DEV PREVIEW

Step 1: Creating the app structure

Your first Polymer application

Edit on GitHub

▶ **Table of contents**

Step 1: Creating the app structure

In this step, you'll use some pre-built Polymer elements to create the basic application structure, with a toolbar and tabs.

In this step, you'll learn about:

- Using HTML imports.
- Using Polymer elements with standard HTML, CSS and JavaScript.

Edit index.html

Go to the `starter` directory and open the `index.html` file in your favorite editor. The starting file looks like this:

```
<!doctype html>
<html>

<head>

  <title>unquote</title>

  <meta name="viewport"
    content="width=device-width, minimum-scale=1.0, initial-scale=

  <script src="../components/webcomponentsjs/webcomponents.js">
  </script>

  <link rel="import"
    href="../components/font-roboto/roboto.html">
  ...
```

Key information

- This bare-bones file defines some styles and embeds the `webcomponents.js` script, which supplies any missing platform features.

- The `link rel="import"` element is an *HTML import*, a new way of including resources into an HTML file.

> **Note:** The `font-roboto` import loads the `RobotoDraft` font using the Google Fonts API. If you're working offline or cannot access the Google Fonts API for any reason, this can block rendering of the web page. If you experience this problem, comment out the import for `font-roboto`.

Skipping over the styles for now, at the end of the file you'll find something new:

```
...
<body unresolved>

</body>
...
```

Key information

- The `unresolved` attribute on the `<body>` element is used to prevent a flash of unstyled content (FOUC) on browsers that lack native support for custom elements. For details, see the Polymer styling reference.

Add HTML import links to import the `<core-header-panel>`, `<core-toolbar>`, and `<paper-tabs>` elements:

```
<script
  src="../components/webcomponentsjs/webcomponents.js">
</script>

<link rel="import"
  href="../components/font-roboto/roboto.html">
<link rel="import"
  href="../components/core-header-panel/core-header-panel.html">
<link rel="import"
  href="../components/core-toolbar/core-toolbar.html">
<link rel="import"
  href="../components/paper-tabs/paper-tabs.html">
<style>
```

Key information

- Polymer uses HTML imports to load components. HTML imports provide dependency management, ensuring that your elements and all of their dependencies are loaded before you use them.
- Throughout this tutorial, the code you need to add appears in **`bold black text`**.

To add a toolbar, add the following code inside the `<body>` tag.

```
<core-header-panel>

  <core-toolbar>
  </core-toolbar>

  <!-- main page content will go here -->

</core-header-panel>
```

Key information

- The `<core-header-panel>` element is a simple container that holds a header (in this case a `<core-toolbar>` element), and some content. By default, the header stays at the top of the screen, but it can also be set to scroll with the content.
- The `<core-toolbar>` element serves as a container for tabs, menu buttons, and other controls.

Add the tabs.

The application will use tabs for navigating between two different views, a list of all messages and a list of favorites. The `<paper-tabs>` element works much like a `<select>` element, but it's styled as a set of tabs.

```
...
<core-toolbar>

  <paper-tabs id="tabs" selected="all" self-end>
    <paper-tab name="all">All</paper-tab>
    <paper-tab name="favorites">Favorites</paper-tab>
  </paper-tabs>

</core-toolbar>
...
```

Key information

- `<paper-tabs>` identifies the selected child by its name value or its index value.
- `selected="all"` chooses the first tab as the initially selected tab.
- In this case, the children are `<paper-tab>` elements, which provide styling and the "ink ripple" animation when you touch a tab.

- `self-end` is a layout attribute.

Add styles for the new elements. Add the following CSS rules inside the `<style>` element.

```
html,body {
  height: 100%;
  margin: 0;
  background-color: #E5E5E5;
  font-family: 'RobotoDraft', sans-serif;
}
core-header-panel {
  height: 100%;
  overflow: auto;
  -webkit-overflow-scrolling: touch;
}
core-toolbar {
  background: #03a9f4;
  color: white;
}
#tabs {
  width: 100%;
  margin: 0;
  -webkit-user-select: none;
  -moz-user-select: none;
```

```
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
  text-transform: uppercase;
}
```

Key information

- The `<core-header-panel>` is a generic element that can be used as either a full-page layout or for a card with a toolbar. To use it as a full-page, scrollable container, set its height explicitly.
- Here, the height is set to 100%. This works because the existing style rules ensure that its parent elements, `<html>` and `<body>`, take up 100% of the viewport height.
- The `overflow` and `-webkit-overflow-scrolling` properties ensure that scrolling works smoothly on touch devices, especially iOS.
- The `#tabs` selector selects the `<paper-tabs>` element. The toolbar adds a default margin on its children, to space controls appropriately. The tabs don't need this extra spacing.
- The `user-select` properties prevent the user from accidentally selecting the tab text.

Add a `<script>` tag near the end of the file to handle the tab switching event.

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

  tabs.addEventListener('core-select', function() {
    console.log("Selected: " + tabs.selected);
  });
</script>
</body>
```

Key information

- The `<paper-tabs>` element fires a `core-select` event when you select a tab. You can interact with the element just like a built-in element.
- Right now there's nothing to switch; you'll finish hooking it up later.

Save the file and open the project in your browser (for example, http://localhost:8000/starter/). You have a Polymer app!

Note: If you have the console open, you'll notice that you get two `core-select` events each time you switch tabs — one for the previously-selected tab and one for the newly-selected tab. The `<paper-tabs>` element inherits this behavior from `<core-selector>`, which supports both single and multiple selections.

If something isn't working, check your work against the `index.html` file in the `step-1` folder:

- `index.html`

In this step, you used HTML imports to import custom elements, and used them to create a simple app layout.

Explore: Can you use other children inside the `<paper-tabs>`? Try an image or a text span.

← GETTING STARTED

→ STEP 2: CREATING YOUR OWN ELEMENT

+POLYMER @POLYMER

/POLYMER FILE A BUG