

Module 3

INTRO TO VUE.JS

JAVA – MODULE 3, DAY 11

ELEVATE  YOURSELF





PULSE SURVEY

socrative.com – Rooms PHL7JAVAPULSESURVEY

The image shows two overlapping web pages from the Socrative platform. On the left, a 'Student Login' dialog box is displayed over a main navigation bar. The navigation bar includes icons for Launch, Quizzes (which is selected), Rooms, Reports, and Results. The 'Quizzes' section also has an 'Align Quiz to Standard' toggle and a pencil icon for editing. The main page displays a 'Daily Pulse Survey (PHL)' quiz with five questions. Question 1 asks about the pace of yesterday's class, with options A through E. Question 2 asks about the content of the previous class, with options A through E. Option E for both questions is a humorous outlier.

Student Login

Room Name: [Change Room](#)

This room requires a student ID. Please enter your student ID to continue.

Student ID: STUDENTEMAIL@GMAIL.COM

SUBMIT

Daily Pulse Survey (PHL)

1. The pace of yesterday's class was:

- A Way too slow
- B A little too slow
- C Just right
- D A little too fast
- E Way too fast

2. The content of the previous class was:

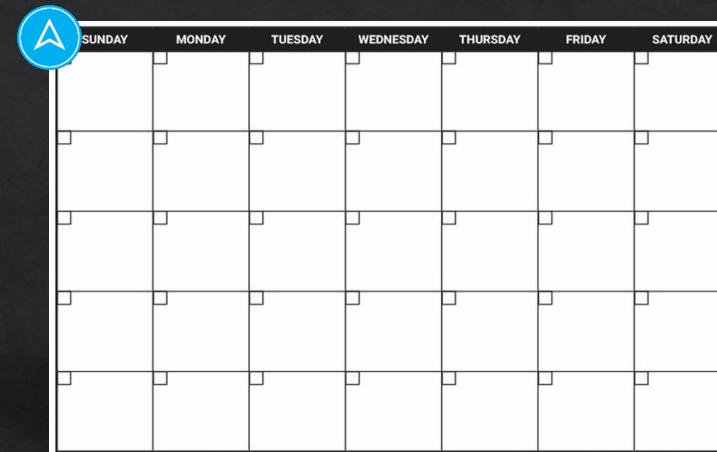
- A Mind-numbingly boring
- B Somewhat uninteresting
- C Mostly interesting
- D Very interesting
- E I couldn't sleep last night because I was too excited about what I just learned!



Announcements



CAMPUS CALENDAR



March 28th, 2022

Week 11

Announcements

**Monday**

Intro to Vue.js

Tuesday**Wednesday**Event Handling
with Vue**Thursday**Vue Component
Communication**Friday**

Review



Announcements



Week 12

Monday**Tuesday**

Vue Router

Wednesday

Calling Web Services (GET)

Thursday

Calling Web Services (POST, PUT, DELETE)

FridayReview
Capstone Kick-off/Group photo



Lecture



AGENDA



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





BOLDLY GO!



Lecture



VUE.JS



Lecture



VUE IS JAVASCRIPT



= =



And More





Lecture



Event
Management

Easy DOM
Manipulation

Established
Patterns

Integration
Points

Reusable
Components

Data Binding

URL
Management

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.

Lecture





Lecture



AUTOMAGIC!





Lecture



DIFFERENT FRAMEWORKS





Lecture



New Vue Project?

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



Lecture



Node Package Manager (NPM)



```
> npm run serve
```



Lecture



VUE CLI



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.



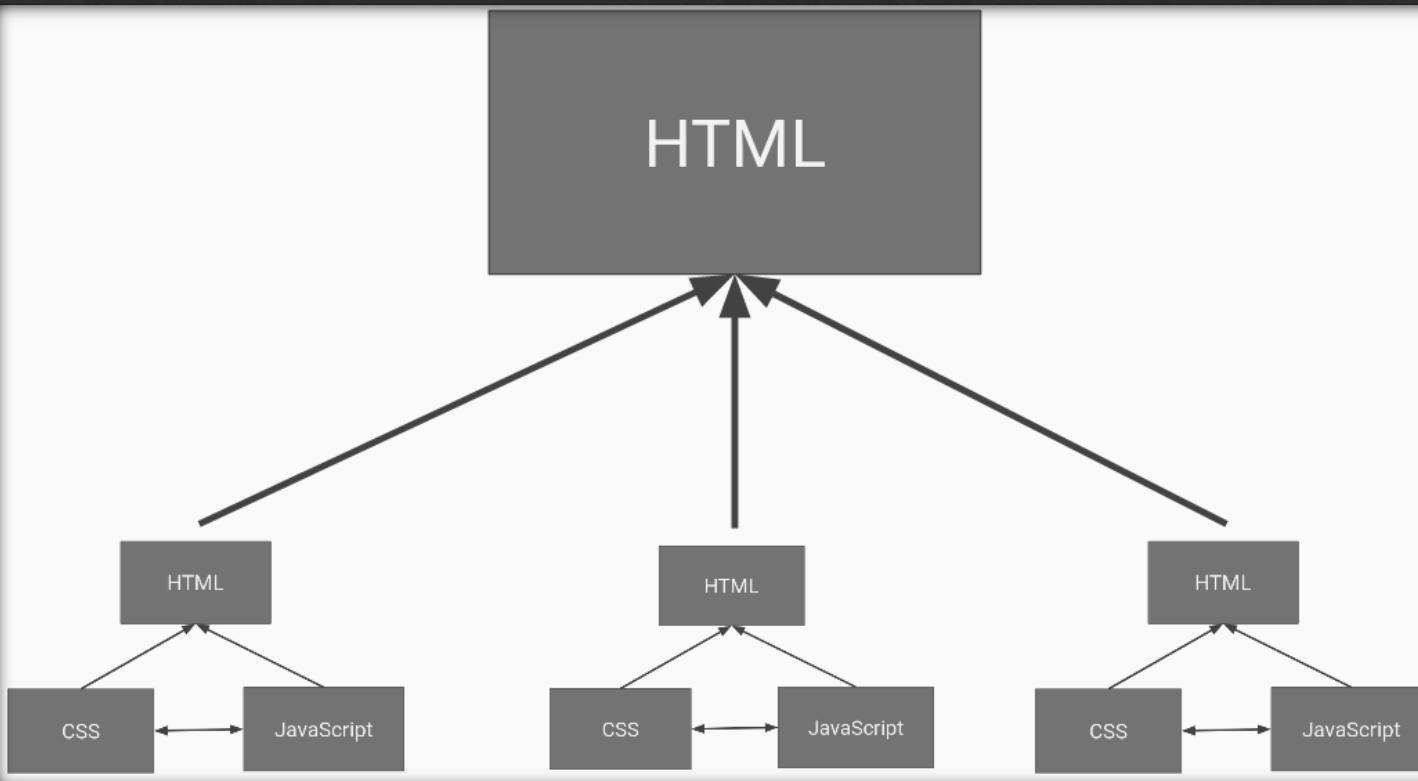
Lecture



VUE COMPONENTS



Lecture

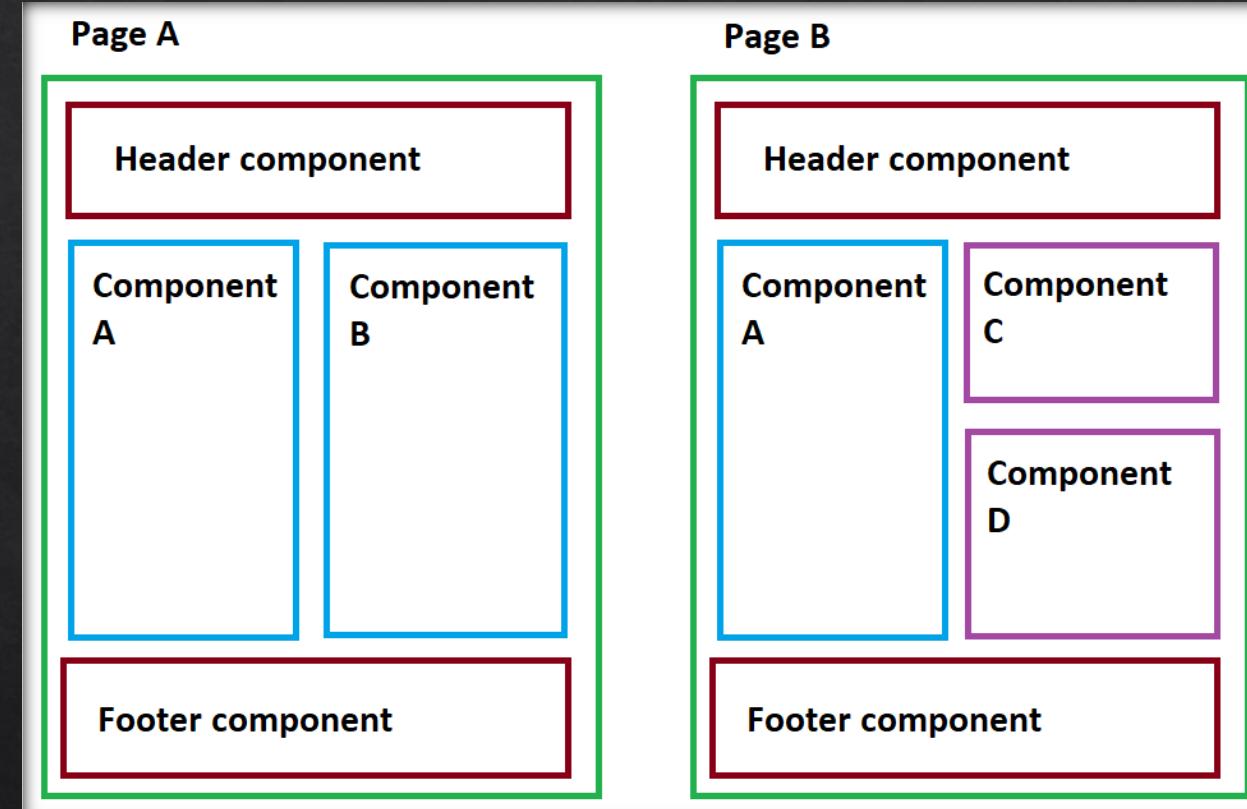




Lecture



VUE COMPONENTS





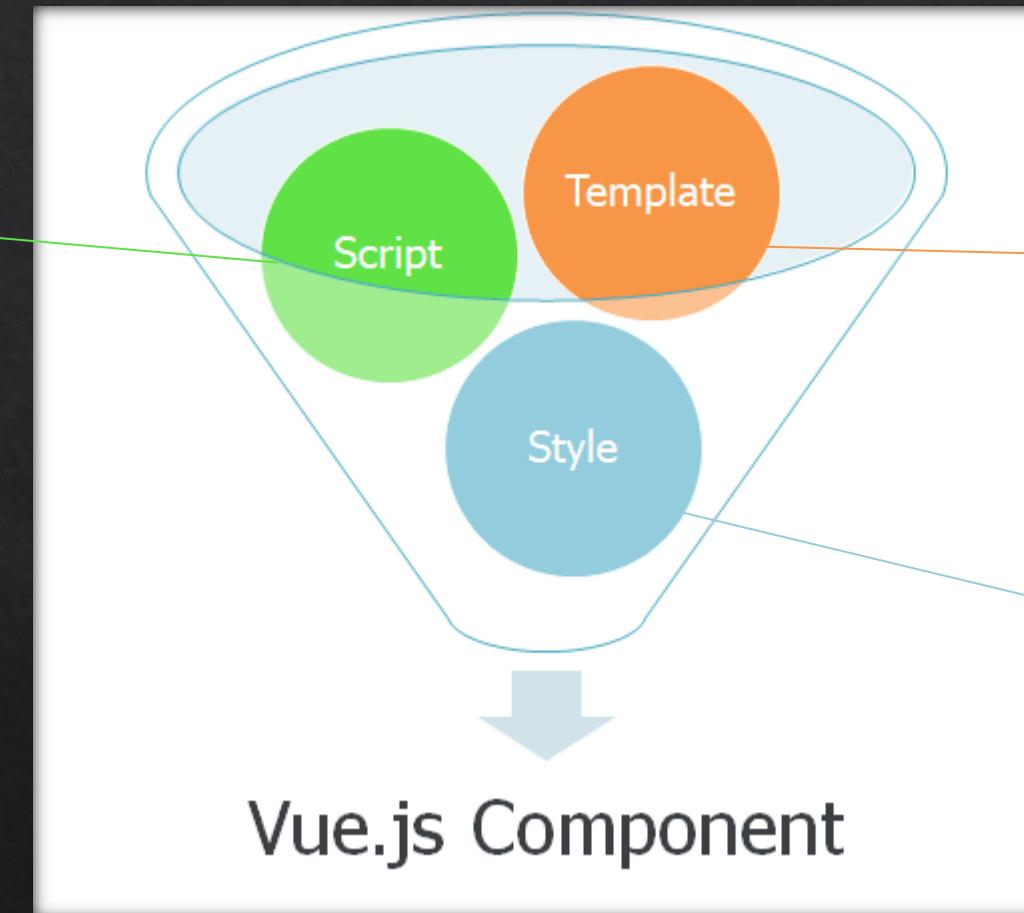
JavaScript



Lecture



PARTS OF A COMPONENT





TEMPLATE



Lecture



```
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>
```

HTML



Lecture



SCRIPT



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



Lecture



STYLE

CSS

```
1 <style>
2   table {
3     font-size: medium;
4     color: #ffffff;
5     text-shadow: #3b3e42 1px 1px;
6     border: #8cc63f solid 0.15rem;
7     border-collapse: collapse;
8   }
9
10  thead th {
11    text-align: left;
12    text-transform: capitalize;
13    background-color: #00adee;
14  }
15
16  tbody th {
17    text-align: left;
18    padding-right: 0.5rem;
19  }
20
21  tbody tr:nth-child(odd) {
22    background-color: #999999;
23  }
24  tbody tr:nth-child(even) {
25    background-color: #3b3e42;
26  }
27 </style>
```



Lecture



STYLE SCOPING

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

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



Lecture



package.json

```
1 {
2   "name": "your-project-name",
3   "version": "0.1.0",
4   "private": true,
5   <!-- anything in scripts can be run via npm run name-of-script -->
6   "scripts": {
7     "serve": "vue-cli-service serve",
8     "build": "vue-cli-service build",
9     "lint": "vue-cli-service lint",
10    "test:unit": "vue-cli-service test:unit"
11  },
12  <!-- These are included with production builds -->
13  "dependencies": {
14    "core-js": "^3.4.4",
15    "vue": "^2.6.10"
16  },
17  <!-- These are only needed for development and building files -->
18  "devDependencies": {
19    "@vue/cli-plugin-babel": "^4.1.0",
20    "@vue/cli-plugin-eslint": "^4.1.0",
21    "@vue/cli-plugin-unit-mocha": "^4.1.2",
22    "Even more omitted...": "0.1.1"
23  },
24  "otherThingsMayGoHere": {}
25 }
26
```



Lecture



BUILDING COMPONENTS



Lecture



DATA





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 message",
13       reversedMessage: ""
14     }
15   };
16 };
17 </script>
```

Lecture



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



Lecture



NEW COMPONENT

```
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 message",
13       reversedMessage: ""
14     }
15   };
16 };
17 </script>
```



Lecture



```
1 computed: {
2   filteredQuestions() {
3     return this.questionList.filter(q =>
4       this.doesQuestionMatchSearch(q, this.searchText.toLowerCase())
5     );
6   },
7   isBusy() {
8     return this.workRemaining > 0;
9   }
10 },
```



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         
9       </div>
10      </section>
11    </main>
12 </template>
```



Lecture



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>
```



Lecture



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>
```



Lecture



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.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





Lecture



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>
```

HTML

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

JS

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



ADDITIONAL RESOURCES

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

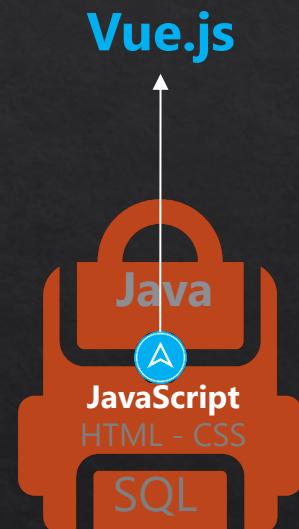
Lecture



Module 3

Introduction To Vue.js

New Tools! Let's Code!



Developer's Toolbox



Exercise HW



STUDENT EXERCISE

DUE WEDNESDAY 11:59PM

A screenshot of a web browser window titled "user-listing" with the URL "localhost:8080". The browser has standard OS X-style window controls (red, yellow, green buttons) and a toolbar with back, forward, and refresh buttons.

FIRST NAME	LAST NAME	USERNAME	EMAIL ADDRESS	STATUS
John	Smith	jsmith	jsmith@gmail.com	Active
Anna	Bell	abell	abell@yahoo.com	Active
George	Best	gbest	gbest@gmail.com	Disabled
Ben	Carter	bcarter	bcarter@gmail.com	Active
Katie	Jackson	kjackson	kjackson@yahoo.com	Active
Mark	Smith	msmith	msmith@foo.com	Disabled