# CS2005: Lab 5

## Vue

### Intro to frontend JS framework

# Objective

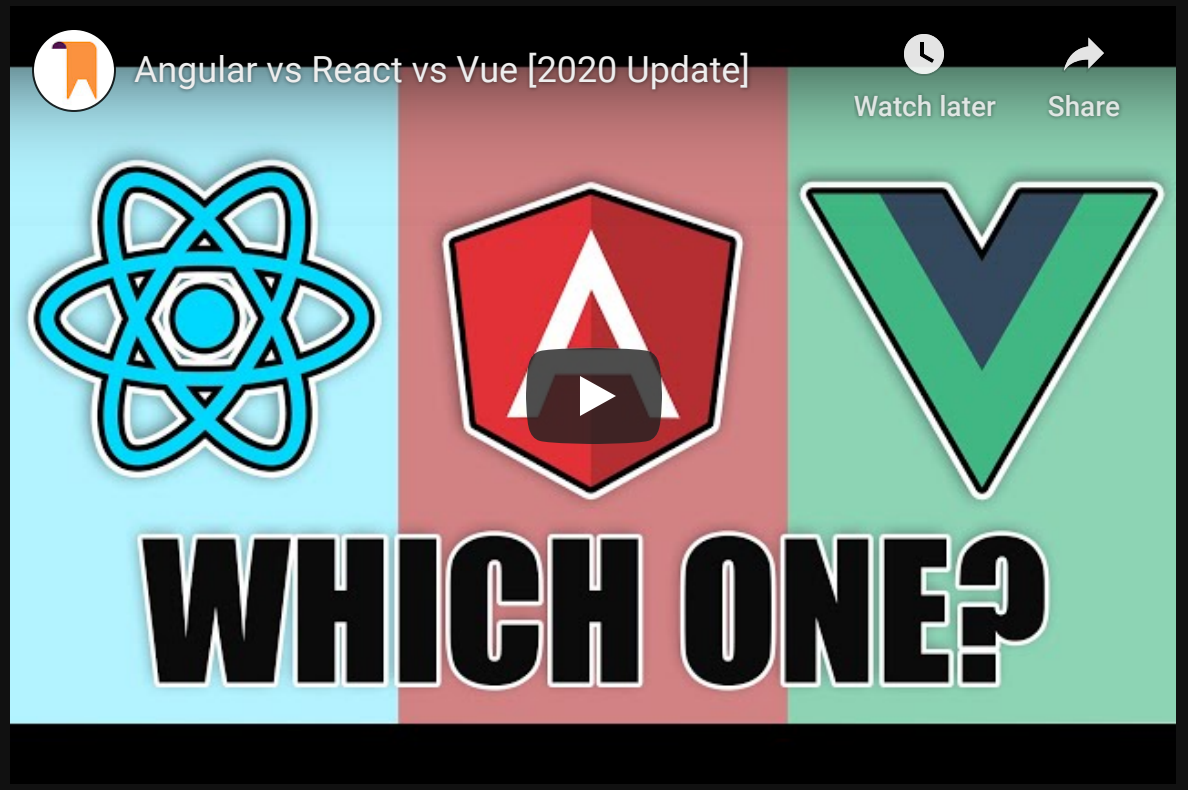
Learn why JS frameworks are popular and used in modern web-development.

JS frameworks typically help with two issues: **reactivity** and **componentization**.

**Reactivity** is the ability of the user interface to update when the application state has changed, and also vice versa.

**Compartmentalization** is how to modularize user interaction interface into components.

# Why Vue?




Pros: It is easier to learn than Angular or React.

Cons: It is less popular than React, especially in terms of jobs (2020).

Personal bias: I like Vue's progressive philiosophy better.

# Vue: setup

There are many ways of installing Vue, but the simplest (for learning) is just embed it in a script tag.

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

## Other useful tools for development:

- Get a syntax text-editor mark-up for Vue (e.g. Vetur for VSC)
- Setup Chrome Vue.js devtools - for debugging

# Vue: Simple template

Let's start with a very simple example template.

"Hello World" is a classic but not very interesting and doesn't show off reactivity very well.

So let's modify this a bit...

More reading: Template syntax

# Vue: reactivity

...And showcase the power of 2-way data binding.

Change the model, and the view updates. Change the view, and the model updates.

A lot of magic for very little code.

Here's the updated example.

More reading: Vue reactivity in depth

# Vue: Computed, watched properties

In-template expressions are very convenient, but they are meant for simple operations.

Putting too much logic in your templates can make them bloated and hard to maintain.

Therefore the idea behind **computed** properties.

More reading: Computed and watched properties

# Vue: Event handling - Methods

As your events get more complicated, you will often call a **method** for handling the event instead of putting all the logic in the template.

More reading: Events and methods

# Vue: Virtual DOM

In a large application, the DOM tree can be huge.

Manipulating this is expensive. You can use a JSON structure to represent DOM nodes instead

This **virtual DOM** is faster, and how Vue tackles manipulations under the hood. (similar to React. Angular uses its own change detection mechanism)

More reading: Render functions and virtual DOM

# Vue: Lifecycle hooks

## Vue Lifecycle

More reading: Vue Instance

The lifecycle hook most often used is on mounted. This is when the binding is done with the virtual DOM and everything is setup.

# Assignment

## Lab 5: Event form submission

# Assignment Lab 5: Setup

To be completed by **2-11-2020**(Mon) 12pm

- On GitHub Pages, create a repo **CSC2005Lab5**
- This repo link will be of the format https://**<username>**.github.io/CSC2005Lab3/
- Create a basic index.html file for your final solution.
- You can put your CSS style tags, HTML and JS script code into one file, or organize them into files/directories.

# Assignment Lab 5: Problem

- Remember your assignment from Lab 4?
- Now you actually have to do something with the form.
- You're going to use Vue to add reactivity to your project.
- And then when you "submit" the content you are going to print out the result on **console.log**, or display on the frontend somewhere.
- (We're assuming that you will write AJAX code to submit to some backend service)

# Assignment Lab 5: Extra Challenges

- Actually write AJAX code to submit the data to a web service.
- For example, write to a Google spreadsheet.
- Or use a free API mocker (mocky.io, postman, etc.) to simulate the response process.
- Or use a simple freemium backend (restdb.io). Scale up all the way to bigger backend-as-a-service.
- Or design and create your own API backend.

# Assignment Lab 5: Here's a template

- You can 100% use your own form template if you do not want to use mine.
- Here's the basic starter template code.
- Fill in the rest, and also the methods for submission.

# Questions?

Chi-Loong | V/R