CS4220

ALBERT F. CERVANTES
CYDNEY AUMAN

# CURRENT TRENDS IN WEB DESIGN AND DEVELOPMENT

# AGENDA

- Course Ranking App
- Data Binding Review
- Computed Properties
- Event Handling
- Components and Props
- Chrome Dev Tools

# COURSE RANKING APP

- To highlight our examples, we will develop a course ranking application that allows Users to upvote various courses on your application.
- Some key features of the app are:
  - Load data from an external data source
  - List and sort all courses in order of # of votes
  - Up-voting
  - Highlighting courses with many votes

# DATA BINDING REVIEW

- Code Example
- No Binding

```
<article class="media">
  <figure class="media-left">
    <img class="image is-64x64" src="../images/courses/cs4220.jpg">
  </figure>
  <div class="media-content">
    <div class="content">
      <p>
        <strong>
          <a href="#" class="has-text-info">CS4220</a>
          <span class="tag is-small">#4</span>
        </strong>
        <br> Current Trends in Web Design and Development
        <br>
        <small class="is-size-7">
          Submitted by:
          <img class="image is-24x24" src="../images/avatars/256_1.png">
        </small>
      </p>
    </div>
  </div>
  <div class="media-right">
    <span class="icon is-small">
      <i class="fa fa-chevron-up"></i>
      <strong class="has-text-info">10</strong>
    </span>
  </div>
</article>
```

# DATA BINDING REVIEW

- Code Example
- Binding using v-for

```
<article class="media" v-for="course in courses">
  <figure class="media-left">
    <img class="image is-64x64" v-bind:src="course.courseImage">
  </figure>
  <div class="media-content">
    <div class="content">
      <p>
        <strong>
          <a v-bind:href="course.url" class="has-text-info">
            {{course.title}}
          </a>
          <span class="tag is-small">{{course.id}}</span>
        </strong>
        <br> {{course.description}}
        <br>
        <small class="is-size-7">
          Submitted by:
          <img class="image is-24x24" v-bind:src="course.avatar">
        </small>
      </p>
    </div>
  </div>
  <div class="media-right">
    <span class="icon is-small">
      <i class="fa fa-chevron-up"></i>
      <strong class="has-text-info">{{course.votes}}</strong>
    </span>
  </div>
</article>
```

# COMPUTED PROPERTIES

- Computed properties are used to perform calculations within the Vue instance.
- The result can then be rendered in the view

```
new Vue({
    el: '#app',
    data: {
      courses: Seed.courses
    },
    computed: {
      sortedCourses () {
        return this.courses.sort((a, b) => {
          return b.votes - a.votes
        });
      }
    }
  }
});
```

# COMPUTED PROPERTIES

- Computed properties are used to perform calculations within the Vue instance.
- The result can then be rendered in the view

```
<article class="media" v-for="course in courses”>
```

```
<article class="media" v-for="course in sortedCourses">
```

# EVENT HANDLING

- To handle an upvote, we will define a click event on the chevron using the v-on directive.

```
<div class="media-right">
  <span class="icon is-small" v-on:click="upvote(course.id)">
    <i class="fa fa-chevron-up"></i>
    <strong class="has-text-info">{{course.votes}}</strong>
  </span>
</div>
```

# EVENT HANDLING

- To handle an upvote, we will define a click event on the chevron using the v-on directive.
- A shorthand for v-on is the @ symbol.

```
<div class="media-right">
  <span class="icon is-small" @click=“upvote(course.id)">
    <i class="fa fa-chevron-up"></i>
    <strong class="has-text-info">{{course.votes}}</strong>
  </span>
</div>
```

# EVENT HANDLING

- Next, we must define the event handler in the Vue instance.
- Note: Some code is omitted for simplicity.

```
new Vue({
    el: '#app',
    …
    methods: {
      upvote: function(courseId) {
        const course = this.courses.find(
          course => course.id === courseId
        );
        course.votes++;
      }
    }
});
```

# CLASS BINDING

- Next, lets conditionally apply a class to each course (article) whenever the number of votes is greater than 20

```
<article class="media"
         v-for="course in sortedCourses"
         v-bind:class="{ 'blue-border': course.votes >= 20}">
  <!-- The rest of the article HTML -->
</article>
```

# CLASS BINDING

- Next, lets conditionally apply a class to each course (article) whenever the number of votes is greater than 20

```
<article class="media"
         v-for="course in sortedCourses"
         v-bind:class="{ 'blue-border': course.votes >= 20}">
  <!-- The rest of the article HTML -->
</article>
```

- What happens to the `media` class that was initially applied to the `article`?

# COURSE RANKING APP – WHAT'S NEXT?

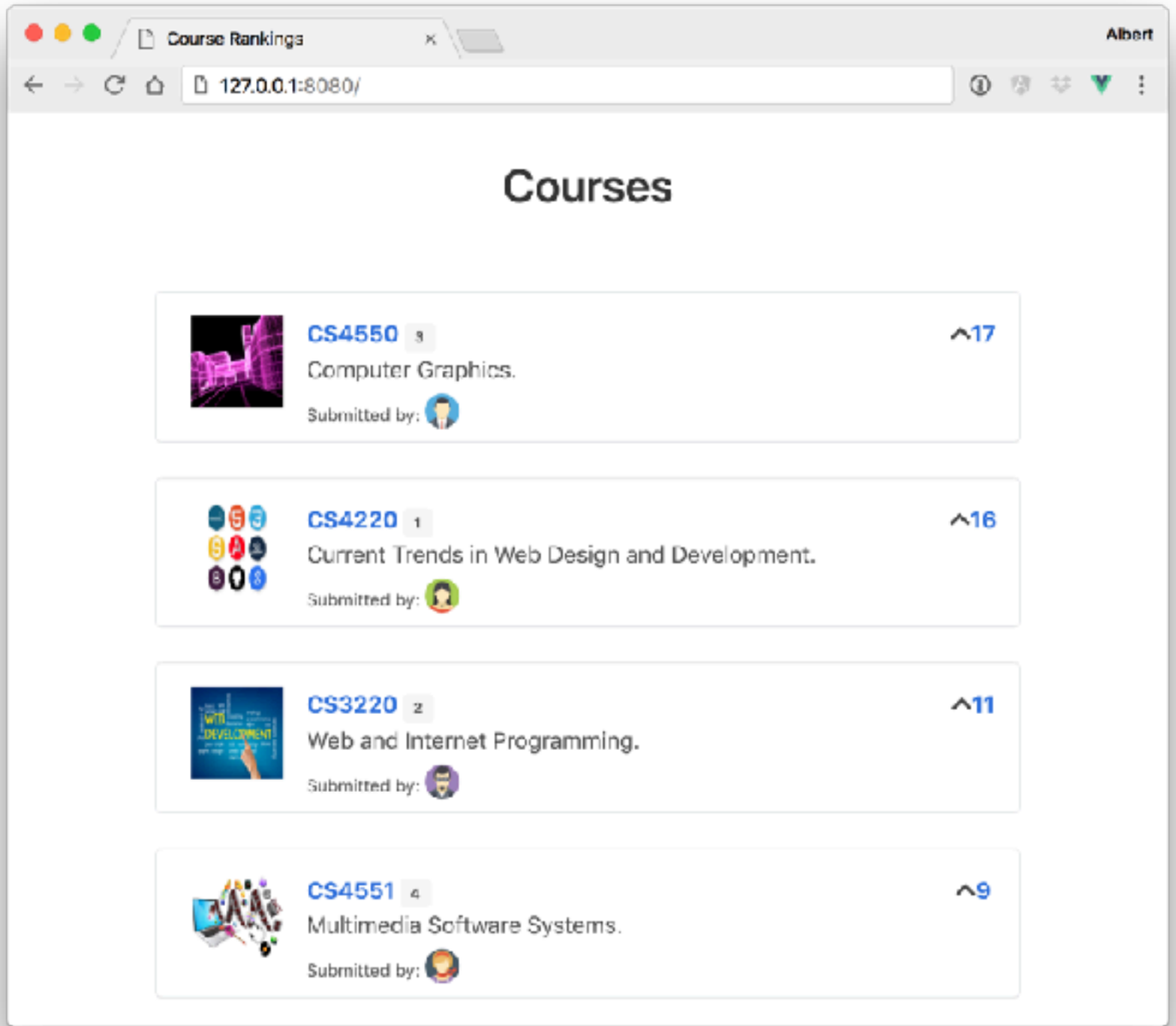
- We've completed our application by implementing all of the functionality we initially defined.
- How can we expand the functionality?
  - Header/Navbar, Sidebar to add new courses, Footer, etc…
- Using our current approach will bloat our HTML, and force us to add many more methods and datum to our Vue instance
- Solution? Isolated Components

# COMPONENTS

- Vue provides users the ability to create isolated components within their application
- Components are intended to be self-contained modules that  group markup (HTML), logic (JS), and even styles (CSS) within them.
- Motivation? Reusability and Maintainability

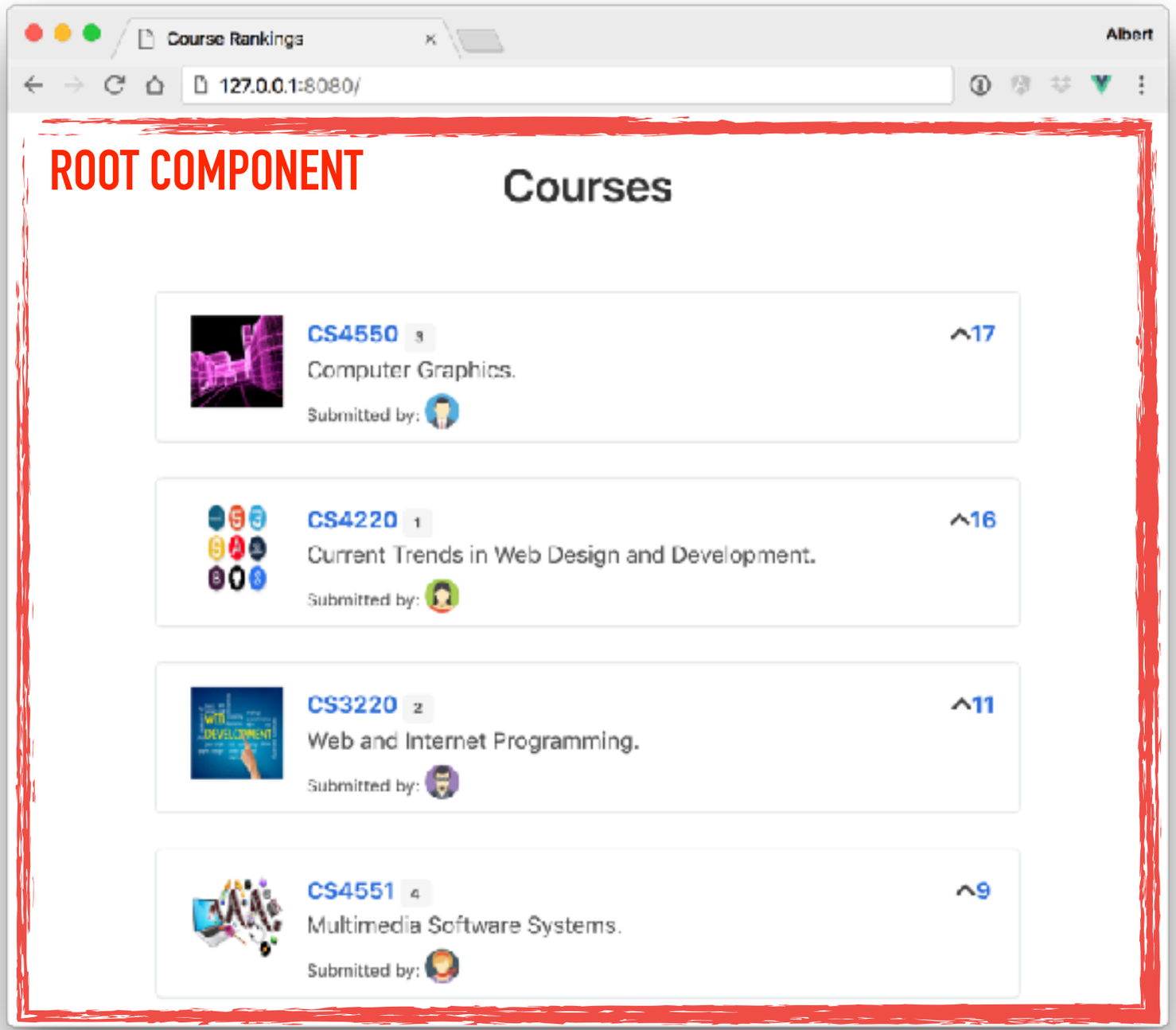
# DIVIDE AND CONQUER

- Let's divide our application into two separate components



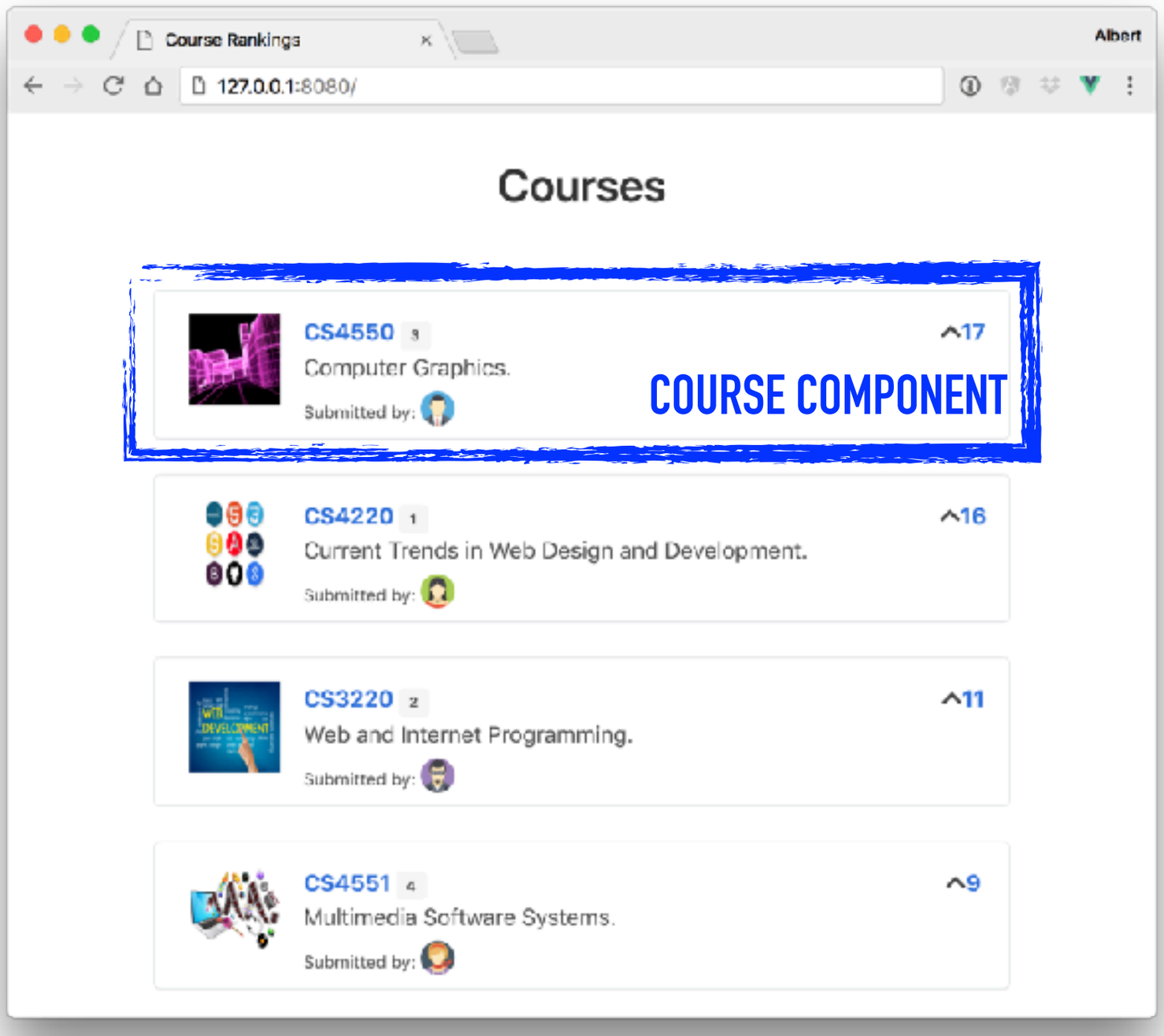
# DIVIDE AND CONQUER

- The Root Component.
- This is our existing Vue instance.



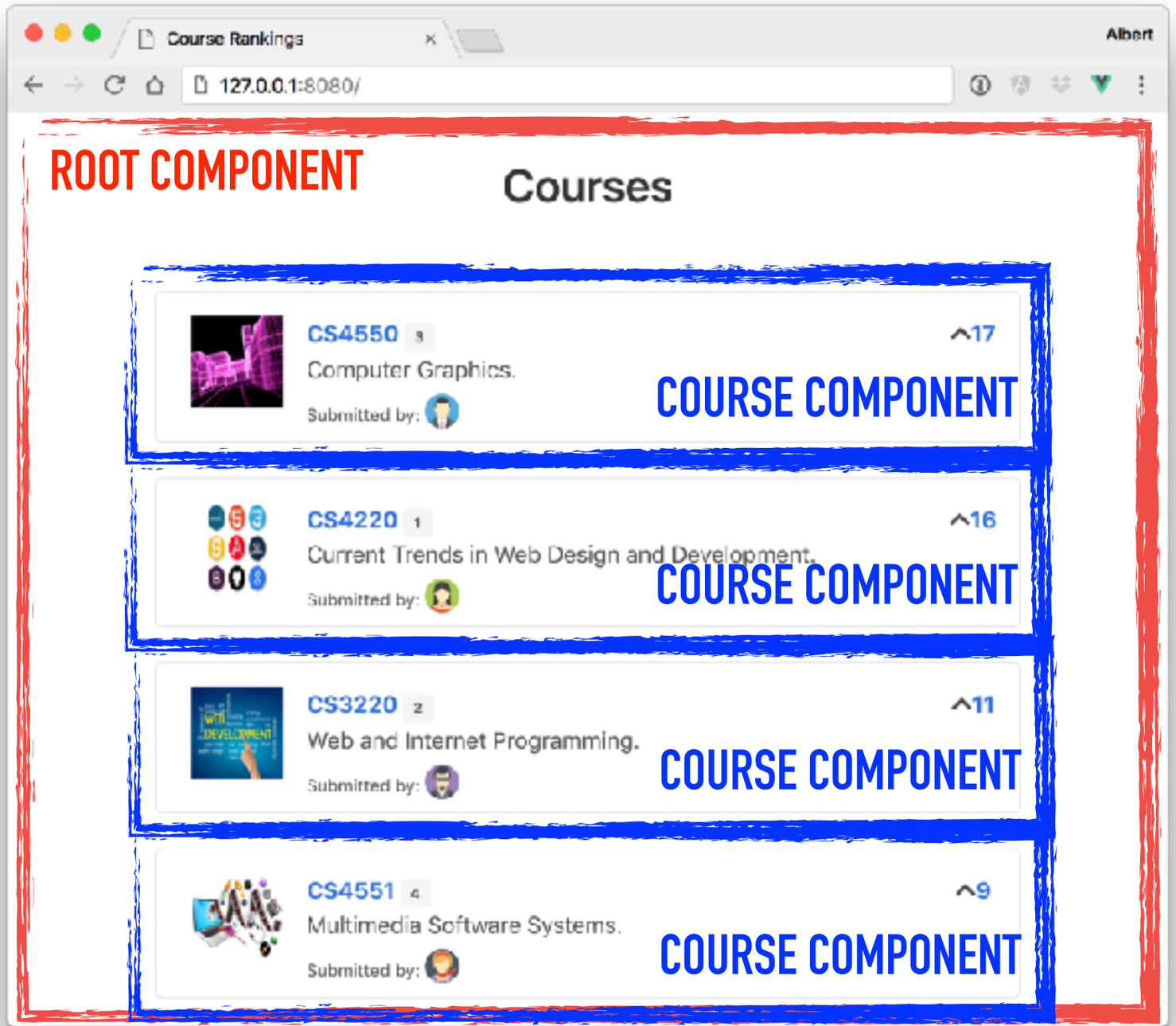
# DIVIDE AND CONQUER

- The Course Component.
- This will be a new component representing a single course.



# DIVIDE AND CONQUER

- Ultimately, our goal is to re-use the course component for every course we are displaying.



# CREATING A COMPONENT

- Creating a Global Vue Component

```
Vue.component('course-component', {
  // options
});
```

- Although this would work, we are instead going to define our component within the context of our application.

# CREATING A COMPONENT

- We will create a new `course-component` and register it with our main Vue instance using the `component` option.

```
const courseComponent = {

};

new Vue({
  // ...
});
```

# CREATING A COMPONENT

- Vue components *are* Vue instances.
- We can define the template of a component using strings.
  - Template literals are useful when defining our templates.

```
const courseComponent = {
  template:
  ` <div>
      Hello from our Course Component!
    </div>`
}
```

- *NOTE: We're writing templates for a Vue application that isn't being precompiled. Next lecture we'll define component templates that will be precompiled during build.*

# CREATING A COMPONENT

- For our course component, we need our template to represent all HTML elements that are contained within a single course.

```
const courseComponent = {
  template:
  `
  <div style="display: flex; width: 100%”>
    <figure class="media-left">
      <img class="image is-64x64" v-bind:src="course.courseImage">
    </figure>
    <div class="media-content">
      <div class="content">
        <p>
          <strong>
            <a v-bind:href="course.url" class="has-text-info">{{course.title}}</a>

     …The rest of the template has been omitted for readability
```

# CAVEATS WHEN CREATING A COMPONENT

- Vue imposes a strict limitation requiring the template of a component be enclosed within a single root element.
  - This is why the template is wrapped with a single `div`
- The `course` object in this template is currently `undefined`.
  - We will use Vue props to pass the data from the root component down to this component when the component is declared.
- The upvote() click listener method needs to be migrated to the courseComponent for it to work.

# USING A COMPONENT

- We replace the inner content of our article with the course component we just created.

```
<article class="media"
         v-for="course in sortedCourses"
         v-bind:class="{ 'blue-border': course.votes >= 20}">
  <course-component></course-component>
</article>
```

- *Our Vue instance doesn't know what a `course` component is…yet!*

# USING A COMPONENT

- We define the `course` component as a key in the components property of our Vue instance to give the instance awareness of our custom component

```
new Vue({
    // …
     components: {
       'course-component': courseComponent
     }
   });
```

# PROPS

- In Vue, data is passed from a parent component to a child component using props.
- Props can only flow in a single direction: parent to child.
- The `v-bind` directive is used to bind dynamic values (or objects) as props in a parent instance.

# BINDING PARENT DATA TO PROPS

- We will pass the course and sortedCourses objects to the `course-component`.
  - The `course` object will be used in the template of the `course-component`.
  - The `sortedCourses` array will be used in the `upvote` function of the `course-component`.
- To do so, we modify the course-component as follows:

```
<article class="media"
        v-for="course in sortedCourses"
        v-bind:class="{ 'blue-border': course.votes >= 20}">
        <course-component
          v-bind:course="course"
          v-bind:courses=“sortedCourses">
        </course-component>
</article>
```

# USING PROPS IN A CHILD COMPONENT

- A child component needs to explicitly declare the props it receives with the props option.

```
const courseComponent = {
  // ...
  props: ['course', 'courses']
}
```

- Now the `course` object and `courses` array can be safely used in the course component.

# MIGRATING UPVOTE

- Next, we must migrate the upvote function to the `course` component.
- Note, that the `courses` prop becomes part of the courseComponent instance, so using `this.courses` allows us to directly access the courses that were passed to our component.

```
const courseComponent = {
  template:
  `<div style="display: flex; width: 100%">
    ...
  </div>`,
  props: ['course', 'courses'],
  methods: {
    upvote: function(courseId) {
      const course = this.courses.find(
        course => course.id === courseId
      )
      course.votes++
    }
  }
}
```

# MORE SHORTHAND

- Shorthand for the v-on directive is an @

```
<span v-on:click="upvote(course.id)">
```

```
<span @click="upvote(course.id)">
```

- Shorthand for the v-bind directive is a colon :

```
<img v-bind:src="course.courseImage">
```

```
<img :src="course.courseImage">
```

# VUE DEV TOOLS

- Vue provides a useful Google Chrome Extension
  - https://github.com/vuejs/vue-devtools
- Simplifies debugging applications
- Note: We'll be working with applications opened via file:// protocol. To make the Vue devtools work for these pages, you'll need to check "Allow access to file URLs" for the extension in Chrome's extension manager:
-