Vue.js

The Progressiv JavaScript Framework

What is Vue.js?

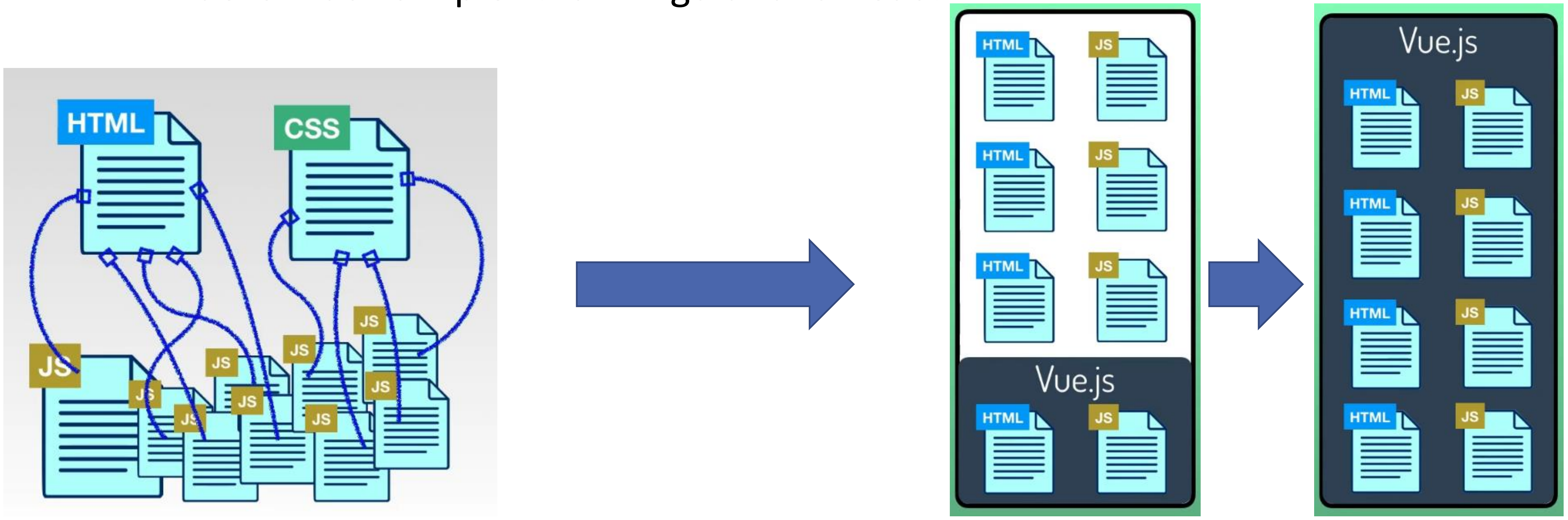
- Vue is a JavaScript framework **for building user interfaces**
- The core library is focused on the view layer only
  - And Vue is easy to pick up and integrate with other libraries or existing projects
- But Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries
- Vue was created and open sourced by Evan You, after working for Google using AngularJS in a number of projects

what if I could just extract the part that I really liked about Angular and build something really lightweight without all the extra concepts involved?

Why use Vue?

- It typically takes developers less than a day reading the guide to learn enough to build non-trivial applications with Vue

- Vue is much simpler than Angular and React

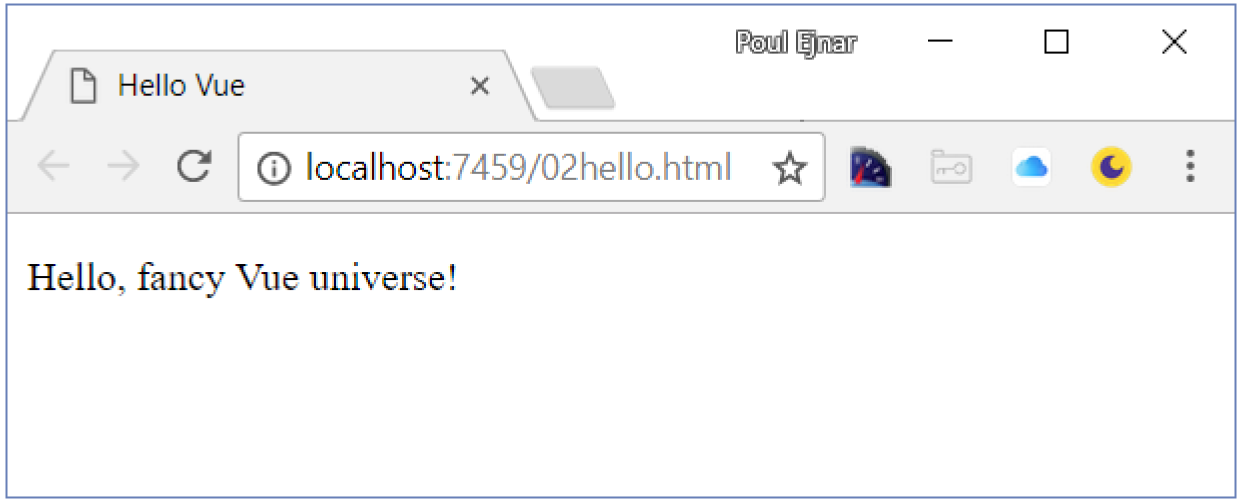


Vue's Hello World

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Hello Vue</title>
    <script src="https://unpkg.com/vue"></script>
</head>
<body>
    <div id="app">
        <p>Hello, {{ greetee }}!</p>
    </div>
    <script>
        new Vue({
            el: '#app',
            data: {
                greetee: 'fancy Vue universe'
            }
        });
    </script>
</body>
</html>
```

Vue library

Data binding



Hello, fancy Vue universe!

Vue Basics

Templates

- **Vue uses an HTML-based template syntax** that allows you to **declaratively bind** the rendered DOM to the underlying Vue instance's data
- All Vue templates are valid HTML that can be parsed by spec-compliant browsers and HTML parsers

```
<div id="app">
    <p>Hello, {{ greeting }}!</p>
</div>
```

Vue instance’s data

- Vue uses an HTML-based template syntax that allows you to declaratively bind the rendered DOM to **the underlying Vue instance’s data**

Vue instance

```
<script>
    var vm = new Vue({
        el: '#app',
        data: {
            greeting: 'fancy Vue universe'
        }
    });
</script>
```

- We often use the variable vm (short for ViewModel) to refer to our Vue instance
  - Vue’s design was partly inspired by the MVVM pattern

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Instance options

- When you create a Vue instance, you pass in an **options object**
- When a Vue instance is created, it adds all the properties found in its data object to Vue's reactivity system

```
// Our data object
var data = { a: 1 }

// The object is added to a Vue instance
var vm = new Vue({
    data: data
})
```

- Properties in data are only reactive if they existed when the instance was created

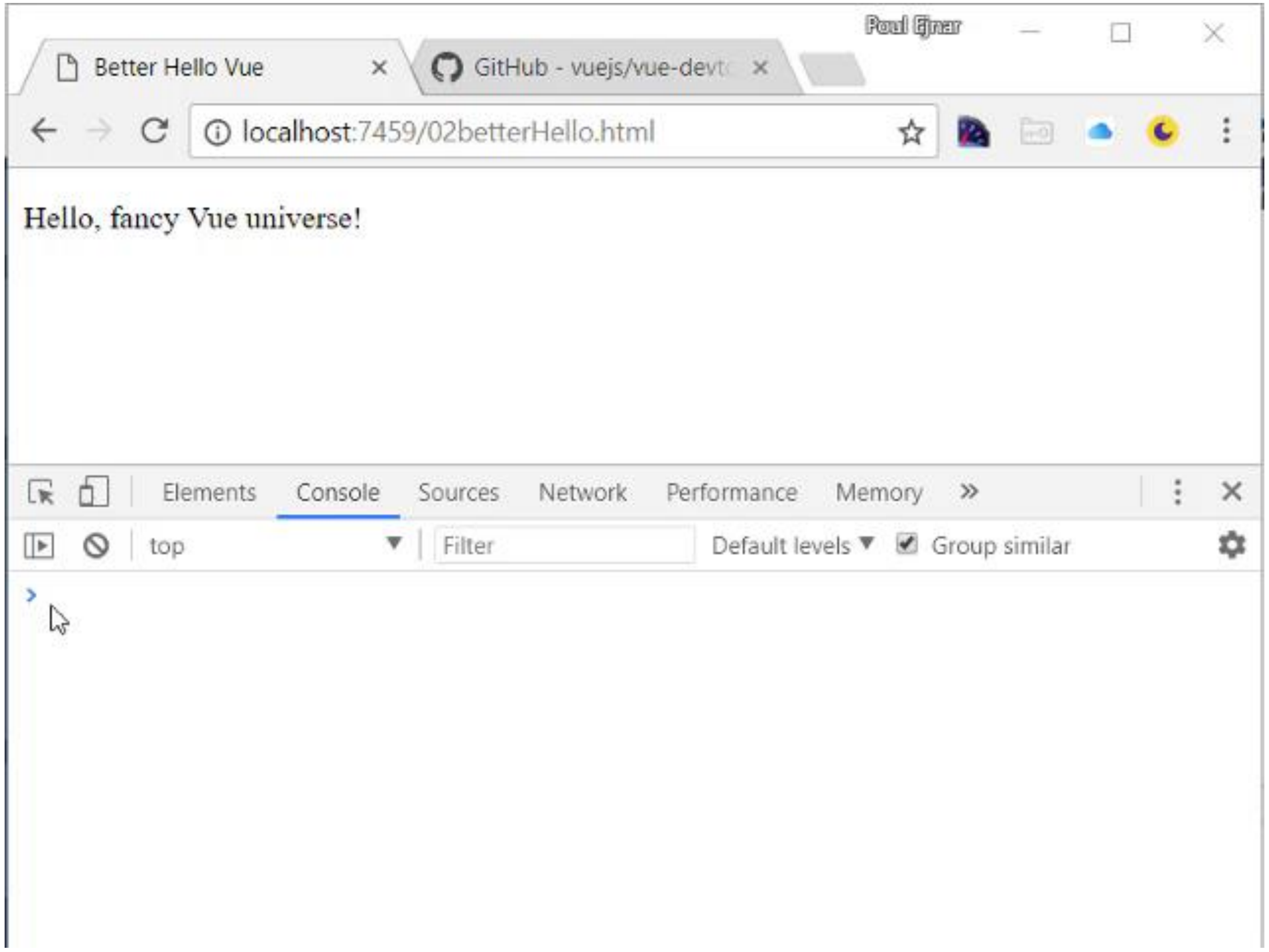
Instance methods

- Storing a function as a property of the methods object makes it available in your templates:

```
new Vue({
    el: '#app',
    data: {
        status: 2
    },
    methods: {
        statusFromId(id) {
            const status = ({
                0: 'Asleep',
                1: 'Eating',
                2: 'Learning Vue'
            })[id];
            return status || 'Unknown status: ' + id;
        }
    }
});
```

Reactive

- One of Vue's most valued features is the reactivity system
- Models are just plain JavaScript objects
- **The view updates when you modify data in the model**
- Vue provides optimized re-rendering out of the box without you having to do anything



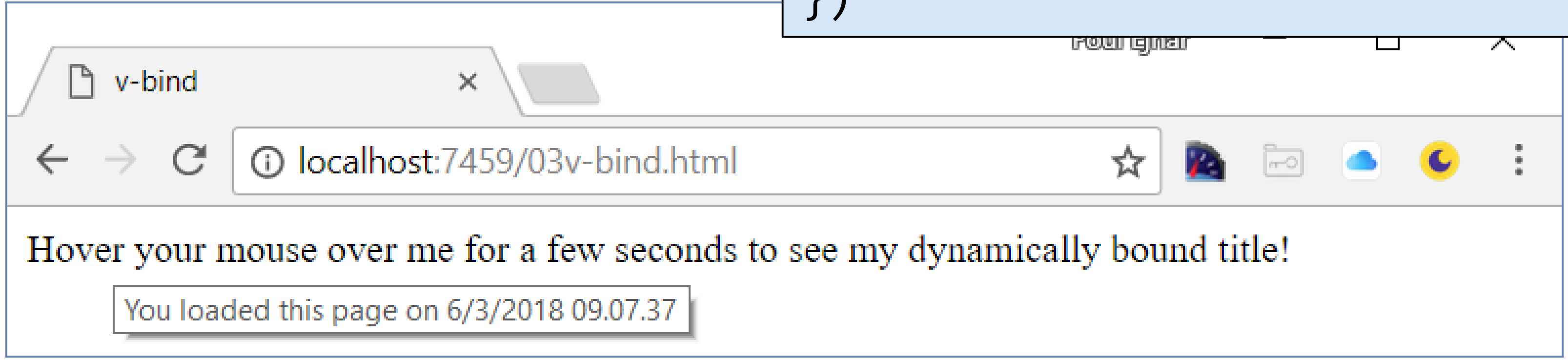
Hello, fancy Vue universe!

Directives

- Directives are html attributes that are prefixed with v- to indicate that they are special attributes provided by Vue
- Directives apply special reactive behaviour to the rendered DOM
- **Use v-bind to bind a html attribute to a property in the model**

```html
<div id="app-3">
  <span v-bind:title="message">
    Hover your mouse over me for a few seconds
    to see my dynamically bound title!
  </span>
</div>
```

```javascript
var app3 = new Vue({
    el: '#app-3',
    data: {
        message: 'You loaded this page on ' +
        new Date().toLocaleString()
    }
})
```



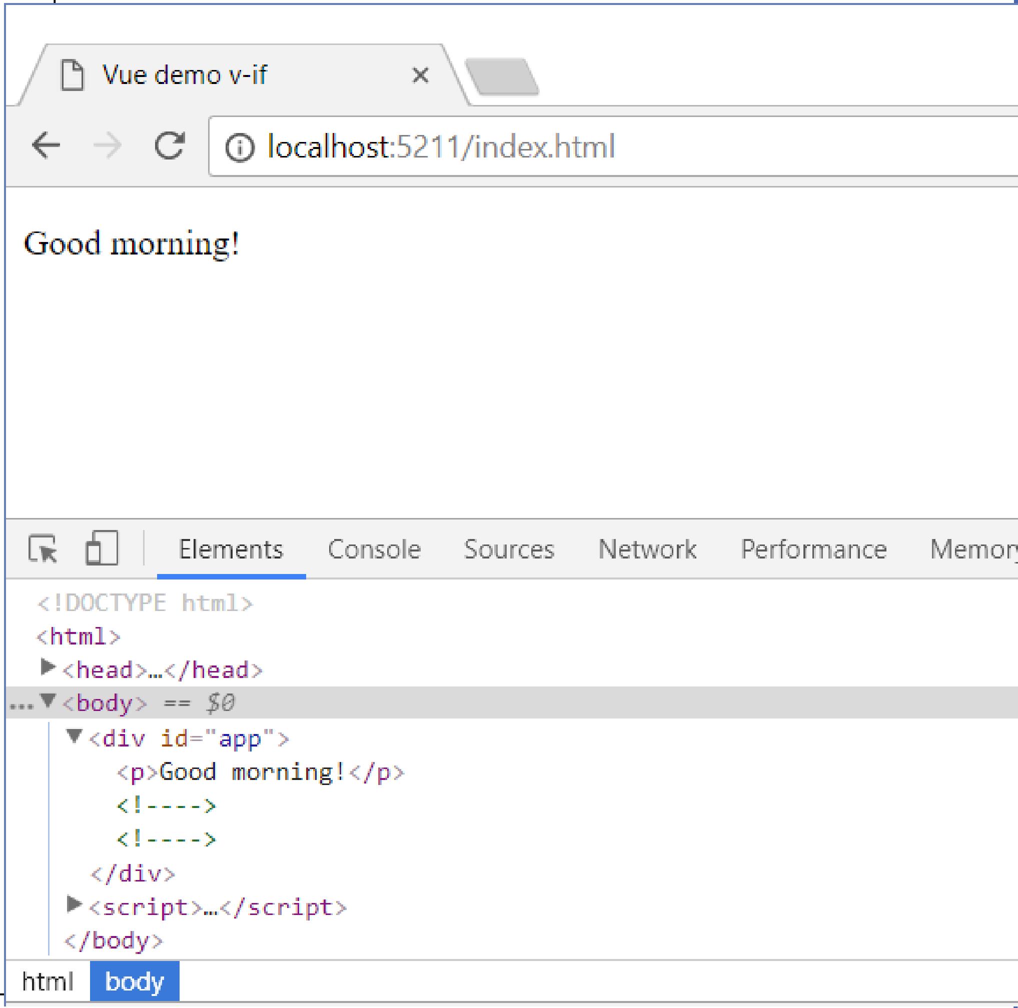
Hover your mouse over me for a few seconds to see my dynamically bound title!

You loaded this page on 6/3/2018 09.07.37

Notice: `v-bind` is optional `:title` does the same

v-if

```
<body>
    <div id="app">
        <p v-if="isMorning">Good morning!</p>
        <p v-if="isAfternoon">Good afternoon!</p>
        <p v-if="isEvening">Good evening!</p>
    </div>
    <script>
        var hours = new Date().getHours();
        new Vue({
            el: '#app',
            data: {
                isMorning: hours < 12,
                isAfternoon: hours >= 12 && hours < 18,
                isEvening: hours >= 18
            }
        });
    </script>
</body>
```



Good morning!

```
<!DOCTYPE html>
<html>
▶<head>…</head>
…▼<body> == $0
  ▼<div id="app">
      <p>Good morning!</p>
      <!---->
      <!---->
    </div>
  ▶<script>…</script>
  </body>
```

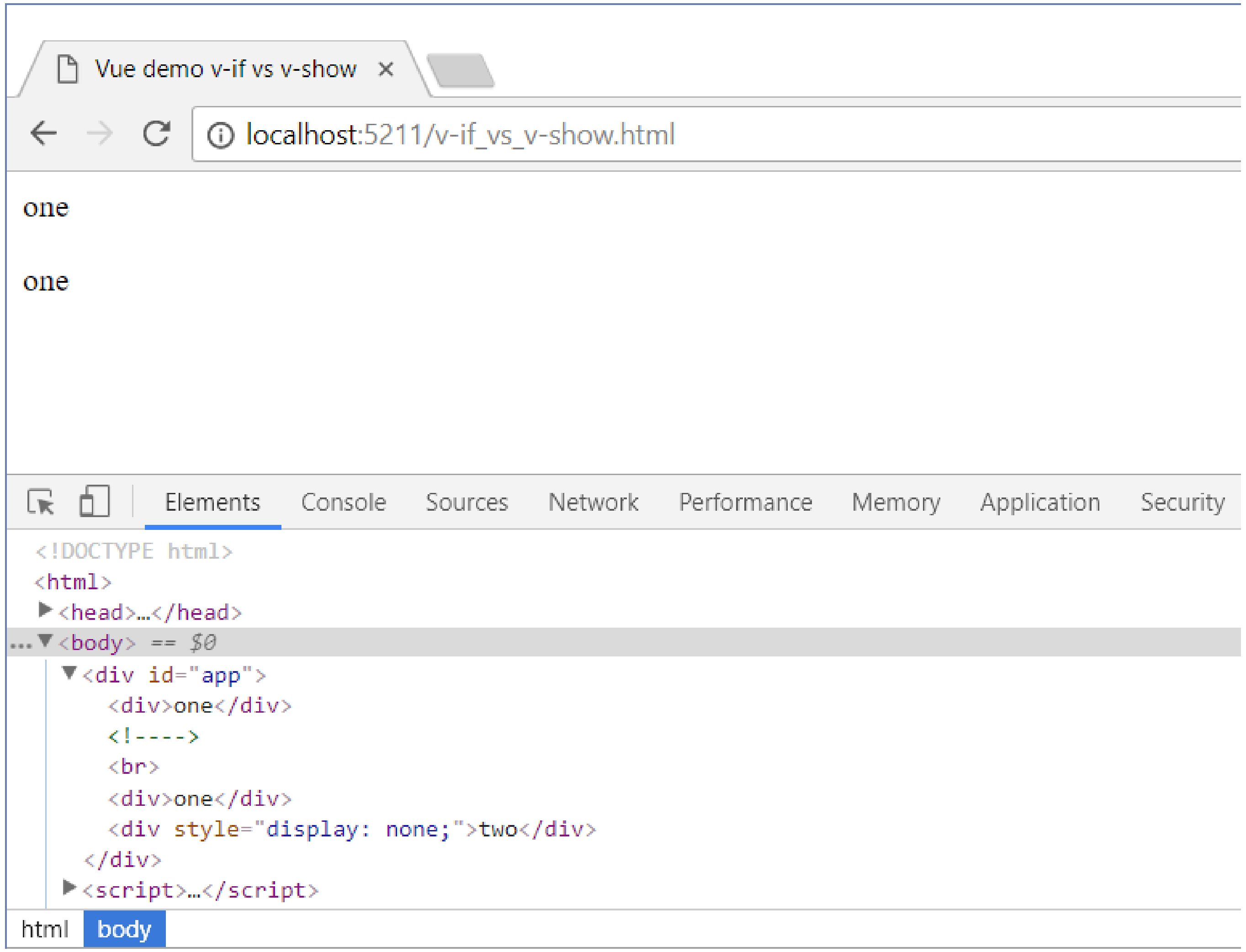
v-if variations

```
<div id="app">
        <p v-if="hours < 12">Good morning!</p>
        <p v-if="hours >= 12 && hours < 18">Good afternoon!</p>
        <p v-if="hours >= 18">Good evening!</p>
    </div>
    <script>
        new Vue({
            el: '#app',
            data: {
                hours: new Date().getHours()
            }
        });
    </script>
```

```
<div v-if="state === 'loading'">Loading…</div>
<div v-else-if="state === 'error'">An error occurred</div>
<div v-else>…our content!</div>
```

v-if vs. v-show

```
<div id="app">
        <div v-if="true">one</div>
        <div v-if="false">two</div>
        <br />
        <div v-show="true">one</div>
        <div v-show="false">two</div>
    </div>
    <script>
        new Vue({
            el: '#app',
            data: {
            }
        });
    </script>
```



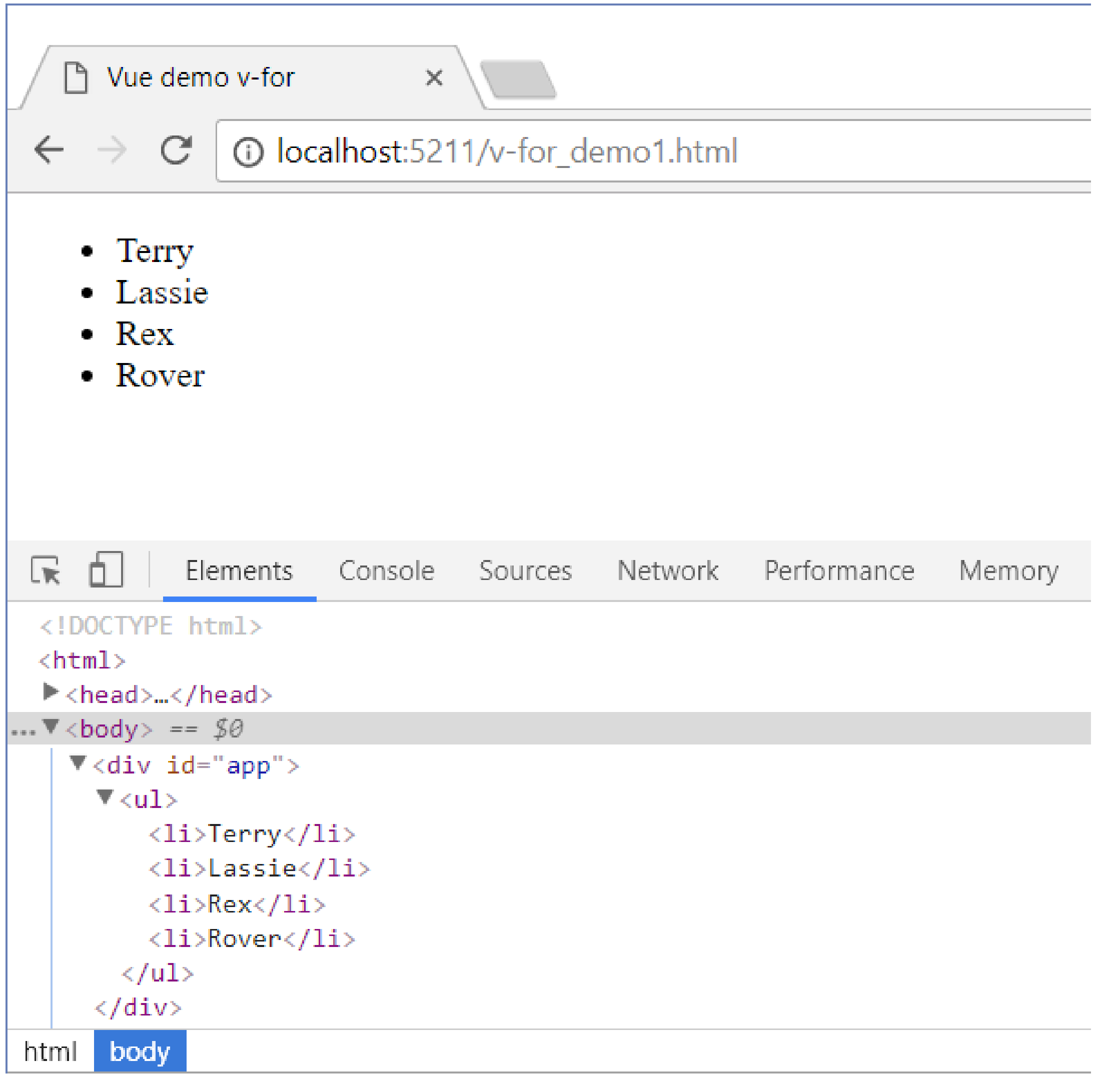
```
one

one
```

```
<!DOCTYPE html>
<html>
▶<head>…</head>
…▼<body> == $0
  ▼<div id="app">
     <div>one</div>
     <!---->
     <br>
     <div>one</div>
     <div style="display: none;">two</div>
   </div>
  ▶<script>…</script>
```

v-for

```
<div id="app">
  <ul>
    <li v-for="dog in dogs">{{ dog }}</li>
  </ul>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      dogs: ['Terry', 'Lassie', 'Rex', 'Rover']
    }
  });
</script>
```



- Terry
- Lassie
- Rex
- Rover

```
<!DOCTYPE html>
<html>
▶<head>…</head>
…▼<body> == $0
  ▼<div id="app">
    ▼<ul>
       <li>Terry</li>
       <li>Lassie</li>
       <li>Rex</li>
       <li>Rover</li>
     </ul>
   </div>
```

v-for variations

```
<div id="app">
       <ul>
           <li v-for="dog in dogs">{{ dog.name }}</li>
       </ul>
   </div>
   <script>
       new Vue({
           el: '#app',
           data: {
               dogs: [
                   {
                       name: 'Terry',
                       age: 3,
                   },
                   {
                       name: 'Lassie',
                       age: 11,
                   },
```

You can also use **of** as the delimiter instead of in, so that it is closer to JavaScript's syntax for iterators:

```
v-for="dog of dogs"
```

v-for also supports an optional second argument for the index of the current item:

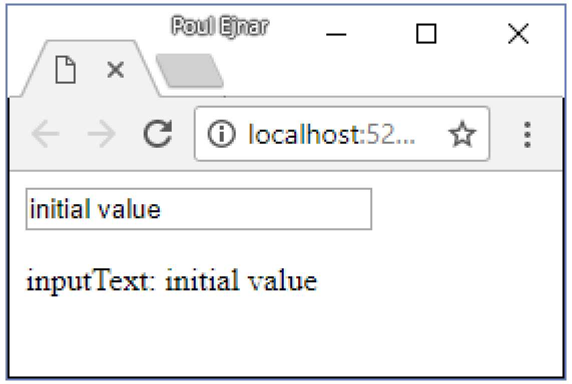
```
<li v-for="(dog, index) in dogs">
{{ index }} - {{ dog.name }}</li>
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Two-Way Data Binding

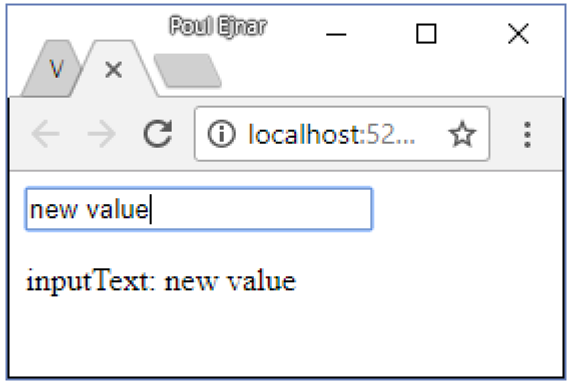
Use `v-model`

```
<div id="app">
  <input type="text" v-model="inputText">
  <p>inputText: {{ inputText }}</p>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      inputText: 'initial value'
    }
  });
</script>
```



initial value

inputText: initial value



new value

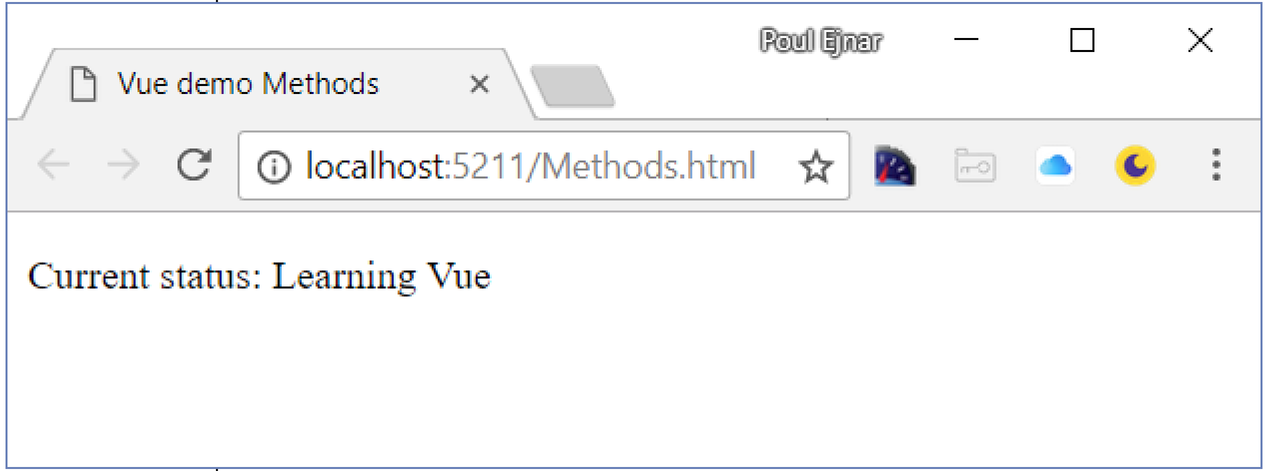
inputText: new value

AARHUS UNIVERSITY SCHOOL OF ENGINEERING

Methods

- Storing a function as a property of the methods object makes it available in your templates

```
<div id="app">
    <p>Current status: {{ statusFromId(status) }}</p>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            status: 2
        },
        methods: {
            statusFromId(id) {
                const status = ({
                    0: 'Asleep',
                    1: 'Eating',
                    2: 'Learning Vue'
                })[id];
                return status || 'Unknown status: ' + id;
            }
        }
    });
</script>
```

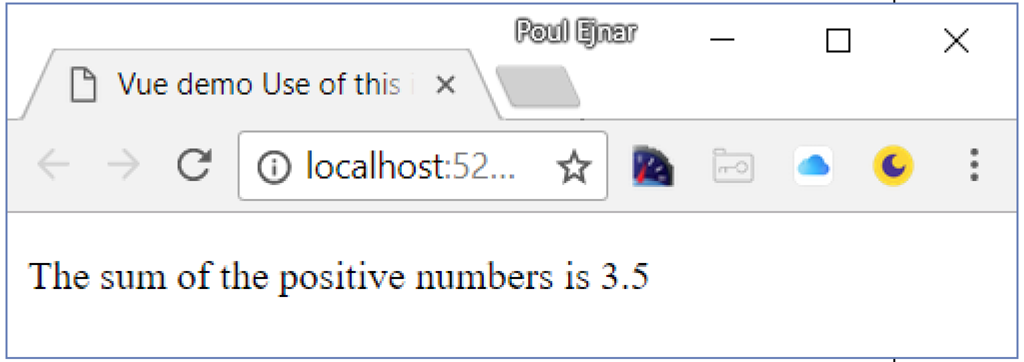


Current status: Learning Vue

Use of this in methods

- In a method, **this** refers to the component that the method is attached to

```
<div id="app">
    <p>The sum of the positive numbers is {{ getPositiveNumbersSum() }}</p>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            numbers: [-5, 0, 2, -1, 1, 0.5]
        },
        methods: {
            getPositiveNumbers() {
                return this.numbers.filter((number) => number >= 0);
            },
            getPositiveNumbersSum() {
                return this.getPositiveNumbers().reduce((sum, val) => sum + val);
            }
        }
    });
</script>
```

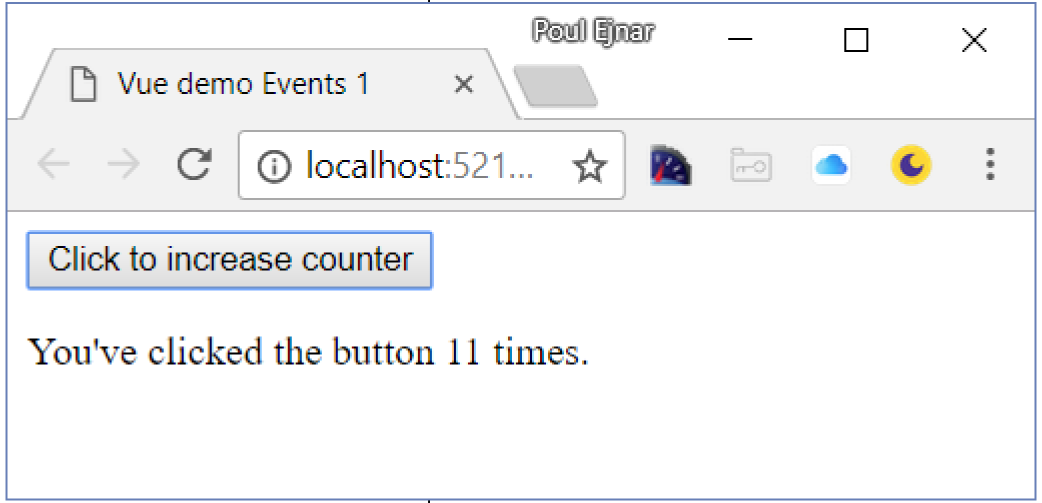


The sum of the positive numbers is 3.5

Events

- Use the v-on directive to bind an event to an event handler

```
<div id="app">
  <button v-on:click="increase">Click to increase counter</button>
  <p>You've clicked the button {{ counter }} times.</p>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      counter: 0
    },
    methods: {
      increase(e) {
        this.counter++;
      }
    }
 });
</script>
```



You've clicked the button 11 times.

Use @ as a v-on shortcut :

```
<button @click="increase">Click to increase counter</button>
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Event Modifiers

- To prevent the default action of the event from happening
```
<a @click.prevent="handleClick">Demo</a>
```
- To stop the event from propagating
```
<button @click.stop="handleClick">Demo</button>
```
- To have the event listener be triggered only the first time the event is fired
```
<button @click.once="handleFirstClick">Demo</button>
```
- To use capture mode, meaning that the event will be triggered on this element before it is dispatched on the elements below it in the tree
```
<div @click.capture="handleCapturedClick">...</div>
```
- To only trigger the handler when the event was triggered on the element itself
```
<div @click.self="handleSelfClick">...</div>
```
- You can chain modifiers together
```
<div @click.stop.capture.once>...</div>
```

key modifiers

- These are used on keyboard events so that you can fire the event only when a certain key is pressed

```
<div id="app">
  <form @keyup.esc="handleKeyup">
     <label>Dette er en form</label>
     <input type="text" v-model="message" />
  </form>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      message: "Demo",
    },
    methods: {
      handleKeyup(e) {
        alert("You pressed esc!");
      }
    }
  });
</script>
```

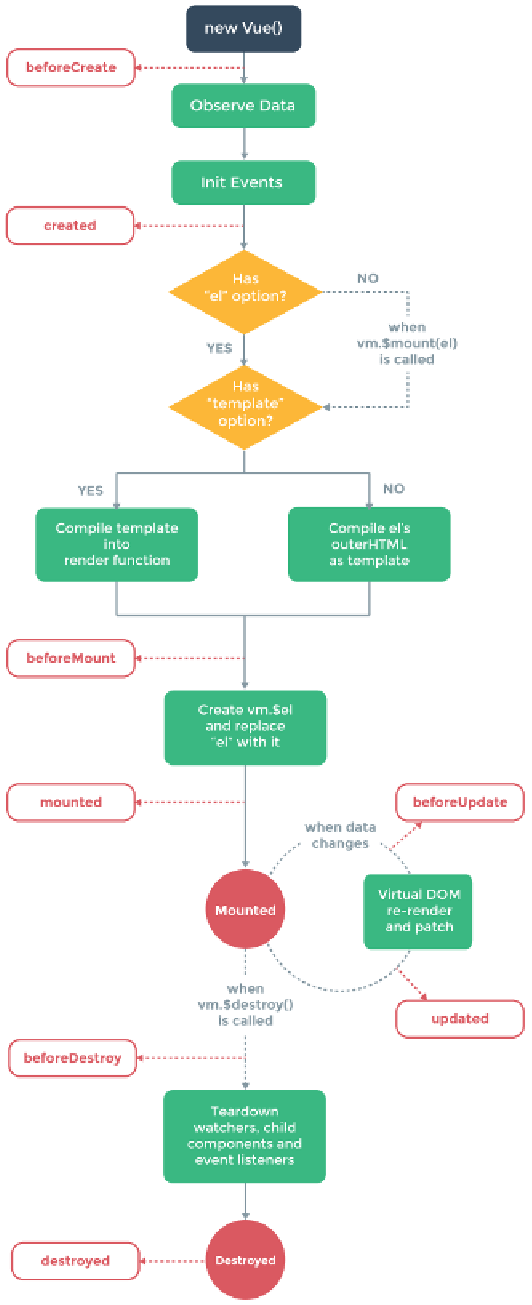
```
<div id="app">
  <form @keyup="handleKeyup">
    <label>Dette er en form</label>
    <input type="text" v-model="message" />
  </form>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      message: "Demo",
    },
    methods: {
      handleKeyup(e) {
        if (e.keyCode === 27) {
          alert("You pressed esc!");
        }
      }
    }
  });
</script>
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Life-Cycle Hooks

- Vue has eight life-cycle hooks (events):
  - beforeCreate
    - is fired before the instance is initialized
  - Created
    - is fired after reactivity is initiated
  - beforeMount
    - is fired when the DOM element is ready to be created
  - Mounted
    - is fired after the DOM element is created (but it isn’t guaranteed to be inserted into the DOM yet - use nextTick for that)
  - beforeUpdate
    - is fired when there are changes to be made to the DOM output
  - updated
    - is fired after changes have been written to the DOM
  - beforeDestroy
    - is fired when the component is about to be destroyed and removed from the DOM.
  - destroyed
    - is fired after the component has been destroyed

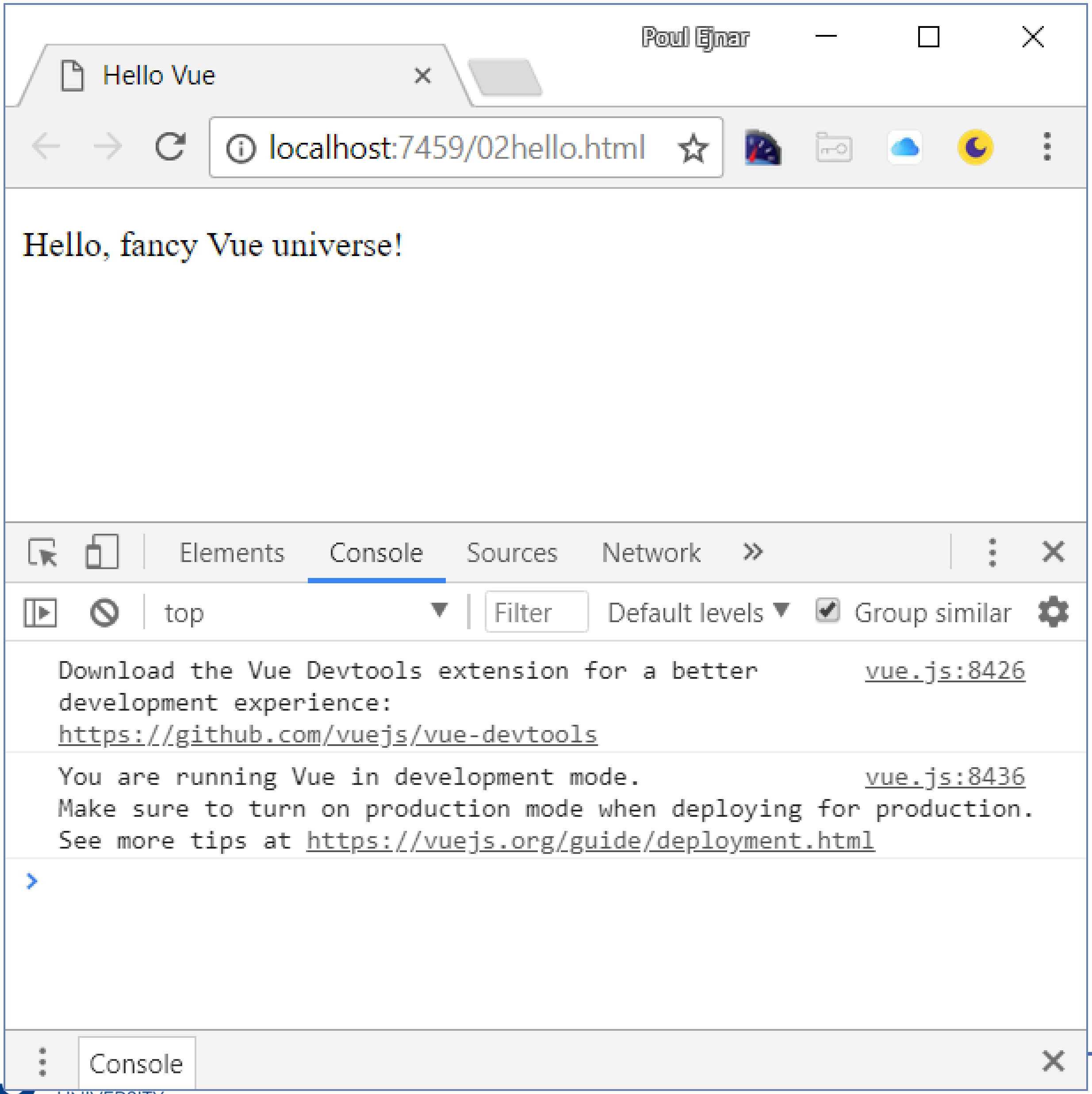
Vue 2


new Vue()
beforeCreate
Observe Data
Init Events
created
Has "el" option?
NO
when vm.$mount(el) is called
YES
Has "template" option?
YES
NO
Compile template into render function
Compile el's outerHTML as template
beforeMount
Create vm.$el and replace "el" with it
mounted
beforeUpdate
when data changes
Mounted
Virtual DOM re-render and patch
when vm.$destroy() is called
updated
beforeDestroy
Teardown watchers, child components and event listeners
destroyed
Destroyed

Life-Cycle Hooks - mounted

```
<div id="app">
   <p>Hello world</p>
</div>
<script>
    new Vue({
        el: '#app',
        mounted() {
            // Element might not have been added to the DOM yet
            this.$nextTick(() => {
                // Element has definitely been added to the DOM now
                alert("element has been mounted :-)");
            });
        }
    });
</script>
```

Vue Devtools



Hello, fancy Vue universe!

```
Download the Vue Devtools extension for a better          vue.js:8426
development experience:
https://github.com/vuejs/vue-devtools

You are running Vue in development mode.                  vue.js:8436
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html
```

UNIVERSITY
SCHOOL OF ENGINEERING

References & Links

- Getting Started – The official site:
  https://vuejs.org/v2/guide/
- Replacing jQuery With Vue.js by Sarah Drasner
  https://www.smashingmagazine.com/2018/02/jquery-vue-javascript/
  https://sarahdrasnerdesign.com/

- Cookbook
  https://vuejs.org/v2/cookbook/index.html
- Vue.js Up & Running Ch1-2Excerpt
  http://shop.oreilly.com/product/0636920103455.do
- Bootstrap + Vue
  https://bootstrap-vue.js.org/