DEV PREVIEW

Step 2: Creating your own element

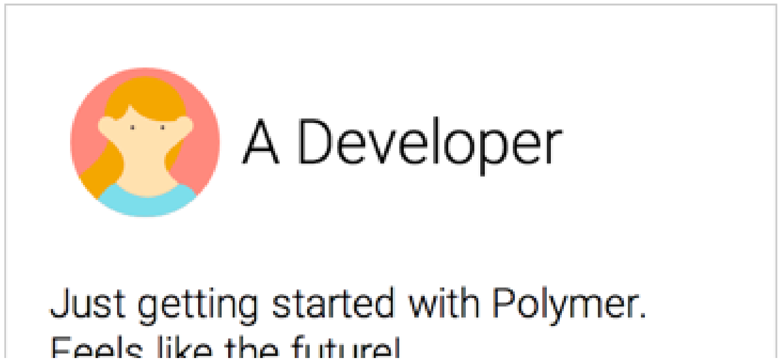
Your first Polymer application

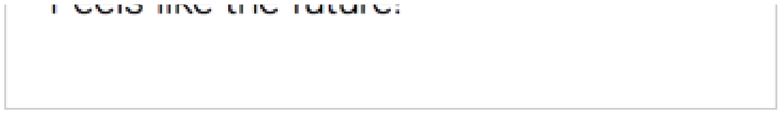
Edit on GitHub

▶ Table of contents

Step 2: Your own element

Now that you have a basic application structure, you can start building a card element to display a post. The finished card includes space for a profile picture, name, favorite button, and a content area.





In this step, you'll create a `<post-card>` element that controls the layout and styling of its children, so you can create a card like the one above using simple markup like this:

```
<post-card>
  <img src="profile-picture.png">
  <h2>A Developer</h2>
  <p>
    Just getting started with Polymer.<br>
    Feels like the future!
  </p>
</post-card>
```

In this step, you'll learn about:

- Creating a custom element using Polymer.
- Working with shadow DOM.

Learn more: Shadow DOM provides you a way to add a local DOM tree inside a DOM element, with local styles and markup that are decoupled from the rest of the web page.

To learn more about shadow DOM, see the Shadow DOM polyfill docs.

Edit post-card.html

Open `post-card.html` in your editor. This file contains the skeleton of a custom element, starting with some imports:

```
<link rel="import"
  href="../components/polymer/polymer.html">
<link rel="import"
  href="../components/core-icon-button/core-icon-button.html">
...
```

Key information

- As in the previous step, `<link rel="import">` is used to import elements the `post-card` element relies on.

Next is the definition of the element itself:

```
<polymer-element name="post-card">
  <template>
    <style>
    :host {
      display: block;
      position: relative;
      background-color: white;
      padding: 20px;
      width: 100%;
      font-size: 1.2rem;
      font-weight: 300;
    }
    .card-header {
      margin-bottom: 10px;
    }
    </style>

    <!-- CARD CONTENTS GO HERE -->
  </template>
  ...
```

Key information

- The `<polymer-element>` element is how you define a new custom element in Polymer. In this case, you're creating an element called "post-card".

- The `<template>` defines the element's internal DOM structure, or *shadow DOM*. This is where you'll add markup for your custom element.

- Used inside a shadow DOM tree, the `:host` pseudo-class matches the element that *hosts* the tree. In this case, it matches the `<post-card>` element.

- Ordinary selectors used inside the shadow DOM are *scoped* to the shadow DOM. The `.card-header` here only matches elements in this element's shadow DOM.

Note: The `<polymer-element>` tag can include only one `<template>` tag as a *direct* descendant. This tag defines the shadow DOM for the element. Other `<template>` tags may be nested inside the outer template tag.

At the end of the element definition is a `<script>` tag:

```
...
  <script>
  Polymer({});
  </script>
</polymer-element>
```

Key information

- The `Polymer` call at the end of the file *registers* the element so it's recognized by the browser. You'll do more with this in a later step as well.

Learn More: When you create an instance of `<post-card>`, the contents from its shadow DOM `<template>` are inserted as the element's *shadow root*. These elements are rendered in the browser, but are not included in the element's `children` collection.

By default, any children added by the user don't render. For example:

```
<post-card><h3>Hello!</h3></post-card>
```

Creates a `<post-card>` with a single `<h3>` element as a child. To render the `<h3>` inside your `<post-card>`, you need to add an *insertion point*, which tells the browser where to render children in the shadow DOM tree.

Create the card structure.

Find the `CARD CONTENTS GO HERE` comment and replace it with the `<div>` and `<content>` tags shown below.

```
</style>

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

```
</div>
<content></content>
```

Key information

- The `layout horizontal center` attributes are Polymer shorthand to create a flexbox layout.
- The three `<content>` elements create *insertion points*. (The shadow DOM spec calls this process of selecting nodes *distribution*).
- Any `<img>` children match the first `<content>` tag and are inserted here.
- The second `<content>` tag selects any `h2` children.
- The final `<content>` tag, with no `select` attribute, selects any nodes that haven't already been inserted. (This is probably the most common form of `<content>` element.)

Selecting content : The `select` attribute on a `content` element accepts a limited set of CSS selectors. You can only select direct children of the host node, not descendents.

Style the imported content.

There are a number of new CSS selectors to work with. The `post-card.html` file already includes a `:host` selector, discussed earlier, to style the top-level `<post-card>` element.

To style the children added using the `<content>` element, add the following CSS inside the `<style>` tag after the existing rules:

```
.card-header {
  margin-bottom: 10px;
}
polyfill-next-selector { content: '.card-header h2'; }
.card-header ::content h2 {
  margin: 0;
  font-size: 1.8rem;
  font-weight: 300;
}
polyfill-next-selector { content: '.card-header img'; }
.card-header ::content img {
  width: 70px;
  border-radius: 50%;
  margin: 10px;
}
</style>
```

Key information

- The `::content` pseudo element selects an insertion point (created by a `<content>` tag). Here, `::content h2` selects any `h2` that's distributed through an insertion point.

- For browsers that don't support shadow DOM natively the `polyfill-next-selector` rule tells the shadow DOM polyfill how to transform the `::content` rule into a non-shadow DOM rule. For example, without shadow DOM, `post-card h2` matches any `<h2>` element inside the card.

Note: You can't style the insertion point itself, so the `::content` pseudo element is always used with a descendent selector.

Edit index.html

Import the new element into `index.html`.

Save the `post-card.html` file and open `index.html` in your editor. Add the import for `post-card.html` after your existing imports:

```
...
<link rel="import"
  href="../components/paper-tabs/paper-tabs.html">
<link rel="import" href="post-card.html">
```

```
...
```

Key information

- This makes the `<post-card>` element available for use in `index.html`.

Add a `<post-card>` element to `index.html` directly after the `<core-toolbar>` element:

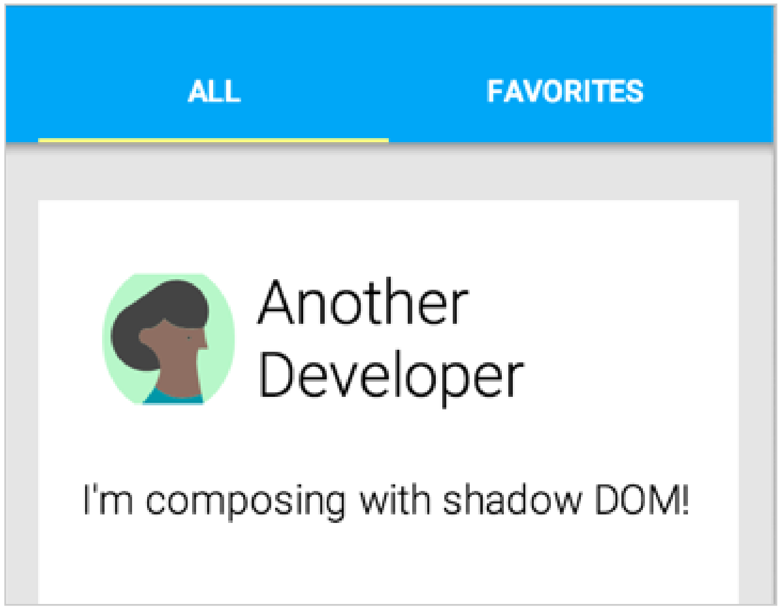
```
...
<div class="container" layout vertical center>

  <post-card>
    <img width="70" height="70"
      src="../images/avatar-07.svg">
    <h2>Another Developer</h2>
    <p>I'm composing with shadow DOM!</p>
  </post-card>

</div>
...
```

Key information

- The child elements you specify here are *distributed* into the `<post-card>` element's insertion points.

Test your work

Save your changes and reload the page. Your application should now look like this:



The card still needs a favorite button, but it's starting to take shape.

If something isn't working, check your work against the files in the `step-2` folder:

- `post-card.html`

- `index.html`

Explore: Play around with the insertion points to get a feeling for how they work. Does anything change if you reorder the `<post-card>`'s children in `index.html`? What if you include multiple images, or add plain text? You can also try swapping the two `select=` attributes in `post-card.html`.

← STEP 1: CREATING THE APP STRUCTURE → STEP 3: USING DATA BINDING

+POLYMER @POLYMER

/POLYMER FILE A BUG