Ember 2

Section 4 - Templates

Handlebars Basics
Built-in Helpers
Conditionals and Looping
Binding HTML Element Attributes
Links, Actions
Custom Helpers

Section Objectives

- Understand the Handlebars templating syntax
- Be able to use Ember's built-in helpers
- Be able to conditionally display markup on the page
- Be able to display markup for each item in a collection
- Be able to link to other routes
- Be able to define and execute actions in a component or controller
- Be able to create a custom helper

Templates

Handlebars Basics

Handlebars Basics
Built-in Helpers
Conditionals and Looping
Binding HTML Element Attributes
Links, Actions
Custom Helpers

Handlebars - Vanilla

- A templating framework for modularizing sections or defining whole pages which uses data to drive the final markup

```
<script id="index" type="text/x-handlebars-template">
    <h1>Hello, {{name}}!</h1>
</script>
<script>
    let source = document.querySelector("#index").innerHTML;
    let template = Handlebars.compile(source);
    let context = {name: "Joe"};
    document.write(template(context));
</script>
```

Handlebars - Ember

application.js

```
import Ember from 'ember';

export default Ember.Controller.extend(
        { firstName: 'Joe', lastName: 'Rizzo' }
);
```

application.hbs

```
 Hello, <strong>{{firstName}} {{lastName}}</strong>!
```

Templates

Built-in Helpers

Handlebars Basics
Built-in Helpers
Conditionals and Looping
Binding HTML Element Attributes
Links, Actions
Custom Helpers

Built-in Helpers

- {{get}} helper

```
{{get address part}} => this.get('address.city'); //part = city
                                    or
                        this.get('address.zip'); //part = zip
```

- Nesting built-in helpers

```
{{get "address" (concat "street" index)} =>
this.get('address.street1'); // index = 1
```

Templates

Conditionals and Looping

Handlebars Basics
Built-in Helpers
Conditionals and Looping
Binding HTML Element Attributes
Links, Actions
Custom Helpers

Conditional Block Helpers

- #if

```
{{#if model.singleNote}}
    note
{{/if}}
```

- #if-else

```
{{#if model.singleNote}}
    note
{{else}}
    notes
{{/if}}
```

- #unless

```
{{#unless model.selectedNote}}
    Add Note
{{/unless}}
```

Looping and Other Block Helpers

- #each

```
{{#each model.notes as |note|}}
    {{note-row note=note selectNote=(action "showNote")}}
{{/each}}
```

- #with

```
{{#each model.notes as |note|}}
    {{#with note}}
        <h1>{{title}} - {{date}}</h1>{{content}}
    {{/with}}
{{/each}}
```

Templates

Binding HTML Element Attributes

Handlebars Basics
Built-in Helpers
Conditionals and Looping
Binding HTML Element Attributes
Links, Actions
Custom Helpers

Binding HTML Elements

```
<img src={{logoUrl}} alt="Logo">
```

```
<input type="checkbox" disabled={{isDisabled}}>
```

Templates

Links, Actions, Helpers

Handlebars Basics
Built-in Helpers
Conditionals and Looping
Binding HTML Element Attributes
Links, Actions, Helpers
Custom Helpers

Links

- {{#link-to <route> <model>}}

router.js

```
Router.map(function() {
  this.route('note', {path: '/note/:note_id'});
});
```

application.hbs

```
{{#each model.notes as |note|}}
    {{#link-to 'note' note.id}}
        {{note-row note=note showNote=(action "showNote")}}
    {{/link-to}}
{{/each}}
```

index.html (compiled)

```
<div class="notes">
    <a href="/note/0">…</a>
    <a href="/note/1”>…</a>
    <a href="/note/2">…</a>
    <a href="/note/3">…</a>
</div>
```

Actions

- {{action “<method_name>”}}
- onClick is the default event trigger

application.hbs

```
<div {{action ‘toggleEdit’}}>
    {{#if editMode}}
        Clear
    {{else}}
        Edit
    {{/if}}
</div>
```

application.js

```
export default Ember.Component.extend({
    actions: {
        toggleEdit() {
            this.toggleProperty(‘editMode’);
        }
    }
});
```

Action Parameters

- {{action “<method_name>” <param>}}

application.hbs

```
<div {{action ‘toggleEdit’ master}}>
    {{#if editMode}}
        {{master.cancel_title}}
    {{else}}
        {{master.edit_title}}
    {{/if}}
</div>
```

Specifying an Action Event Type

- {{action “<method_name>” <param> on=“<event>”}}

application.hbs

```
<div {{action ‘toggleEdit’ master on=“mouseover”}}>
    {{#if editMode}}
        {{master.cancel_title}}
    {{else}}
        {{master.edit_title}}
    {{/if}}
</div>
```

Lab 1 – Conditional Block Helpers

- We need the ability to toggle between ’edit’ and ‘selection’ mode with each note. Clicking on the ‘Edit’ button should reveal check circles to the left of the note that can be selected, and reveal the trash bucket on the right side of the toolbar.

- Currently the trash bucket doesn’t exist and only the add icon is visible. We want to toggle between these two.

1. Implement the appropriate conditional block helpers and actions to toggle between the Edit button displaying Cancel when in ‘editMode’ and Edit when not. Also, only show the trash bucket if ’editMode’ is true. Don’t forget to add a property to the model to handle this.

Lab 1 – Conditional Block Helpers

2. When ‘editMode’ is true, we need to show check circles to the left of the note to be selected to indicate these will be deleted. Update the ‘note-row’ component to have an ’editMode’ attribute that takes in the model’s editMode value. In the ‘note-row’ component, add an editMode property to the source and an if-block condition to toggle the check circles visible. Add the following HTML above the ‘content’ in the ’note-row’ template:

note-row.hbs

```
<div class="edit">
    <div>{{fa-icon "circle-thin"}}</div>
</div>
```

Lab 1 – Conditional Block Helpers

3. If a circle is selected, it is to show a ‘check-circle’. Use an if-else block helper and necessary action to choose between displaying a ‘circle-thin’ or ‘check-circle’ icon from Font Awesome.

note-row.hbs

```
<div class="edit">
    <div>{{fa-icon ”check-circle"}}</div>
    <div>{{fa-icon "circle-thin"}}</div>
</div>
```

Lab 1 – Conditional Block Helpers

4. We're getting close but not exactly. If we are in 'editMode', we don't want to change the currently selected row. Implement code to make sure we don't change the currently selected row, if one is selected while in 'editMode'. Also, ensure we don't select a row if one wasn't previously selected.

5. Currently, if we have selected any of the circles while in 'editMode', that state is stored if we Cancel and go back into 'editMode'. Clear the state of selected check circles if cancelled.

6. Lastly, we want to change the title of the Master View to show the number of selected rows while in 'editMode'.

Templates

Custom Helpers

Handlebars Basics
Built-in Helpers
Conditionals and Looping
Binding HTML Element Attributes
Links, Actions
Custom Helpers

Custom Helpers

- ***ember g helper format-date***

```
{{format-date model.date}}
```

format-date.js

```
export function formatDate([params]/*, hash*/) {

..

return passed_date.toDateString() +
        " "+ passed_date.getHours() +
        ":"+ passed_date.getMinutes() +
        ":"+ passed_date.getSeconds();
}
```

Lab 2 – Custom Helper

- Create a custom helper, ‘format-date’, and apply it to all locations where the date is displayed to the screen. Implement your own personalized date format to display.

Section Review

- Understand the Handlebars templating syntax
- Be able to use Ember's built-in helpers
- Be able to conditionally display markup on the page
- Be able to display markup for each item in a collection
- Be able to link to other routes
- Be able to define and execute actions in a component or controller
- Be able to create a custom helper

End of Section

- This slide is purposely devoid of any information