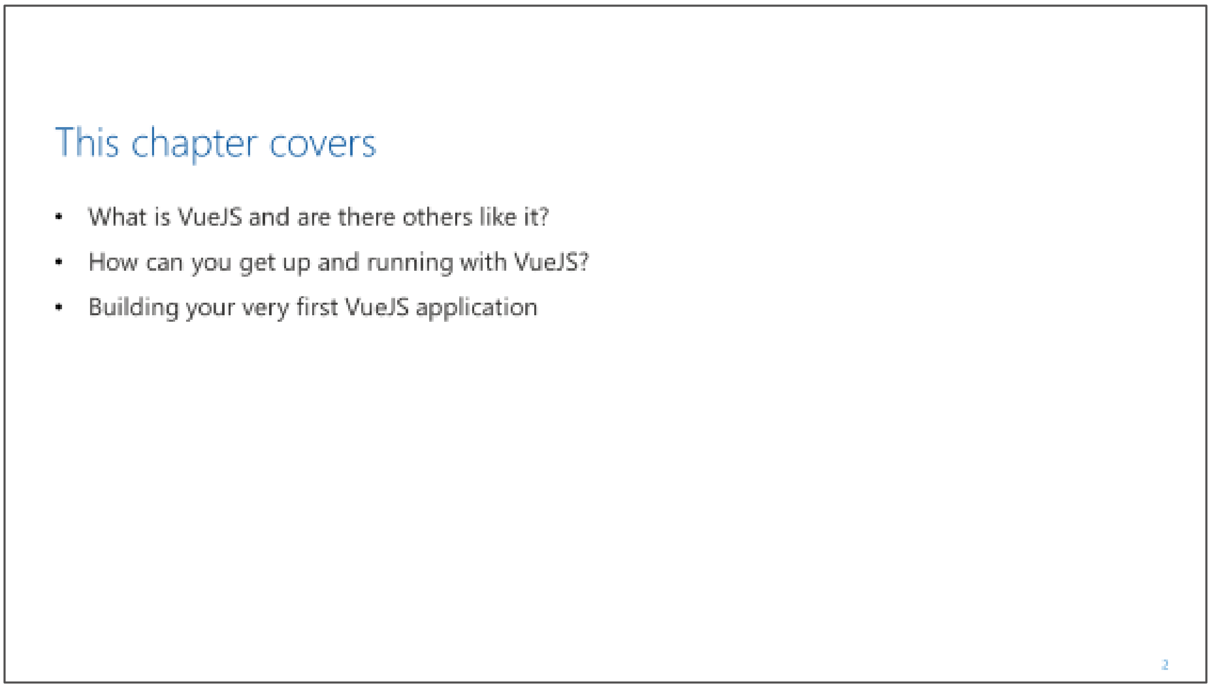
VueJS

PROGRAMMING WITH JAVASCRIPT

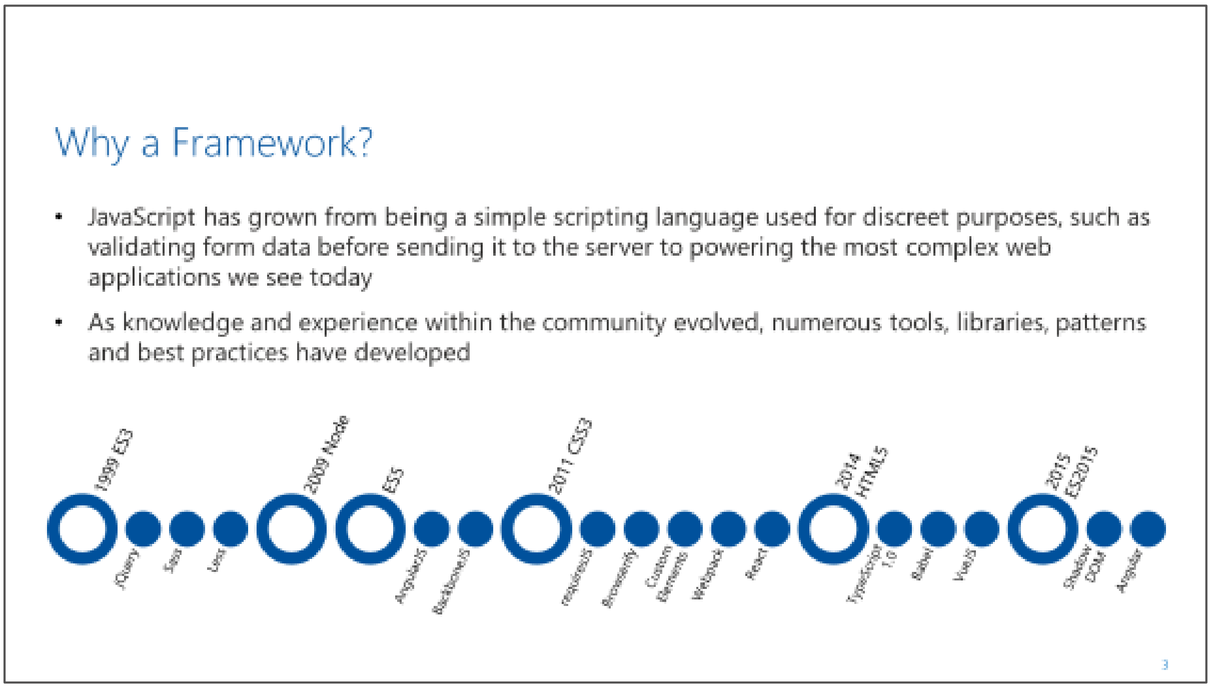
QA



This chapter covers

- What is VueJS and are there others like it?
- How can you get up and running with VueJS?
- Building your very first VueJS application

2



Why a Framework?

- JavaScript has grown from being a simple scripting language used for discreet purposes, such as validating form data before sending it to the server to powering the most complex web applications we see today
- As knowledge and experience within the community evolved, numerous tools, libraries, patterns and best practices have developed

1999 ES3
jQuery
Sass
Less
2009 Node
ES5
AngularJS
BackboneJS
2011 CSS3
requireJS
Browserify
Custom Elements
Webpack
React
2014 HTML5
TypeScript 1.0
Babel
VueJS
2015 ES2015
Shadow DOM
Angular
3

1999 Saw the release of the 3rd edition of the ECMAScript standard and proved to be widely adopted by the major browsers of the day. The 4th edition was not so lucky.

2006 jQuery was released and so began the great romance of the web with jQuery – becoming the ubiquitous tool to use across the web for cross-browser compatibility and writing less code.

That same year Sass saw the light of day and the front-end community started the journey into the world of transpilation.

2009You know what they say, less is more. Certainly seemed to be the case as in 2009 less appeared and the front-end had an alternative for CSS transpilation.

At the end of 2009 ES5 was released and proved to be the unifying JavaScript force for browsers, with them all widely implementing it.

And how dare I leave NodeJS till last? Providing server -side JavaScript and the platform on which so many of the tools we have used and will use are built on.

2010The year of AngularJS and BackboneJS. Looking to bring order to chaos, structure to confusion – both frameworks looked at how we build scalable single page web applications. Design patterns and modules aplenty.

2011The first CSS3 modules started to appear, requireJS and browserify hit the scene and the specification for Cutom Elements started to solidify.

2012Webpack is released – a worthy successor to Gulp and Grunt – it is now used by all 3 big frameworks (Angular, React and Vue) to power their CLIs

2014 HTML5 is a W3C Recommendation and widely adopted in the browsers. Some of course had significant catching up to do.

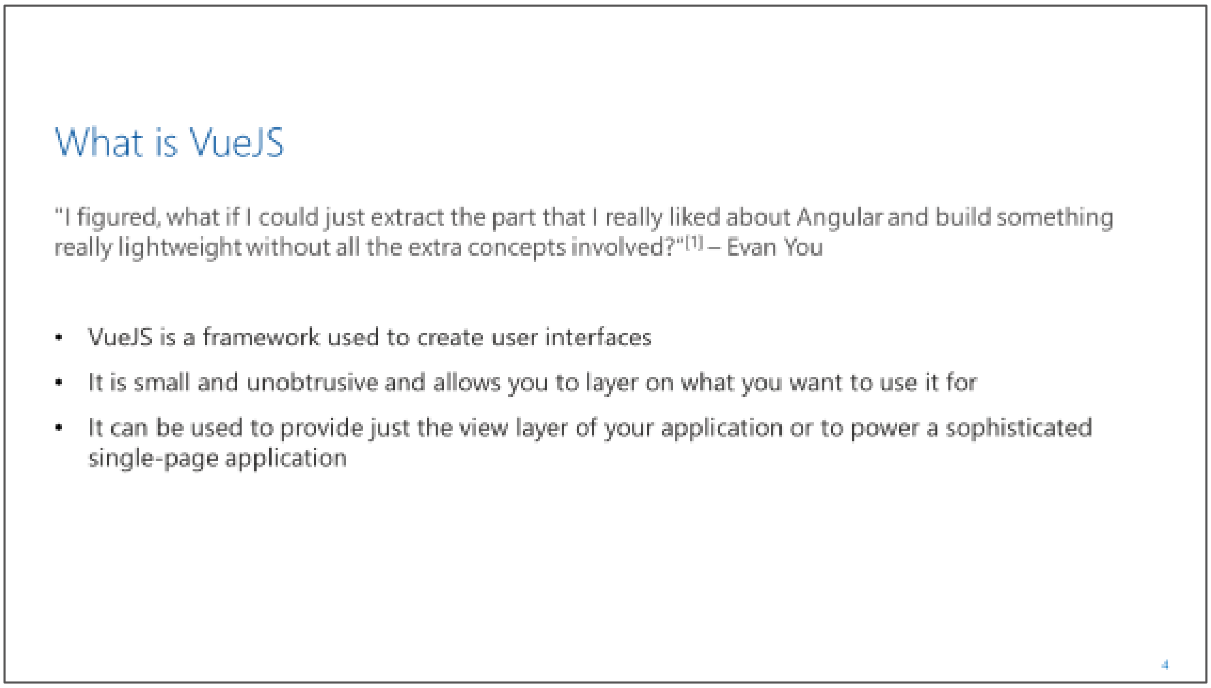
Babel is released and the front-end community gets another dose of transpilation – this time, ES6 to ES5 (hence why Babel was originally called 6to5).

TypeScript (having been made public in 2012) released version 1.0.

VueJS is released by ex-Google employee Evan You.

2015ES2015 is released and the JavaScript community receives Promises, Modules (just the syntax mind!), and a whole host of other features provided by libraries and frameworks until now.

2016Angular is released as a complete re-write of Google's framework. Having worked closely with Microsoft, Angular is primarily used in a TypeScript stack.



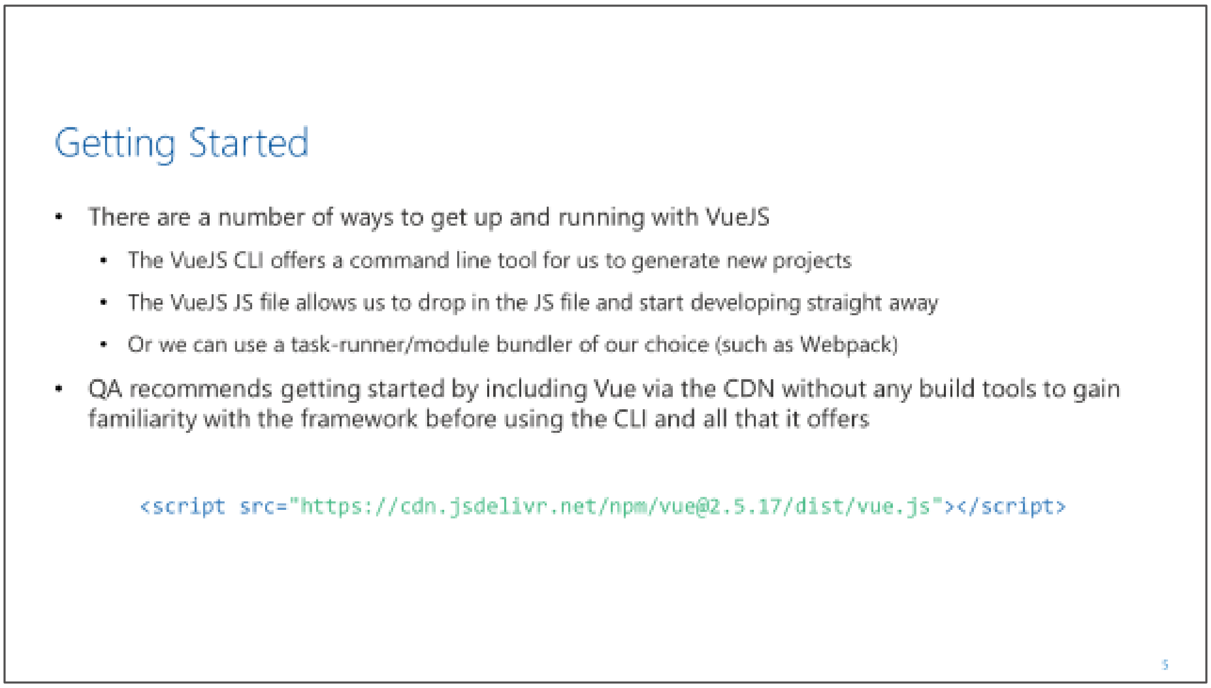
What is VueJS

"I figured, 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?"[1] – Evan You

- VueJS is a framework used to create user interfaces
- It is small and unobtrusive and allows you to layer on what you want to use it for
- It can be used to provide just the view layer of your application or to power a sophisticated single-page application

4

[1] "Between the Wires | Evan You". Between the Wires. 2016-11-03

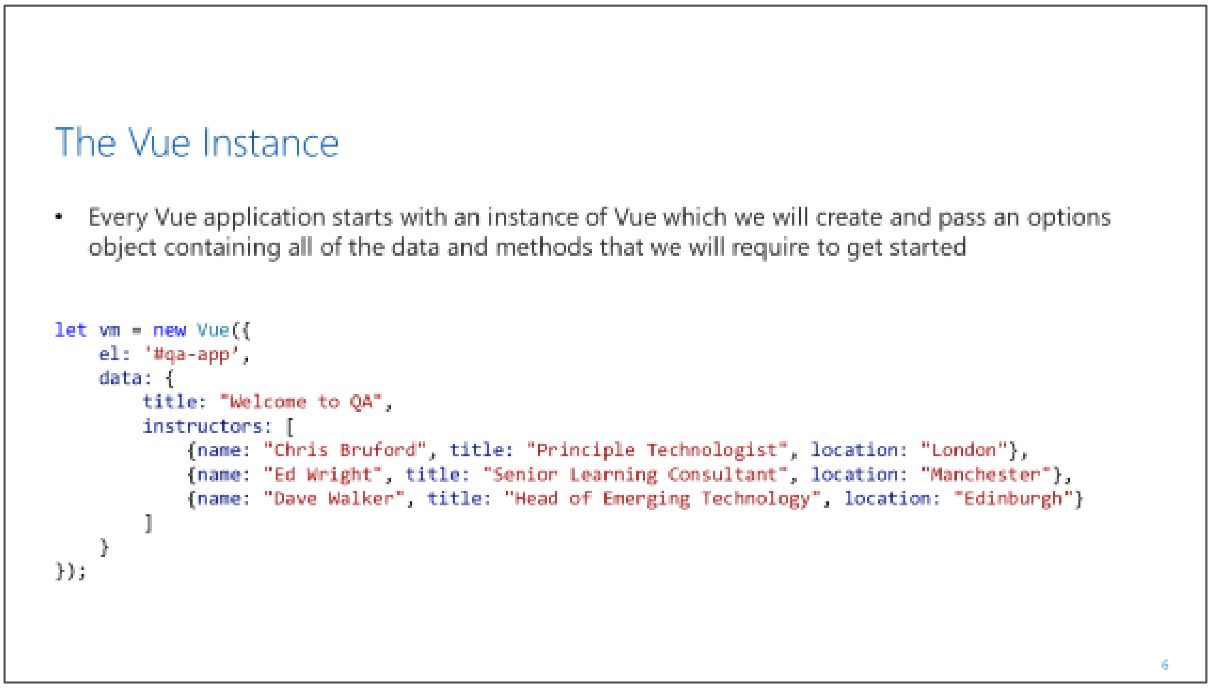


Getting Started

- There are a number of ways to get up and running with VueJS
  - The VueJS CLI offers a command line tool for us to generate new projects
  - The VueJS JS file allows us to drop in the JS file and start developing straight away
  - Or we can use a task-runner/module bundler of our choice (such as Webpack)
- QA recommends getting started by including Vue via the CDN without any build tools to gain familiarity with the framework before using the CLI and all that it offers

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.5.17/dist/vue.js"></script>
```

5

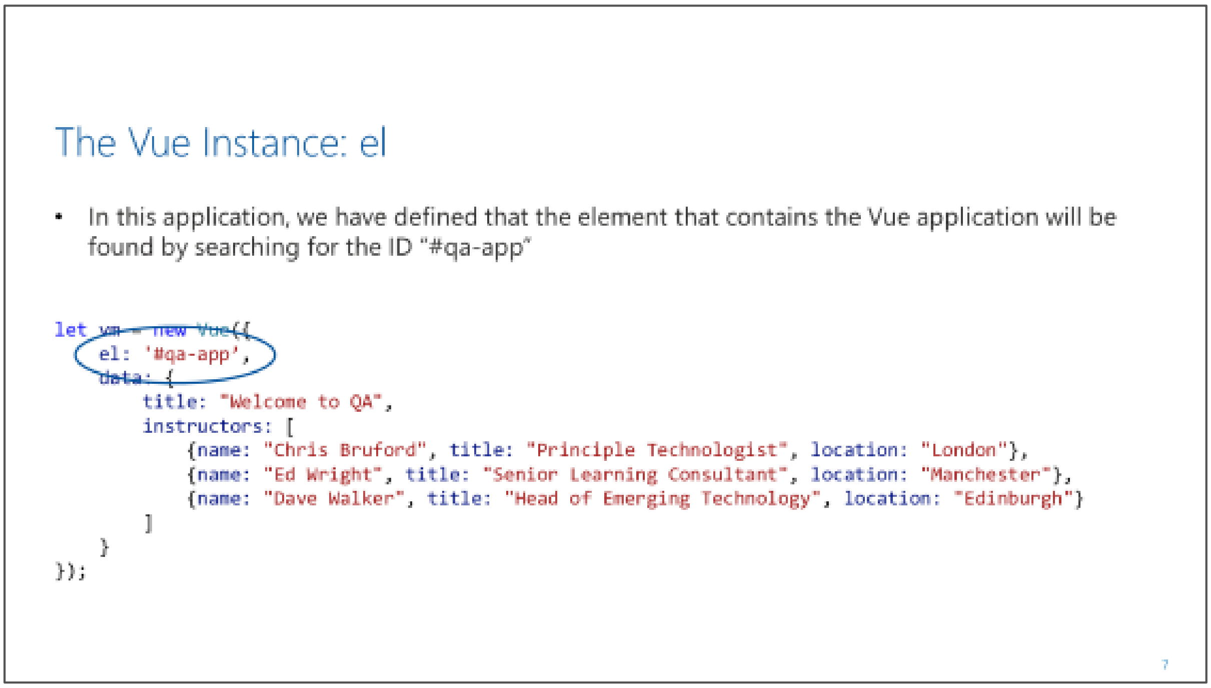


The Vue Instance

- Every Vue application starts with an instance of Vue which we will create and pass an options object containing all of the data and methods that we will require to get started

```
let vm = new Vue({
    el: '#qa-app',
    data: {
        title: "Welcome to QA",
        instructors: [
            {name: "Chris Bruford", title: "Principle Technologist", location: "London"},
            {name: "Ed Wright", title: "Senior Learning Consultant", location: "Manchester"},
            {name: "Dave Walker", title: "Head of Emerging Technology", location: "Edinburgh"}
        ]
    }
});
```

6

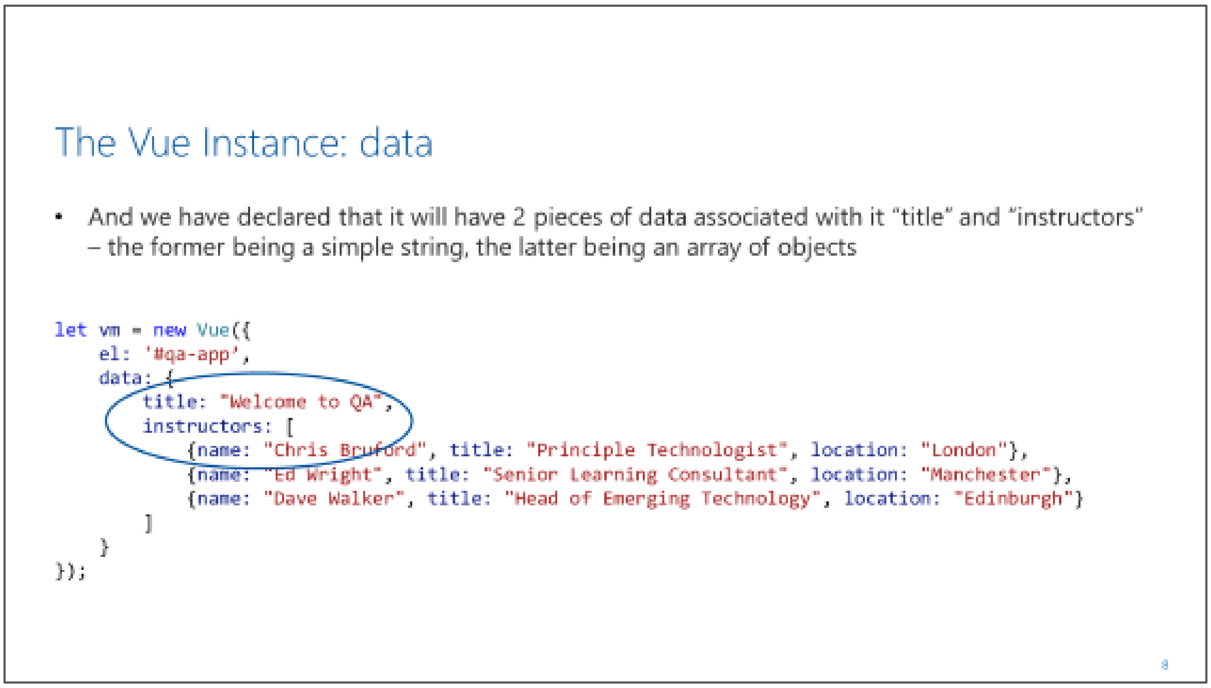


The Vue Instance: el

- In this application, we have defined that the element that contains the Vue application will be found by searching for the ID "#qa-app"

```
let vm = new Vue({
    el: '#qa-app',
    data: {
        title: "Welcome to QA",
        instructors: [
            {name: "Chris Bruford", title: "Principle Technologist", location: "London"},
            {name: "Ed Wright", title: "Senior Learning Consultant", location: "Manchester"},
            {name: "Dave Walker", title: "Head of Emerging Technology", location: "Edinburgh"}
        ]
    }
});
```

7

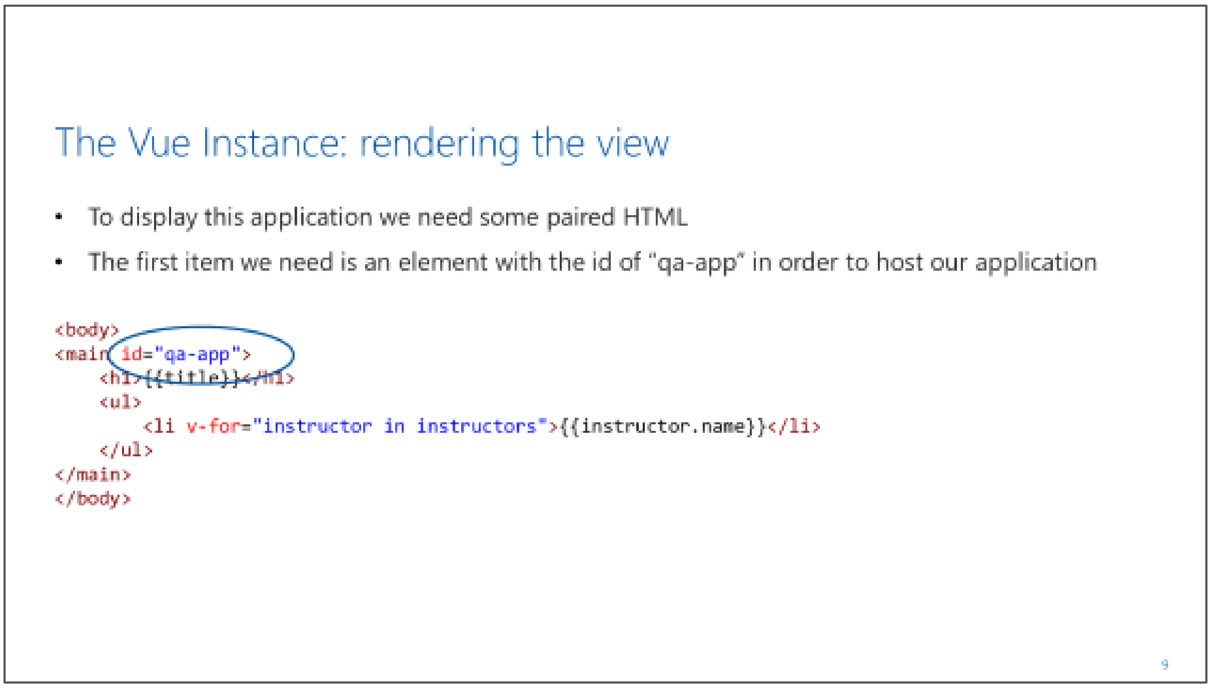


The Vue Instance: data

- And we have declared that it will have 2 pieces of data associated with it “title” and “instructors” – the former being a simple string, the latter being an array of objects

```
let vm = new Vue({
    el: '#qa-app',
    data: {
        title: "Welcome to QA",
        instructors: [
            {name: "Chris Bruford", title: "Principle Technologist", location: "London"},
            {name: "Ed Wright", title: "Senior Learning Consultant", location: "Manchester"},
            {name: "Dave Walker", title: "Head of Emerging Technology", location: "Edinburgh"}
        ]
    }
});
```

8

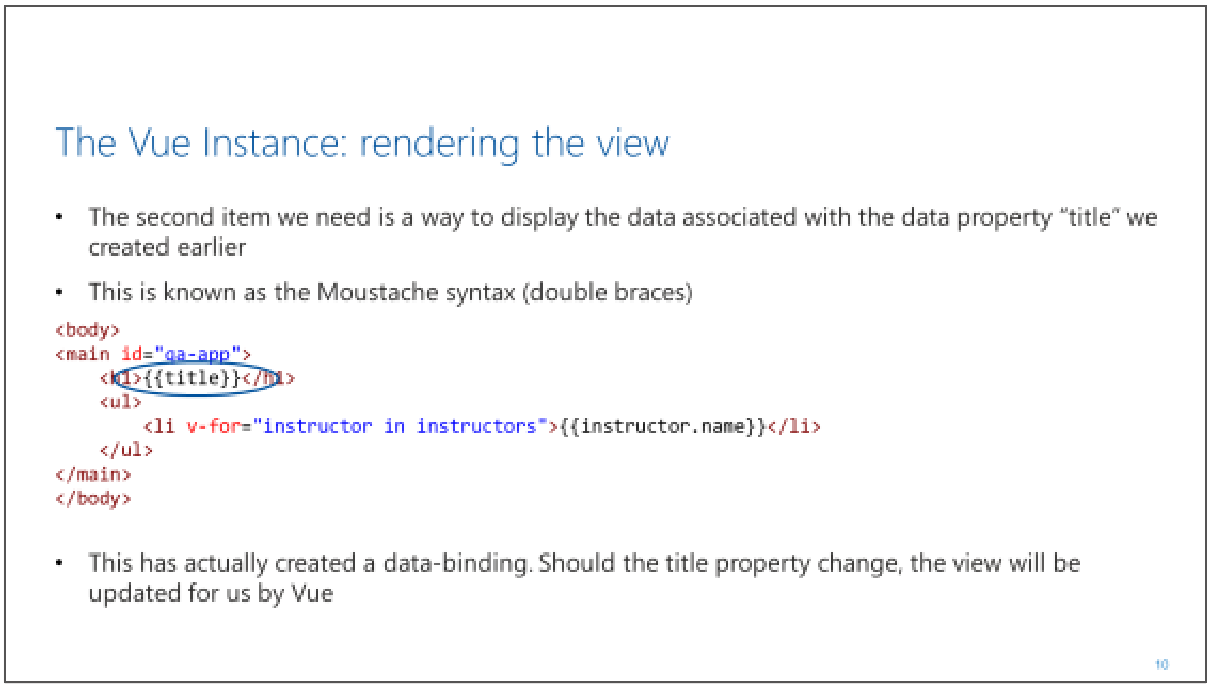


The Vue Instance: rendering the view

- To display this application we need some paired HTML
- The first item we need is an element with the id of "qa-app" in order to host our application

```
<body>
<main id="qa-app">
    <h1>{{title}}</h1>
    <ul>
        <li v-for="instructor in instructors">{{instructor.name}}</li>
    </ul>
</main>
</body>
```

9



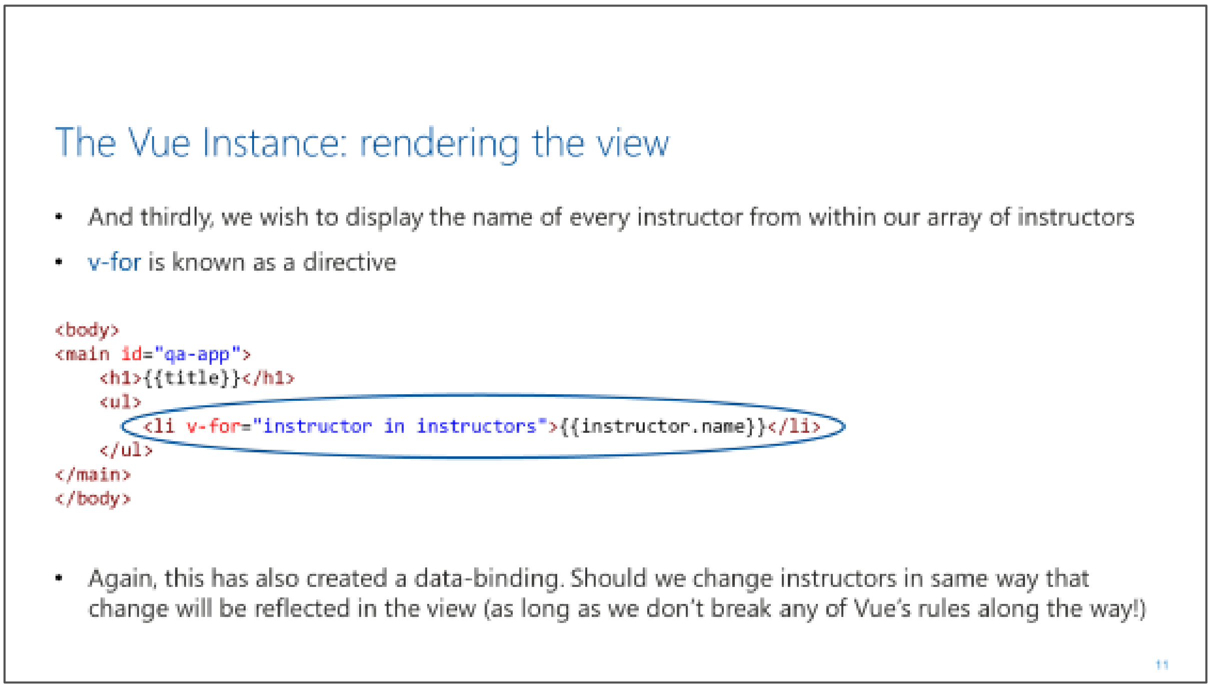
The Vue Instance: rendering the view

- The second item we need is a way to display the data associated with the data property “title” we created earlier
- This is known as the Moustache syntax (double braces)

```
<body>
<main id="qa-app">
    <h1>{{title}}</h1>
    <ul>
        <li v-for="instructor in instructors">{{instructor.name}}</li>
    </ul>
</main>
</body>
```

- This has actually created a data-binding. Should the title property change, the view will be updated for us by Vue

10



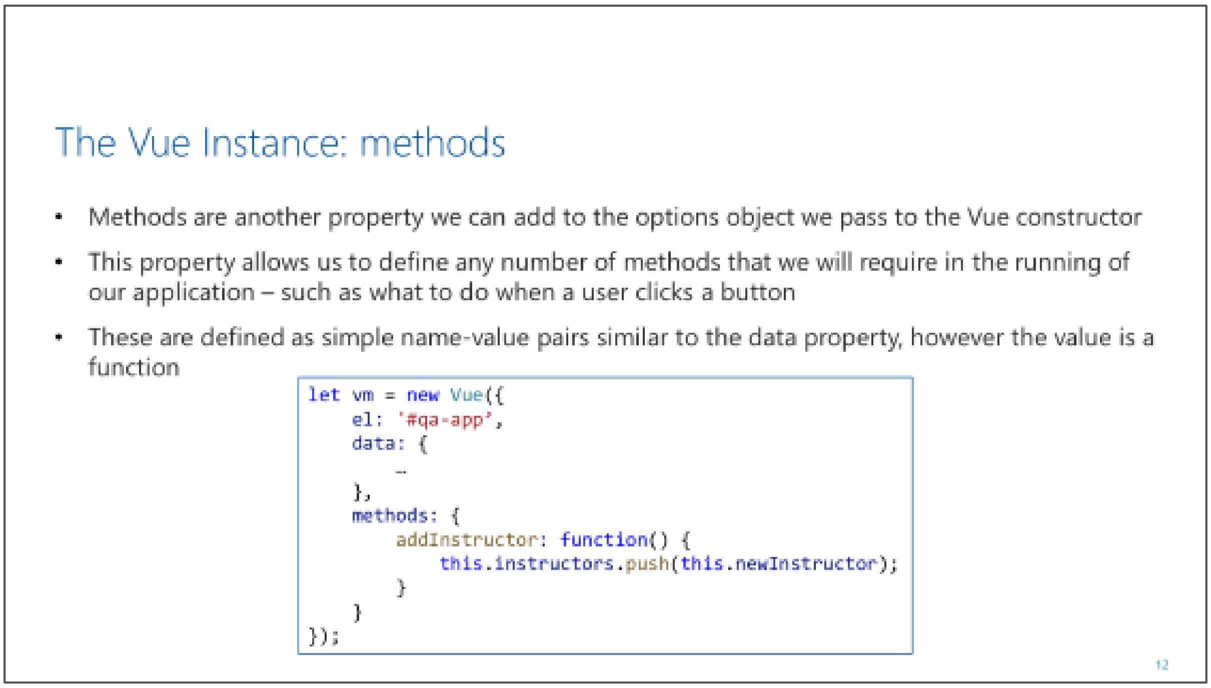
The Vue Instance: rendering the view

- And thirdly, we wish to display the name of every instructor from within our array of instructors
- v-for is known as a directive

```
<body>
<main id="qa-app">
    <h1>{{title}}</h1>
    <ul>
      <li v-for="instructor in instructors">{{instructor.name}}</li>
    </ul>
</main>
</body>
```

- Again, this has also created a data-binding. Should we change instructors in same way that change will be reflected in the view (as long as we don't break any of Vue's rules along the way!)

11

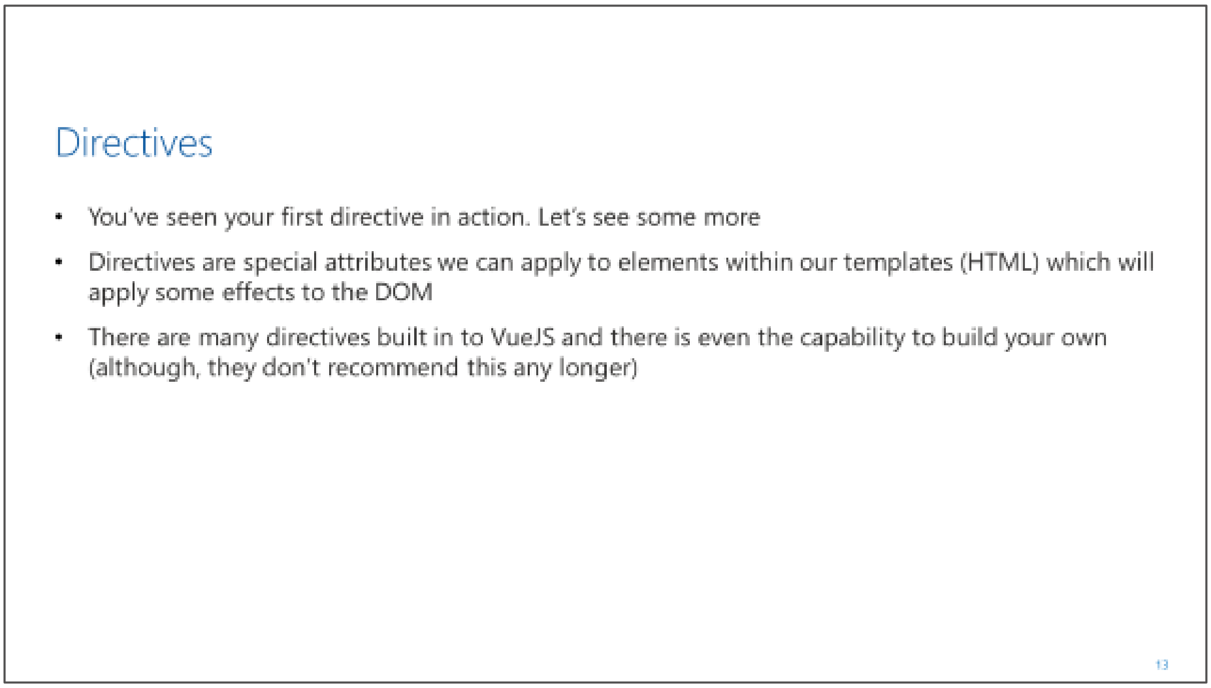


The Vue Instance: methods

- Methods are another property we can add to the options object we pass to the Vue constructor
- This property allows us to define any number of methods that we will require in the running of our application – such as what to do when a user clicks a button
- These are defined as simple name-value pairs similar to the data property, however the value is a function

```
let vm = new Vue({
    el: '#qa-app',
    data: {
        …
    },
    methods: {
        addInstructor: function() {
            this.instructors.push(this.newInstructor);
        }
    }
});
```

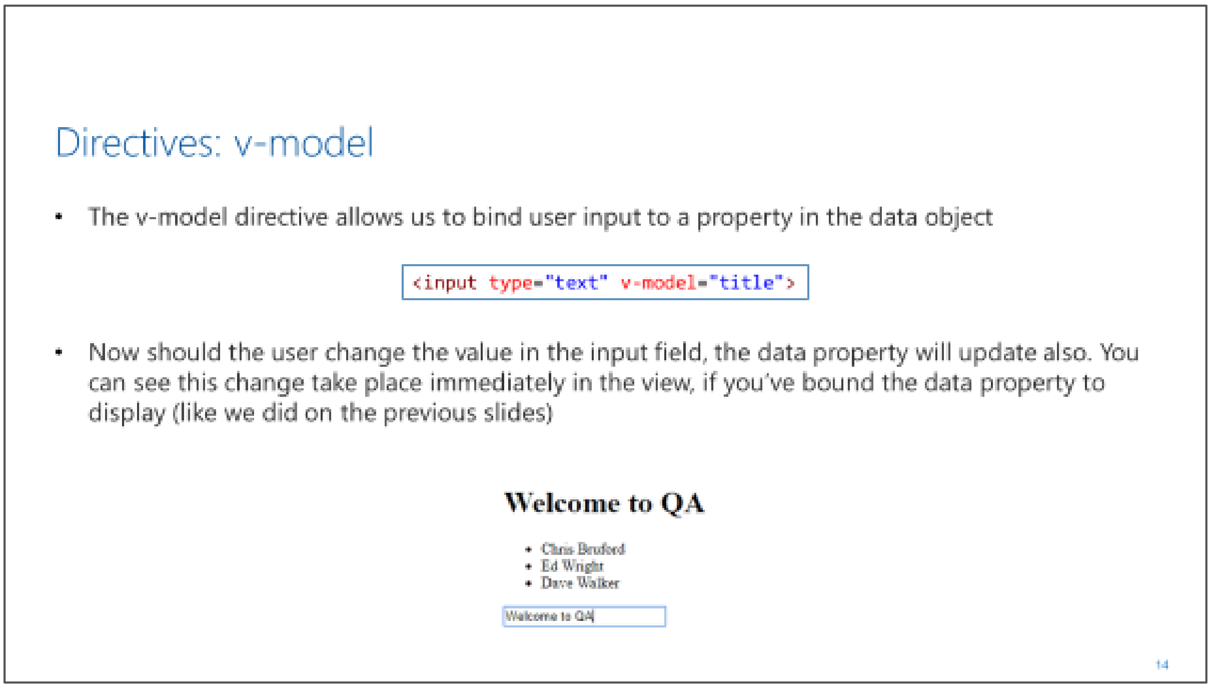
12



Directives

- You've seen your first directive in action. Let's see some more
- Directives are special attributes we can apply to elements within our templates (HTML) which will apply some effects to the DOM
- There are many directives built in to VueJS and there is even the capability to build your own (although, they don't recommend this any longer)

13



Directives: v-model

- The v-model directive allows us to bind user input to a property in the data object

```
<input type="text" v-model="title">
```

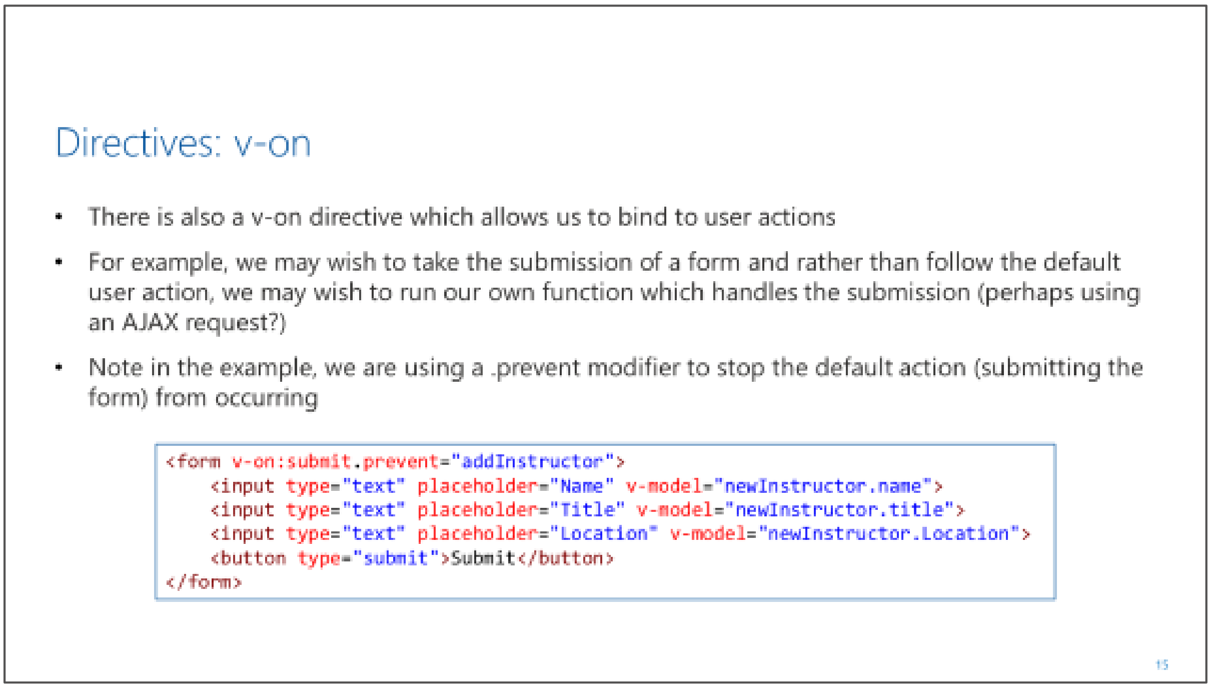
- Now should the user change the value in the input field, the data property will update also. You can see this change take place immediately in the view, if you've bound the data property to display (like we did on the previous slides)

Welcome to QA

- Chris Bruford
- Ed Wright
- Dave Walker

```
Welcome to QA
```

14



Directives: v-on

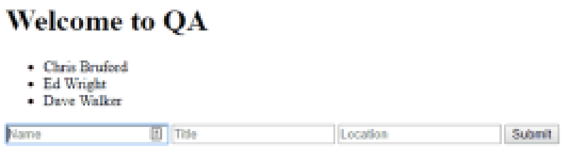
- There is also a v-on directive which allows us to bind to user actions
- For example, we may wish to take the submission of a form and rather than follow the default user action, we may wish to run our own function which handles the submission (perhaps using an AJAX request?)
- Note in the example, we are using a .prevent modifier to stop the default action (submitting the form) from occurring

```
<form v-on:submit.prevent="addInstructor">
    <input type="text" placeholder="Name" v-model="newInstructor.name">
    <input type="text" placeholder="Title" v-model="newInstructor.title">
    <input type="text" placeholder="Location" v-model="newInstructor.Location">
    <button type="submit">Submit</button>
</form>
```

15

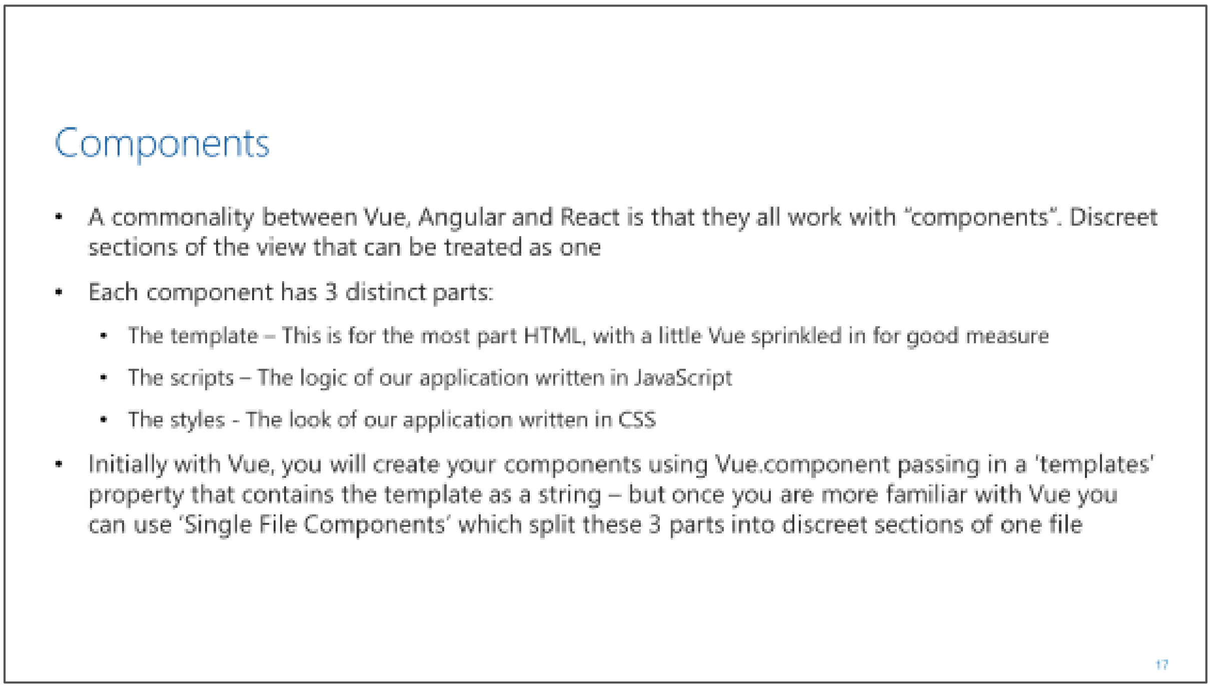
Directives: v-on

- There is also a v-on directive which allows us to bind to user actions
- For example, we may wish to take the submission of a form and rather than follow the default user action, we may wish to run our own function which handles the submission (perhaps using an AJAX request?)
- Note in the example, we are using a .prevent modifier to stop the default action (submitting the form) from occurring


Welcome to QA

- Chris Bruford
- Ed Wright
- Dave Walker

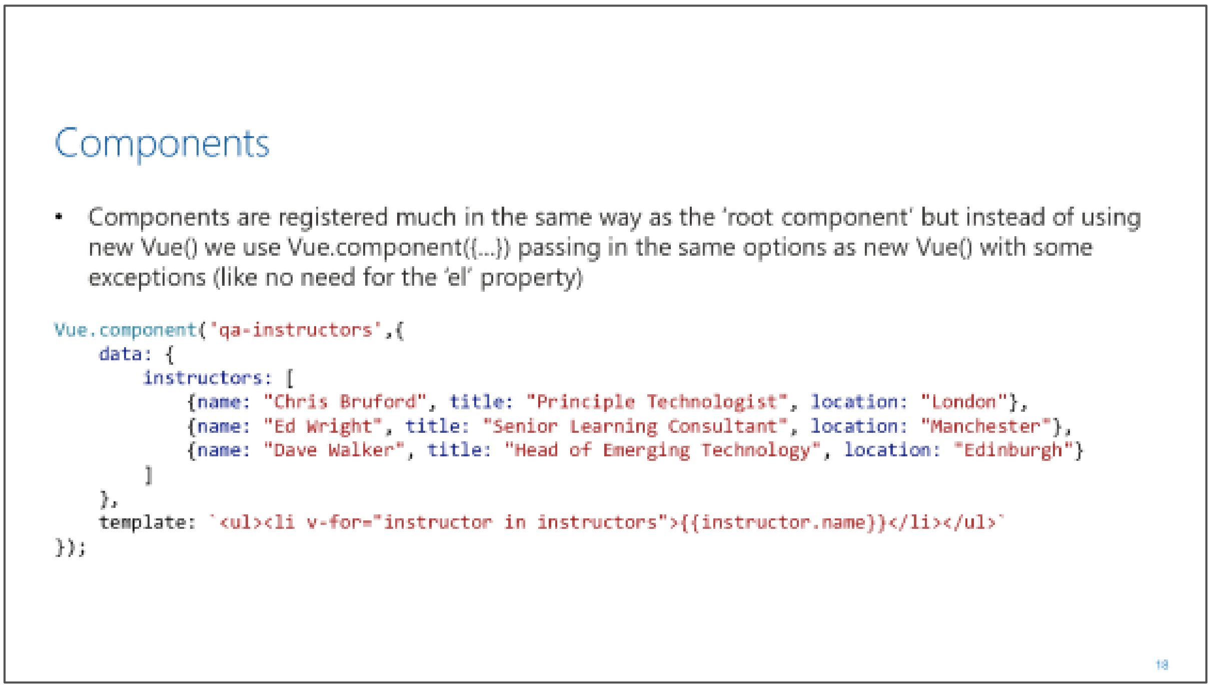
16



Components

- A commonality between Vue, Angular and React is that they all work with "components". Discreet sections of the view that can be treated as one
- Each component has 3 distinct parts:
  - The template – This is for the most part HTML, with a little Vue sprinkled in for good measure
  - The scripts – The logic of our application written in JavaScript
  - The styles - The look of our application written in CSS
- Initially with Vue, you will create your components using Vue.component passing in a 'templates' property that contains the template as a string – but once you are more familiar with Vue you can use 'Single File Components' which split these 3 parts into discreet sections of one file

17

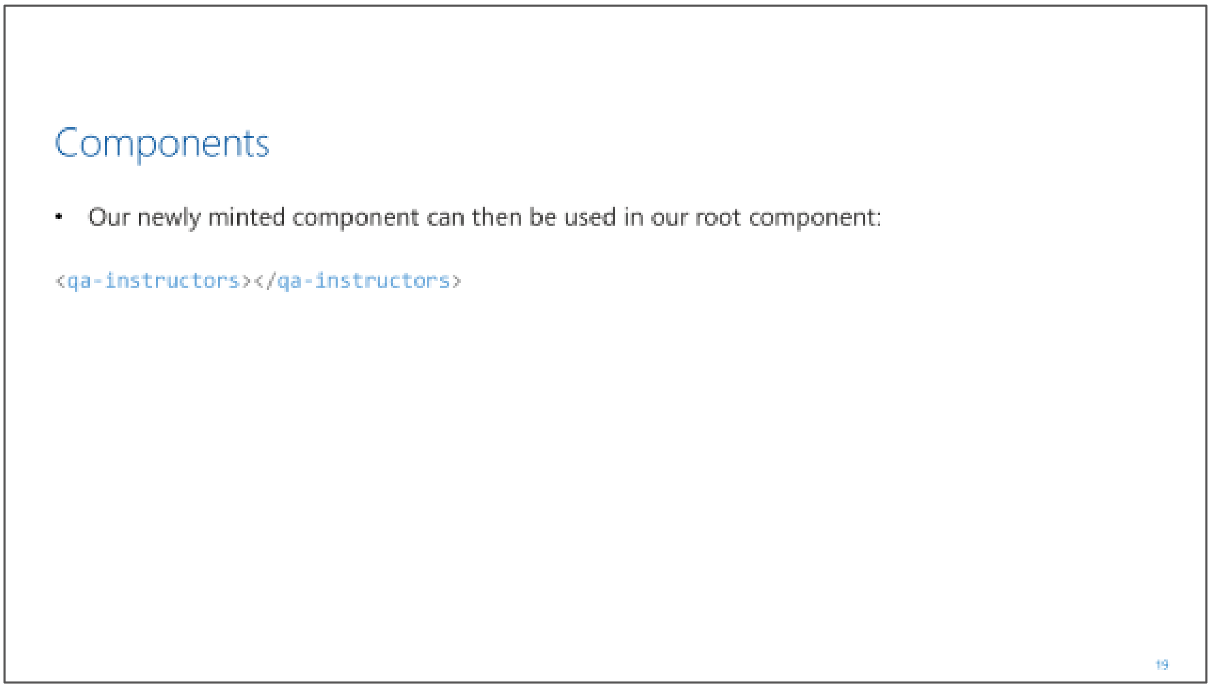


Components

- Components are registered much in the same way as the 'root component' but instead of using new Vue() we use Vue.component({...}) passing in the same options as new Vue() with some exceptions (like no need for the 'el' property)

```
Vue.component('qa-instructors',{
    data: {
        instructors: [
            {name: "Chris Bruford", title: "Principle Technologist", location: "London"},
            {name: "Ed Wright", title: "Senior Learning Consultant", location: "Manchester"},
            {name: "Dave Walker", title: "Head of Emerging Technology", location: "Edinburgh"}
        ]
    },
    template: `<ul><li v-for="instructor in instructors">{{instructor.name}}</li></ul>`
});
```

18

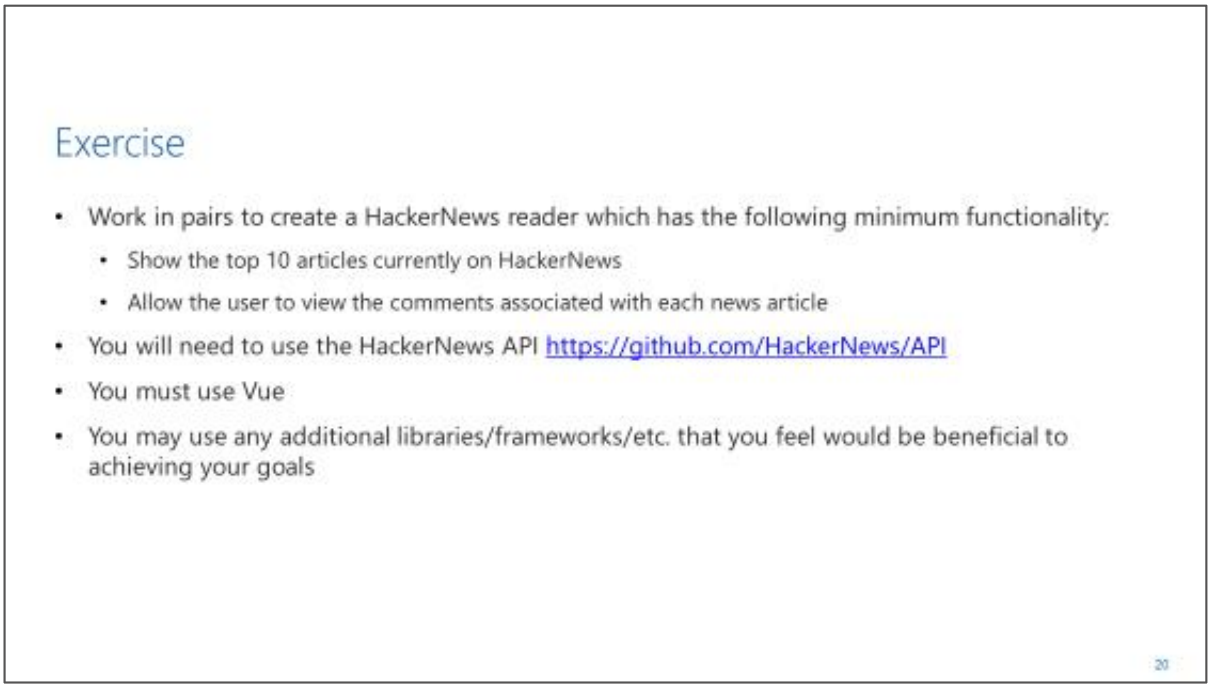


Components

- Our newly minted component can then be used in our root component:

```
<qa-instructors></qa-instructors>
```

19



Exercise

- Work in pairs to create a HackerNews reader which has the following minimum functionality:
  - Show the top 10 articles currently on HackerNews
  - Allow the user to view the comments associated with each news article
- You will need to use the HackerNews API https://github.com/HackerNews/API
- You must use Vue
- You may use any additional libraries/frameworks/etc. that you feel would be beneficial to achieving your goals

20