PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

LESSON 8:
Vue Expanded

HTML 300

OVERVIEW

1. Front-end Framework Comparison
2. Components & Props
3. Component Activity
4. Directives & Modifiers
5. Filters & Mixins
6. Slots

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Front-End Frameworks Comparisons

FRONT-END FRAMEWORKS

- The main three front-end frameworks used today are:
  - ReactJS
  - Angular
  - VueJS
- These frameworks have different pros/cons and use cases
- Generally, projects can be built using any of them (unless looking for specific tool/library based on one of them)
- Let's take a look at some of the other frameworks before jumping back into Vue.

FRONT-END FRAMWORKS

- Angular
  - Built by Google
  - Rebuilt after version 1, so Angular >= 2 is different than the original AngularJS
  - Newer Angular is built using TypeScript, a statically-typed superset language of JavaScript that compiles into JavaScript
  - Overtook backbone/ember as the main front-end framework in the early 2010s
  - Typically associated with enterprise and large corporate work (but can be used for anything)
  - Opinionated, prefers things done ‘the angular way’

FRONT-END FRAMEWORKS

- React
  - Built by Facebook
  - Pioneered the concept of the ‘virtual dom’, where only the app will know to only update components whose states have changed
  - The state concept can be managed with libraries like Redux.
  - React rapidly overtook Angular as the most popular front-end framework
  - Typically associated with startups, but is used across the spectrum
  - Heavy use of external libraries/components to expand core functionality

FRONT-END FRAMEWORKS

- Vue
  - Built by folks that wanted to combine the best aspects of the existing frameworks into a new one
  - Also uses the concept of state and the virtual DOM
  - State concept can be managed with libraries like Vuex
  - Rapidly increasing popularity, especially in places like China that don't want to adhere to React's (Facebook's) licensing.
  - Typically associated with startups and as the default JS framework for the PHP framework Laravel

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Components & Props

COMPONENTS

- We've gone over Vue's HTML-based templates that bind the Vue instance to the DOM; we can expand that into building out components.
- A Vue component is a collection of elements that are grouped together and can be accessed through one element.
- See the example below, the 'callout' component would be comprised of the elements on the left.

```
<article>
  <h2></h2>
  <div></div>
  <p></p>
</article>
```

```
<callout></callout>
```

PROPS & PROP TYPES

- Props
  - Vue uses the concept of props to pass data down from parent to child.
  - Props are intended for one way communication.
  - V-bind is utilized to dynamically bind props to data on the parent.
- Prop types and validation
  - Props can be defined with a type (String, Boolean, Number, etc.).
  - Props can be required by passing true to the required key in the props object.
  - Props can have a default value given as well.

PROPS & PROP TYPES

- Prop declaration on a component

```
Vue.component('blog-post', {
  props: ['title'],
  template: '<h1>{{ title }}</h1>'
})
```

- Component usage in the template

```
<blog-post title="My post 1"></blog-post>
<blog-post title="My post 2"></blog-post>
<blog-post title="My post 3"></blog-post>
```

- Resulting HTML output

```
<h1>My post 1</h1>
<h1>My post 2</h1>
<h1>My post 3</h1>
```

COMPONENTS

```
props: {
    text: {
      type: String,
      required: true,
      default: 'Hello there!'
    }
  },
```

- Why validation?
  - By specifying the data type for the prop, we can be sure that the expecting data won't cause issues due to the wrong type applied
  - Easier to catch mistakes or accidental data passing
  - Give users nice defaults for interactive data

- Components can have have both static content and dynamic props within templates, using dynamic props as needed
- Let's create a component that utilizes its parent's props

COMPONENTS

- When this runs, there will be 2 <p> within the #app div, both with the ‘message’ content found in the parent’s data.
- Note the ‘:text’ is a shorthand for v-bind:, and that the props value for the child component matches the binds in the HTML.

```
Vue.component('child', {
    props: ['text'],
    template: `<p>{{ text }}</p>`
  });

  new Vue({
    el: "#app",
    data() {
      return {
        message: 'Hello there!'
      }
    }
});
```

```
<div id="app">
    <child :text="message"></child>
    <child :text="message"></child>
</div>
```

COMPONENTS

- The end result will be 2 <p> within the #app div, both with the ‘message’ content in the data.
- Note the ‘:text’ is a shorthand for v-bind:

```
Vue.component('child', {
    props: ['text'],
    template: `<p>{{ text }}</p>`
  });

  new Vue({
    el: "#app",
    data() {
      return {
        message: 'Hello there!'
      }
    }
});
```

```
<div id="app">
    <child :text="message"></child>
    <child :text="message"></child>
</div>
```

COMPONENTS

- Components can also use a mix of instance data, static, as well as the dynamically passed props for data
- Note the ':text' is a shorthand for v-bind:

```
Vue.component('child', {
    props: ['text'],
    template: `<p>{{ text }}</p>`
  });

  new Vue({
    el: "#app",
    data() {
      return {
        message: 'Hello there!'
      }
    }
});
```

```
<div id="app">
    <child :text="message"></child>
    <child :text="message"></child>
</div>
```

COMPONENTS

- Components can use instance data, static values, or data passing through props.
- @ is a shortcut syntax for v-on

```
<div id="app">
  <button @click=”plus”>+</button>
  <button @click=”minus”>-</button>
  <h2>This is the app data: <span class="num">{{ count }}</
span></h2>
  <child count="1"></child>
  <p>Child component instance that is using a value as props</p>
  <child :count="count"></child>
  <p>Same child component, but is using the vue instance data as
props</p>
</div>
```

```
Vue.component('child', {
  props: {
    count: {
      type: Number,
      required: true
    }
  },
  template: `<div class="num">{{ count }}</div>`
})
new Vue({
  el: '#app',
  data() {
    return {
      count: 0
    }
  },
  methods: {
    plus() {
      this.count++;
    },
    minus() {
      this.count--;
    }
  }
})
```

COMPONENTS

- In this case, child count="10" is using the a static value, where child :count="count" is dynamically binding to its parent's data.
- Note, each component instance has its own isolated scope.

```
<div id="app">
  <button @click="increment">+</button>
  <button @click="decrement">-</button>
  <h2>This is the app data: <span class="num">{{ count }}</span></h2>
  <child count="10"></child>
  <p>Child component instance that is using a value as props</p>
  <child :count="count"></child>
  <p>Same child component, but is using the Vue instance data as props</p>
</div>
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Let's Try

COMPONENT ACTIVITY

- Let’s give it a try.
  - Clone the activity repo from the UW Front-End Git Hub
  - In your terminal, run `npm install`.
  - Make sure your packages installed successfully, flag if you need assistance.
  - Now run `npm run dev` in your terminal to launch webpack.
  - Given the Vue files in the repo, convert the blog post to a child component.
  - Also add some props to the parent to pass down, and display it within the child component.

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Directives & Modifiers

DIRECTIVES

- Directives
  - Directives are Vue's shortcuts to interaction and functionality within components and templates.
  - We've been utilizing directives so far within our components and templates.
  - Directives include: v-text, v-html, v-show, v-if, v-else, v-else-if, v-for, v-on, v-bind, v-model, v-pre, v-cloak, v-once
  - Check out the links to each to see the documentation, a great resource for looking deeper into Vue.

MODIFIERS

- Modifiers
  - Modifiers give you access to extra functionality on top of v-model
- V-model.trim
  - Strips any whitespace from the start/end of bound string
- V-model.number
  - Forces strings to numbers
- V-model.lazy
  - Will only populate content on changes events rather than input

DIRECTIVES

- Multiple Bindings
  - If a component has multiple events on a single directive that would like to be listened to, they can be added in a comma separated list.
- Ternaries
  - Ternaries can be used directly within directives to accommodate logic.

```
<div v-on="
  click   : onClick,
  keyup   : onKeyup,
  keydown : onKeydown
">
</div>
```

```
<div id="app">
    <div class="item">
        <button @click="counter > 0 ? counter -= 1 : 0">-</button>
        <span>Quantity: {{ counter }}</span>
        <button @click="counter += 1">+</button>
    </div>
</div>
```

DIRECTIVES

- Custom Directives
  - What if you needed some functionality that wasn't readily available in one of the pre-built directives within Vue?
  - You can build your own custom directives.
  - Custom directives have access to the following hooks:
    - **bind** - This occurs once the directive is attached to the element.
    - **inserted** - This hook occurs once the element is inserted into the parent DOM.
    - **update** - This hook is called when the element updates, but children haven't been updated yet.
    - **componentUpdated** - This hook is called once the component *and* the children have been updated.
    - **unbind** - This hook is called once the directive is removed.

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Filters & Mixins

FILTERS

- Filters
  - Despite the name, filters aren't filtering the data but acting more as a presentational tool
    - Formatting dates
    - Formatting currency
  - The filter syntax within the templating is a pipe bar | followed by the name of the filter
  - Filters can be defined globally (used by any component), or locally – similar to a method

```
// Global Filter
Vue.filter(myFilter, function(val) {
  return // do stuff
});

// Local Filter
filters: {
  myFilter(val) {
    return // do stuff
}
```

FILTERS

- Using Filters
  - Once the filter has been defined, you can pass through data via the templating, by using a pipebar '|' followed by the name of the filter.

```
<p>{{ data | myFilter }}</p>
```

- Chaining Filters
  - You can apply multiple filters by chaining the pipebar syntax.

```
<p>{{ data | filter1 | filter2 }}</p>
```

FILTERS

- Arguments
  - Since filters are functions, optional arguments can be passed along through the filter declaration within parenthesis

```
<p>{{ data | myFilter(arg1, arg2) }}</p>
```

  - Within the filter function itself, the arguments will be passed in order of the template args passed, with the filter value being the first value.

```
filters: {
  filterName(value, arg1, arg2) {
    return // do stuff
  }
}
```

MIXINS

- Mixins
  - Mixins are functions (with data and/or methods) that encapsulate some sort of functionality
  - These tools are good for sharing the same functionality across multiple components without writing it over and over
  - Mixins are defined within the component declaration as an array of values mapped to the mixin's name
  - The main concept with mixins is making a 'pure function', one that doesn't modify outside of its scope so its output will always be predictable given the input on every execution.

MIXINS

- Declaring a Mixin
  - Mixin declaration is similar to regular methods on components

```
const showHide = {
  data() {
    return {
      isHidden: false
    }
  },
  methods: {
    toggleShowHide() {
      this.isHidden = !this.isHidden;
    }
  }
}
```

MIXINS

- Using a Mixin
  - When using a mixin within a component declaration, you can add an array of mixins that the component should have access to use.
  - In this instance, we are using the showHide mixin across these components.

```
const Header = {
  template: '#header',
  mixins: [showHide]
};

const Nav = {
  template: '#nav',
  mixins: [showHide]
};
```

PROFESSIONAL & CONTINUING EDUCATION

UNIVERSITY *of* WASHINGTON

Slots

SLOTS

- Slots
  - Slots are a special ‘placeholder’ like element
  - The syntax is <slot></slot>
  - Slots are used within the templating to denote some content that won’t be passed through props or components, but within the actual markup used within that component.

```
<div id="app">
  <h1>Slots!</h1>
  <app-example>
    <h2>Slot 1</h2>
  </app-example>
  <app-example>
    <h2>Slot 2</h2>
    <p>Slots can have as many tags as you'd like</p>
  </app-example>
</div>
```

```
<script type="text/x-template" id="example">
  <div class="example">
    <slot></slot>
    <p>Here is my static content below the slot</p>
  </div>
</script>
```

SLOTS

- Slots
  - The resulting output would return like this:

```
<div class="example">
  <h2>Slot 1</h2>
  <p>Here is my static content below the slot</p>
</div>
<div class="example">
  <h2>Slot 2</h2>
  <p>Slots can have as many tags as you'd like</p>
  <p>Here is my static content below the slot</p>
</div>
```

SLOTS

- Defaults
  - Slots can also have default values if nothing is passed along within the markup.
  - To give a default value, just add the default copy within the <slot> tag.

```
<slot>Here is my slot default</slot>
```

- Named Slots
  - Slots can be given names via the name attribute.
  - Within the markup, give the element a slot attribute with the same name as the slot's name attribute to connect them.

```
<slot name="heading"></slot>
<h1 slot="heading">I will populate the heading slot</h1>
```

SLOTS

- Best Practices
  - Slots are good tools for content or placeholder values that don’t need to be within the overall data of the Vue instance or the individual component.
  - Slots are flexible enough to accommodate multiple tags and can be placed within specific areas via the name/slot attributes.
  - For simpler variable content outside of the site data, slots will be helpful tools.

QUESTIONS

html3@uw.edu

As always feel free to contact the instructional team if you have any questions. They do have a full-time jobs, so they might not get back to you immediately.

If you do not hear back from them in 48 hours, please try again.