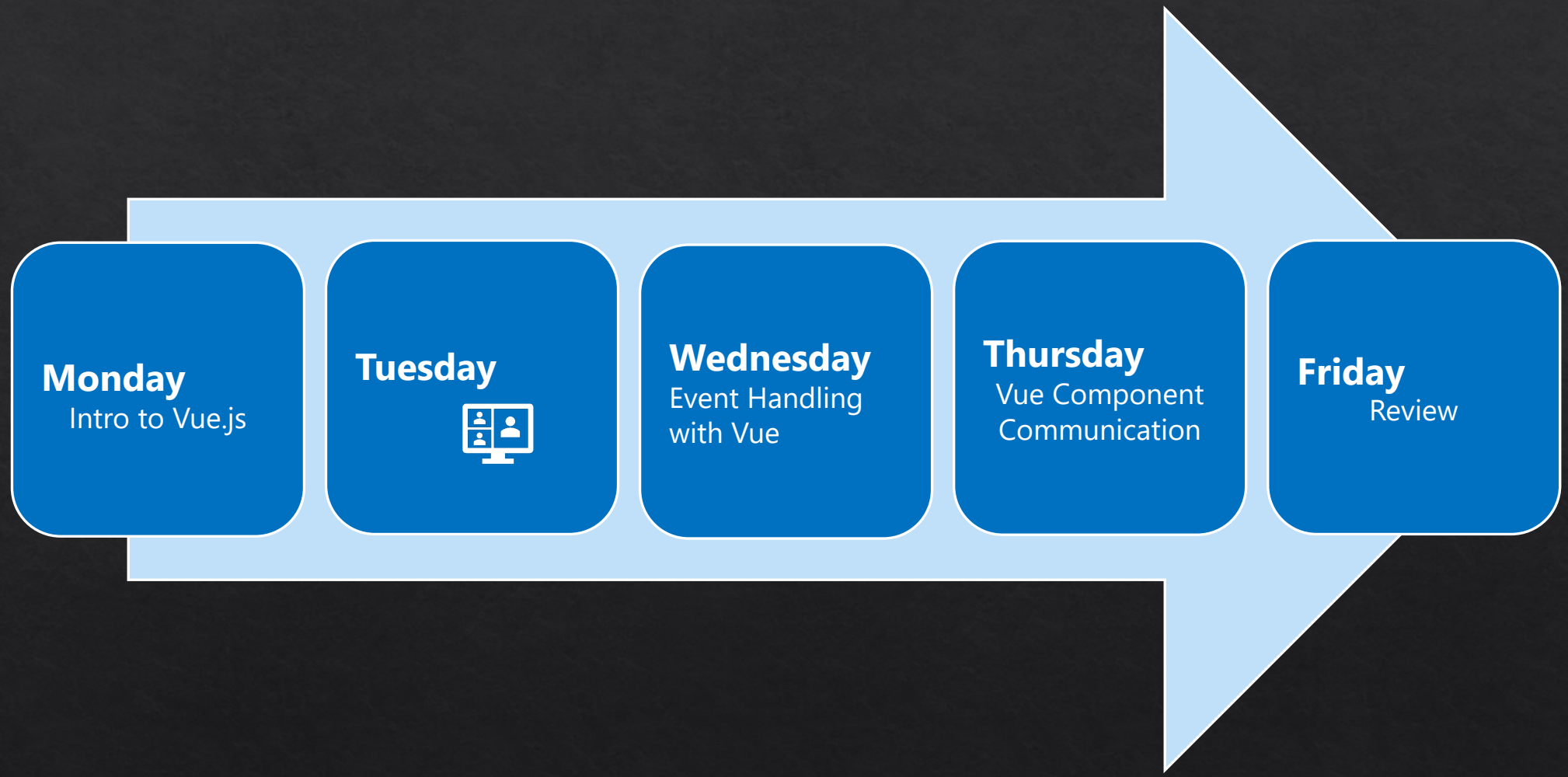


March 28th, 2022

TECH ELEVATOR

Announcements

Week 11



TECH ELEVATOR

Announcements

Week 12

Monday

Tuesday
Vue Router

Wednesday
Calling Web Services (GET)

Thursday
Calling Web Services (POST, PUT, DELETE)

Friday
Review
Capstone Kick-off/Group photo

TECH ELEVATOR

Lecture

AGENDA

- Vue CLI & NPM
- Vue Components
- Data Binding & V-Bind
- Computed Properties & V-For
- Input & V-Model

TECH ELEVATOR

Lecture

TECH ELEVATOR

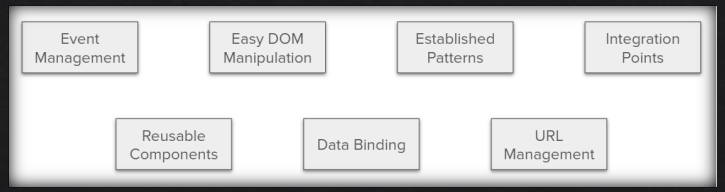
Lecture

VUE.JS

VUE IS JAVASCRIPT

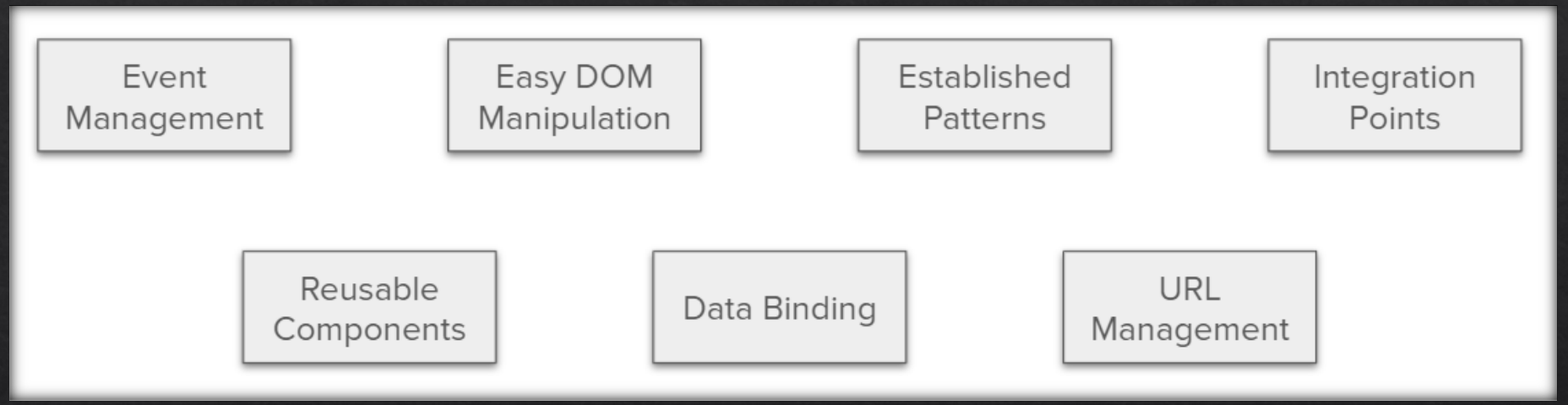
==

And More



Lecture

VUE IS A JS FRAMEWORK



Lecture

FRAMEWORKS

A **programming framework** is a set of *conventions*, *tools*, and/or *processes* that you agree to follow in order to **reap a specific benefit**, usually in the form of *higher productivity* and *built-in solutions to common problems* at the **cost** of *added complexity* in your application and its development.

TECH ELEVATOR

AUTOMAGIC!

Lecture

DIFFERENT FRAMEWORKS

Lecture

New Vue Project?

```
> vue create project-name
```

Node Package Manager (NPM)

> npm run serve

VUE CLI

Lecture

The Vue CLI

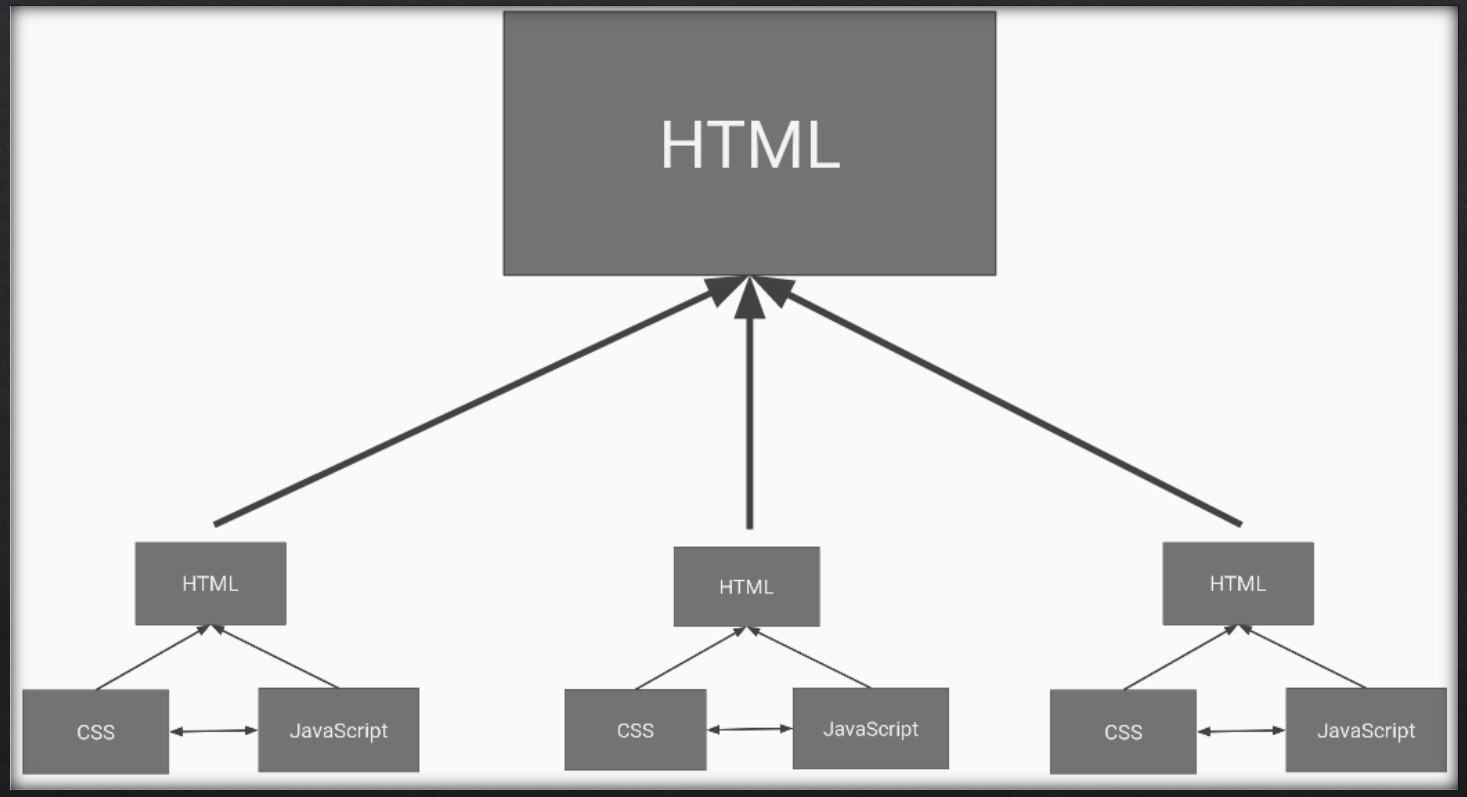
The Vue Command Line Interface, or Vue CLI, simplifies some of the work of setting up a project and creating new components.

TECH ELEVATOR

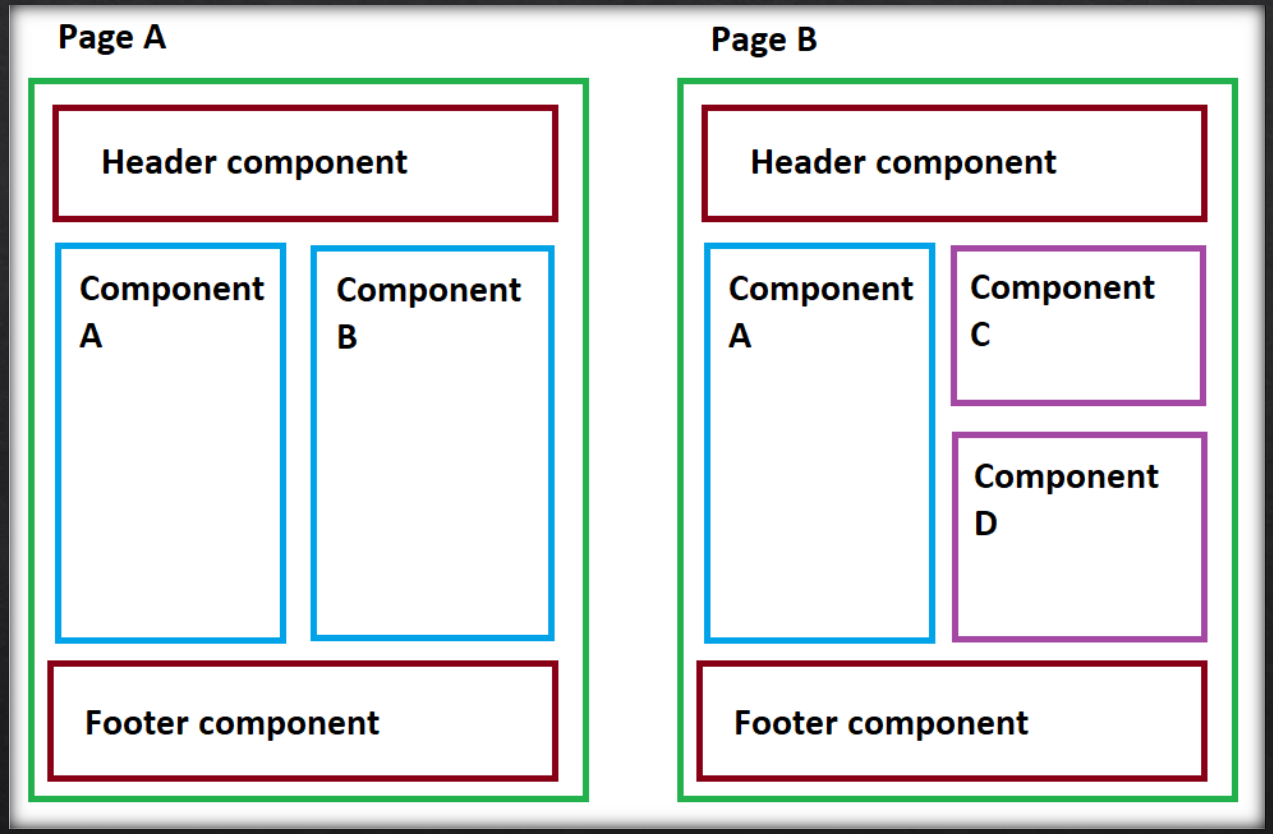
Lecture

VUE COMPONENTS

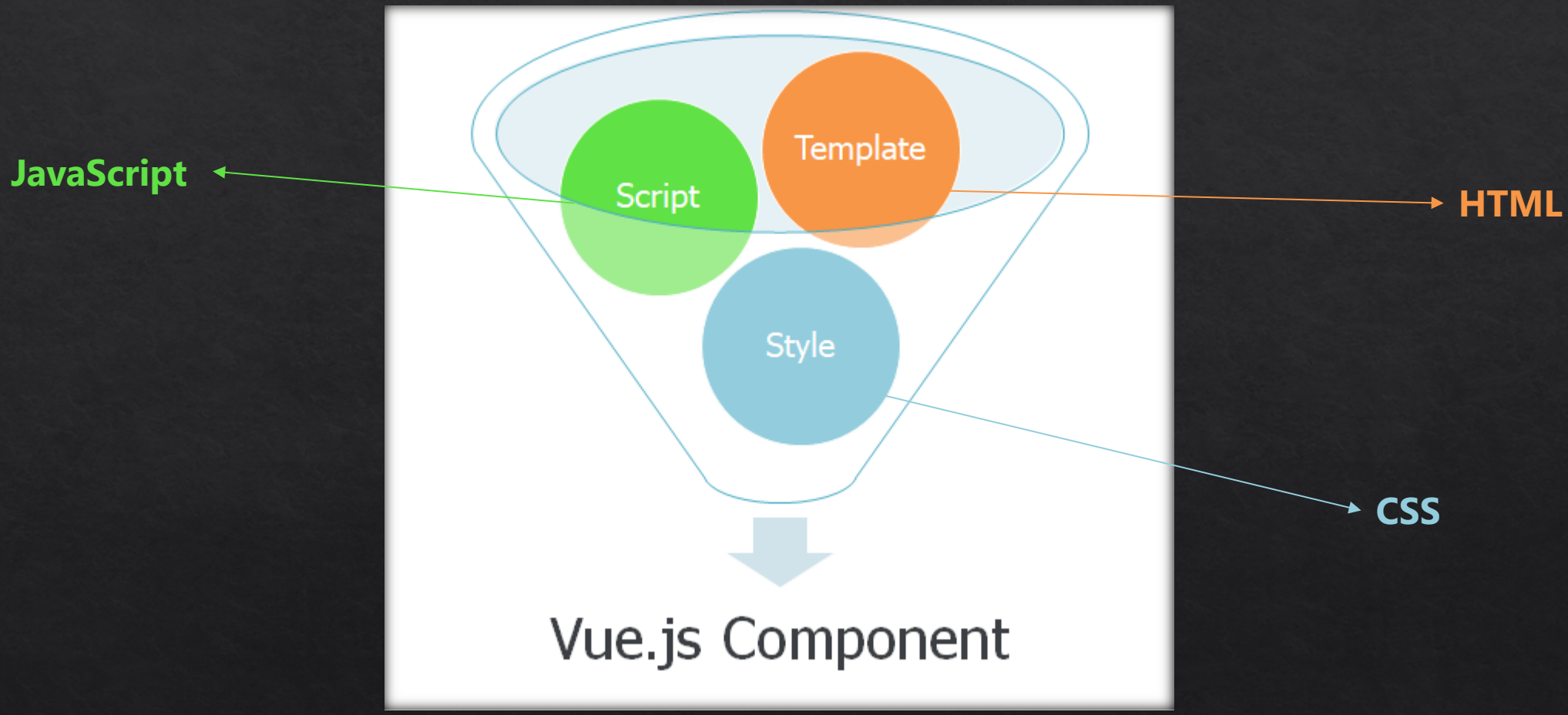
VUE COMPONENTS



VUE COMPONENTS



PARTS OF A COMPONENT



TECH ELEVATOR

TEMPLATE

Lecture

HTML

```
1 <template>
2   <main>
3     <section id="opening">
4       <h1>Aw Nuts!</h1>
5       <div>
6         <p>
7           The page you were looking for couldn't be found.
8           A dog probably ate it.
9         </p>
10      </div>
11    </section>
12  </main>
13 </template>
```

Lecture

SCRIPT

```
1 <script>
2   export default {
3     name: "MyComponentName",
4     data: () => {
5       return {
6         searchText: 'Hello Vue.js'
7       };
8     }
9   };
10 </script>
```

JavaScript

TECH ELEVATOR

Lecture

STYLE

CSS

```
<style>
  table {
    font-size: medium;
    color: #ffffff;
    text-shadow: #3b3e42 1px 1px;
    border: #8cc63f solid 0.15rem;
    border-collapse: collapse;
  }

  thead th {
    text-align: left;
    text-transform: capitalize;
    background-color: #00adee;
  }

  tbody th {
    text-align: left;
    padding-right: 0.5rem;
  }

  tbody tr:nth-child(odd) {
    background-color: #999999;
  }
  tbody tr:nth-child(even) {
    background-color: #3b3e42;
  }
</style>
```

TECH ELEVATOR

STYLE SCOPING

Lecture

```
1 <stlye>
2
3 body{
4     font-family: 'Montserrat', sans-serif;
5     background-color: whitesmoke;
6   }
7
8 </stlye>
9
```

```
1 <stlye scoped>
2
3 body{
4     font-family: 'Montserrat', sans-serif;
5     background-color: whitesmoke;
6   }
7
8 </stlye>
9
```

Lecture

package.json

```
{
  "name": "your-project-name",
  "version": "0.1.0",
  "private": true,
  <!-- anything in scripts can be run via npm run name-of-script -->
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint",
    "test:unit": "vue-cli-service test:unit"
  },
  <!-- These are included with production builds -->
  "dependencies": {
    "core-js": "^3.4.4",
    "vue": "^2.6.10"
  },
  <!-- These are only needed for development and building files -->
  "devDependencies": {
    "@vue/cli-plugin-babel": "^4.1.0",
    "@vue/cli-plugin-eslint": "^4.1.0",
    "@vue/cli-plugin-unit-mocha": "^4.1.2",
    "Even more omitted...": "0.1.1"
  },
  "otherThingsMayGoHere": {}
}

```

TECH ELEVATOR

Lecture

BUILDING COMPONENTS

TECH ELEVATOR

Lecture

DATA

TECH ELEVATOR

Lecture

DATA BINDING

```
1  <template>
2      <div id="example">
3          <p>Original message: "{{ message }}"</p>
4          <p>Computed reversed message: "{{ reversedMessage }}"</p>
5      </div>
6  </template>
7
8  <script>
9  export default{
10     data() {
11         return {
12             message: "this is the original massage",
13             reversedMessage: ""
14         }
15     };
16 };
17 </script>
```

Using the {{ dataAttribute }} syntax is called **one-way data binding**. Data from the data property is bound to the view.

NEW COMPONENT

```
<template>
    <div id="example">
        <p>Original message: "{{ message }}"</p>
        <p>Computed reversed message: "{{ reversedMessage }}"</p>
    </div>
</template>

<script>
export default{
    data() {
        return {
            message: "this is the original massage",
            reversedMessage: ""
        }
    };
};
</script>
```

Lecture

COMPUTED PROPERTIES

```
computed: {
  filteredQuestions() {
    return this.questionList.filter(q =>
      this.doesQuestionMatchSearch(q, this.searchText.toLowerCase())
    );
  },
  isBusy() {
    return this.workRemaing > 0;
  }
},
```

TECH ELEVATOR

V-BIND

Lecture

```
1  <template>
2    <main>
3      <section id="opening">
4        <h1>Search Results</h1>
5        <div>
6          <p>You searched for {{searchText}}</p>
7          <img v-bind:src="imagePath"
8               alt="A squirrel drinking red bull" />
9        </div>
10     </section>
11   </main>
12 </template>
```

V-BIND:CLASS

```
1 <article v-bind:class="{ blur: !question.isAnswerVisible,
2                          correct: question.isCorrect == true,
3                          incorrect: question.isCorrect == false}">
4     <!-- Omitted -->
5 </article>
```

V-FOR


```
1 <div class="form-check" v-for="cat of categories" v-bind:key="cat.id">
2   <input type="checkbox" v-bind:id="cat.id" v-bind:name="cat.tag" />
3   <label>{{cat.tag}}</label>
4 </div>
```

V-MODEL

```
1 <div class="form-group">
2   <label for="question">Question:</label>
3   <input id="question"
4     type="text"
5     placeholder="Enter a question name"
6     v-model="question"
7     required />
8 </div>
```

Two-way data binding
v-model on input fields
You bind a data property to an input element using an attribute on the HTML elements called **v-model**. This style of data binding is two-way data binding. You can do that with text input elements with the v-model syntax:

Lecture

V-MODEL MODIFIERS

- **v-model.number** - Converts the input to a number and stores the number
- **v-model.trim** - Trims any whitespace at the left / right of the string
- **v-model.lazy** - Waits for the input element to lose focus

TECH ELEVATOR

CHECKBOXES AND V-MODEL

Just a note when working with checkboxes.

Multiple checkboxes, bound to the same Array:

HTML

```
<input type="checkbox" id="jack" value="Jack" v-model="checkedNames">
<label for="jack">Jack</label>
<input type="checkbox" id="john" value="John" v-model="checkedNames">
<label for="john">John</label>
<input type="checkbox" id="mike" value="Mike" v-model="checkedNames">
<label for="mike">Mike</label>
<br>
<span>Checked names: {{ checkedNames }}</span>
```

JS

```
new Vue({
  el: '...',
  data: {
    checkedNames: []
  }
})
```

Jack John Mike
Checked names: ["Jack", "John"]

Lecture

ELEVATE YOURSELF

Lecture

ADDITIONAL RESOURCES

- Bootcamp OS
- Vue.js Documentation - (https://vuejs.org/v2/guide/)
- Vue CLI – (https://cli.vuejs.org/)

TECH ELEVATOR

Module 3

Introduction To Vue.js

New Tools! Let's Code!

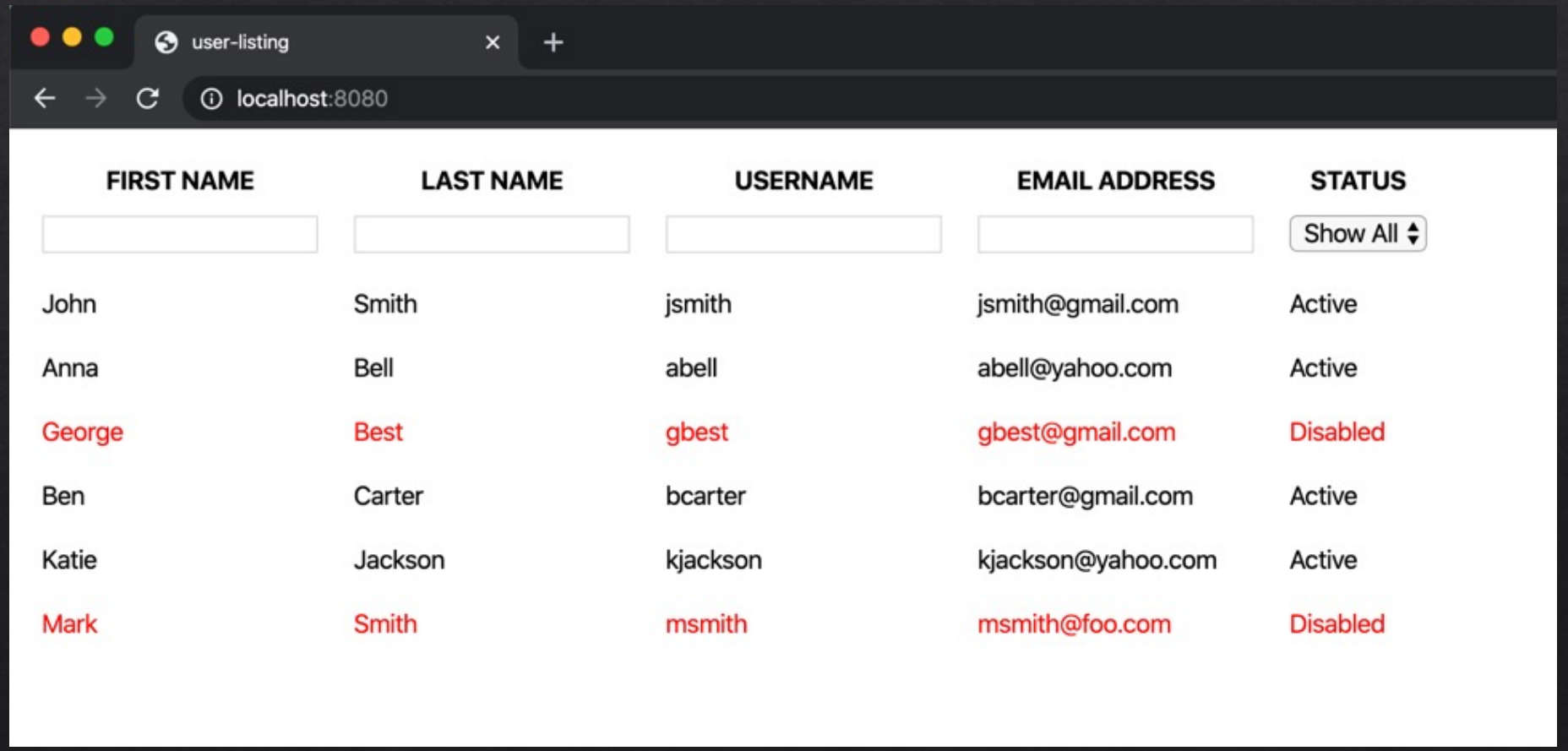
Developer's Toolbox

Exercise HW

TECH ELEVATOR

STUDENT EXERCISE

DUE WEDNESDAY 11:59PM



Exercise HW