DEV PREVIEW

Step 4: Finishing touches

Your first Polymer application

Edit on GitHub

Step 4: Finishing touches

In this section, you'll finish up the app by adding a favorite button to the cards and connecting the tabs to the `<post-list>` control.

In this section you'll learn about:

- Declarative event handling.
- Adding properties and methods to the element's prototype.
- Automatic node finding.

Edit post-card.html

Open `post-card.html` in your editor and add the `<core-icon-button>` element:

```
<div class="card-header" layout horizontal center>
  <content select="img"></content>
  <content select="h2"></content>
</div>

<core-icon-button
  id="favicon"
  icon="favorite"
  on-tap="{{favoriteTapped}}">
</core-icon-button>

<content></content>
```

Key information

- As the name implies, `<core-icon-button>` creates a button with an embedded icon. Polymer includes several sets of scalable icons.
- The `icon="favorite"` attribute selects the heart icon from the default icon set.
- The `on-tap="{{favoriteTapped}}"` attribute specifies a method to call on the `post-card` element when the button is

tapped.

Add the `favorite` property and `favoriteTapped` method to the element's prototype.

```
<script>
Polymer({
  publish: {
    favorite: {
      value: false,
      reflect: true
    }
  },
  favoriteTapped: function(event, detail, sender) {
    this.favorite = !this.favorite;
    this.fire('favorite-tap');
  }
});
</script>
```

- The `publish` object is another way to specify published properties, like the `attributes` attribute shown in Step 3. Here the `favorite` property defaults to `false`, and it *reflects*, meaning the

`favorite` attribute is updated in the DOM whenever the property value changes.

- The `favoriteTapped` event toggles the state of the `favorite` property (`this.favorite`), and also fires a custom event, using the built in `fire` method. (`fire` is one of several utility methods Polymer adds to the prototype of every custom element.)

The net result of these changes is that when the favorite button is tapped, the favorite property is updated and its corresponding attribute is set or unset.

Right now, there's no visual indication that the button is pressed.

Add the following CSS to style the favorite button:

```
core-icon-button {
  position: absolute;
  top: 3px;
  right: 3px;
  color: #636363;
}
:host([favorite]) core-icon-button {
  color: #da4336;
}
```

```
</style>
```

- The `color` property sets the fill color on the icon.
- The `:host([favorite]) core-icon-button` selector sets the fill color when the `favorite` attribute is set on the custom element.

Save `post-card.html`.

At this point, you can reload the page and your favorite buttons should work, but there are still a few steps left to finish the app.

Edit index.html

Open `index.html` and update the tab event handler to switch views in `<post-list>` when the user switches tabs:

```
<script>
var tabs = document.querySelector('paper-tabs');
var list = document.querySelector('post-list');

tabs.addEventListener('core-select', function() {
  list.show = tabs.selected;
```

```
});
</script>
```

Save `index.html`.

Edit post-list.html

Open `post-list.html` in your editor.

Update the template that creates the `<post-card>` elements to wire up the favorites:

```
<template repeat="{{post in posts}}">

  <post-card
    favorite="{{post.favorite}}"
    on-favorite-tap="{{handleFavorite}}"
    hidden?="{{show == 'favorites' && !post.favorite}}">
    <img src="{{post.avatar}}" width="70" height="70">
    <h2>{{post.username}}</h2>
    <p>{{post.text}}</p>
  </post-card>
</template>
```

- `favorite="{{post.favorite}}"` binds the card's `favorite` value to the value in the array owned by the `<post-service>`.
- The `on-favorite-tap` attribute sets up a handler for the `favorite-tap` event fired by the `<post-card>`.
- The `hidden?="{{}}"` expression is special syntax for a boolean attribute, which sets the attribute if the binding expression evaluates to true.

The binding expression for `hidden` actually does the work of switching between the All and Favorites tabs. The `hidden` attribute is a standard HTML5 attribute. The default Polymer style sheet includes a rule to style `hidden` as `display: none` for those browsers that don't support `hidden` natively.

Add an event handler for the `favorite-tap` event to `post-list.html`:

```
<script>
Polymer({
  handleFavorite: function(event, detail, sender) {
    var post = sender.templateInstance.model.post;
    this.$.service.setFavorite(post.uid, post.favorite);
  }
});
</script>
```

```
</script>
```

Key information

- `sender.templateInstance.model` is a reference to the model data used to construct a template instance. In this case, it includes the `post` object used to create a `<post-card>`, so you can retrieve its ID and `favorite` value.

- `this.$.service` returns a reference to the `<post-service>` element. Every element in a custom element's shadow DOM that

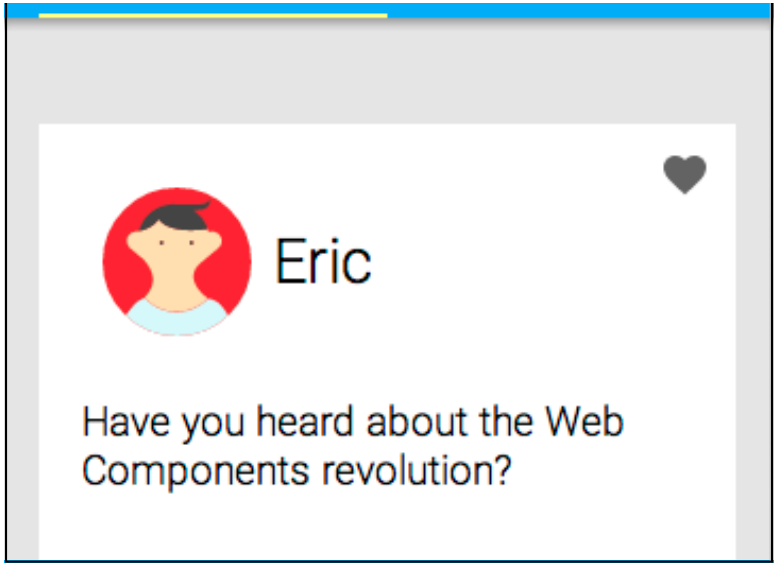
  has an `id` attribute is added to the `this.$` dictionary. This is called automatic node finding.

- If this was a real social networking service, the `setFavorite` method would persist the change to the server. As is, it doesn't do anything other than log a console message.

Finished!

Save `post-list.html` and refresh your page.

That's it — you're done! With a bit of luck, your application looks like this:



Click screenshot for demo

If your project doesn't look quite right, check your work against the files in the `finished` directory:

- `post-card.html`
- `post-list.html`
- `index.html`

Start your next project

Ready to start a project of your own? Install some Polymer components and get to work!

← STEP 3: USING DATA BINDING

→ INSTALLING COMPONENTS

+POLYMER @POLYMER

/POLYMER FILE A BUG