Web and Database Computing

adelaide.edu.au

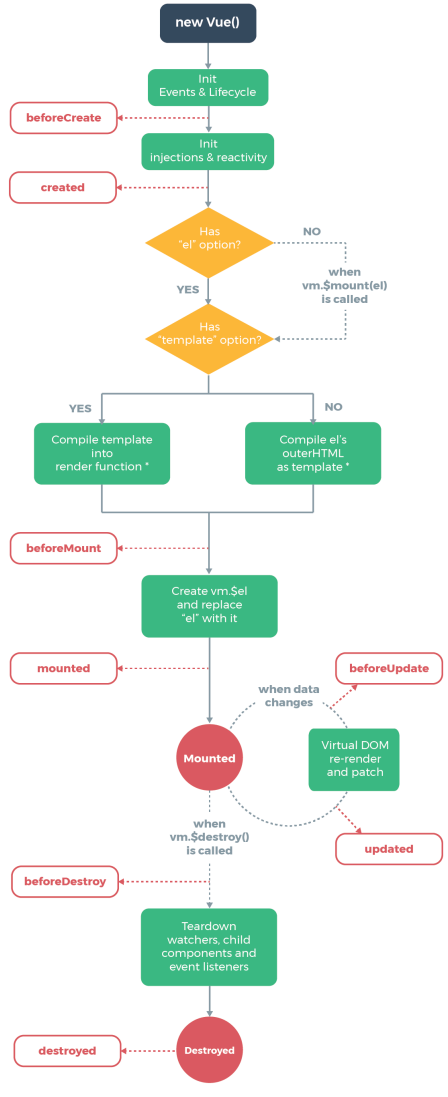
Client-side Frameworks and APIs: Introduction to Vue.js

Client-side frameworks ...

- Provide a method of connecting elements on our page with data in our JavaScript.
  - Data is stored in a data model.
  - The page is defined using a template syntax.
  - Data from the data model is rendered into the page's HTML at runtime.
  - Changes to the data model trigger 'reactive' changes in the HTML.

Getting started with Vue.js

You can follow along in the lecture slides,
but also following the guide at https://vuejs.org/v2/guide/



* template compilation is performed ahead-of-time if using
a build step, e.g. single-file components

How does Vue work?

- Uses **regular HTML**, with a combination of specially named attributes, inline placeholders, and custom components.
- Uses a JavasScript library to connect these attributes and placeholders to:
  - A reactive data model
  - Event listeners
  - Additional context specific functions.
- Provides optimisations to ensure everything runs smoothly

Adding Vue to our website

Add a script tag in our document head:

Store the .js file on our site:

```
<script src="/javascripts/vue.js"></script>
```

Or use a CDN (recommended for performance)

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

Basic Templating

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

https://jsfiddle.net/ian_knight_uofa/3289b16w/3/

Basic Templating; What's happening?

- Use placeholders in our HTML
  - Represented using 'moustaches'

```
{{ placeholder }}
```

- Create a Vue instance
  - Use CSS selector to choose the element to apply the Vue instance to.

```
new Vue({ el: "selector", ... });
```

  - Name the data that will replace the placeholders

```
data: { placeholder:"value" }
```

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

Basic Templating; What's happening?

- Modify the data properties of the Vue instance to automatically update the page.

```
var appdiv = new Vue( ... );
appdiv.text = 'Hi';
```

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

Why Bother?

This all may seem unnecessarily complex, so why bother?

- We're just getting started
- Consider the following example:

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

https://jsfiddle.net/ian_knight_uofa/3289b16w/3/

Templating with objects

- Data properties can be objects as well:

```
<p id="para">{{ obj1.prop1 }} and {{ obj1.prop2 }}</p>
```

```
new Vue({ el: "#para",
          data:  {
                obj1 : { prop1: 'value1',
                         prop2: 2 }
          }
        });
```

Attributes and Style

Vue HTML CSS Result Edit in JSFiddle

```
var vcolour = new Vue({
  el: "#app",
  data: {
    text: "red"
  }
});
```

https://jsfiddle.net/ian_knight_uofa/5v1yw3Lr/3/

Attributes and Style; What's happening?

- Moustache notation only works for text.

```
<h1 id="{{ doesnt_work }}">{{ works_fine }}</h1>
```

- For attributes, replace the desired attribute with `v-bind:attribute_name`.

```
<h1 v-bind:id="now_it_works">{{ works_fine }}</h1>
```

Vue HTML CSS Result Edit in JSFiddle

```
var colour = new Vue({
  el: "#app",
  data: {
    text: "red"
  }
});
```

Attributes and Style; What's happening?

- Classes and styles are special.
- We can use a JavaScript object to specify multiple classes

```
<h1 id="classexample" v-bind:class="{ 'bold_headings': bold_class }">Text</h1>
```

```
new Vue({  el: "#classexample",  data: { bold_class: true }});
```

  - Where the class `bold_headings` will be included if data property `bold_class` is true.
- We can do the same with Styles:

```
<h1 id="styleexample" v-bind:style="{ 'font-family': font }">Some text</h1>
```

```
new Vue({  el: "#styleexample",  data: { font: 'sans-serif' }});
```

  - Where the style `font-family` will be given the value of the data property `font`.

Vue HTML CSS Result Edit in JSFiddle

```
var vcolour = new Vue({
  el: "#app",
  data: {
    text: "red"
  }
});
```

Dynamic Data

We can manipulate the same data to present in different ways:

Vue HTML Result Edit in JSFiddle

```
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      return this.message.split('').reverse().join('')
    }
  }
});
```

https://jsfiddle.net/ian_knight_uofa/wvszc3pt/4/

Dynamic Data; What's happening?

- The return value of a computed function can be used in place of a regular data property.
- A computed function that references a data property of the Vue instance will be run any time that data property is changed.

```
computed: {
    reversedMessage: function () {
        console.log('boop'+this.d2);
        return this.message.split('').reverse().join('');
    }
}
```

- Modifying the data properties of the Vue automatically runs the function, updating the computed properties.

```
{{ reversedMessage }}
```

Vue HTML Result Edit in JSFiddle

```
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      return this.message.split('').reverse().
    }
  }
});
```

Summary

- Vue.js Uses **regular HTML**, with a combination of specially named attributes, inline placeholders, and custom components to make building webpages easier.
- Uses a reactive data model to update the page automatically when the data changes.
- Inline placeholders use moustache notation `{{ variable }}`.
- Attributes can be connected to Vue using the `v-bind:` prefix.
- Computed functions allow for dynamically calculated data.

THE UNIVERSITY
of ADELAIDE