# Step 3: Polymer custom element

In this step, you create a custom element that lets you use a `<hello-world>` tag anywhere in your app. You also specify element-defined style and expose an custom element attribute.

***Keywords**: custom element, ShadowDOM, custom attribute, binding*

## Bootstrap Polymer

Edit `web/index.html`, as follows.

```
<html>
  <head>
    <!-- ... -->

    <!-- Polymer bootstrap -->
    <script type="application/dart">export 'package:polymer/init.dart';</script>
  </head>
  <body>
```

Key information:

- The first `<script>` tag automatically initializes polymer elements without having to write a main for your application.
- When run in browser, the `index.html` file loads the app and calls the `main()` function in `package:polymer/init.dart` script.

## Add a basic custom element

Continue to edit `web/index.html`.

→ Declare a custom element named `hello-world`.

```
<html>
  <head>
    <!-- import polymer-element's definition -->
    <link rel="import" href="packages/polymer/polymer.html">
    <!-- ... -->
  </head>
  <body>
    <!-- Declare a custom element named hello-world -->
    <polymer-element name="hello-world" noscript>
      <template>
        <h3>Hello World!</h3>
      </template>
    </polymer-element>
```

→ Use it twice in the HTML code.

```
<body>
  <!-- ... -->

  <div>
    TO DO: Put the UI widgets here.
    <!-- Use the hello-world custom element -->
    <hello-world></hello-world>
    <hello-world></hello-world>
  </div>
</body>
```

→ Run in Dartium `web/index.html`.

You should see twice *"Hello World!"*.

**Got questions? Having trouble?** Ask the organizer team.

Key information:

- The `polymer-element` tag is the way to declare your custom element.
- The `name` attribute is required and **must** contain a `-`. It specifies the name of the HTML tag you'll instantiate in markup (in this case `<hello-world>`).
- The `noscript` attribute indicates that this is a simple element that doesn't include any script. An element declared with noscript is registered automatically.
- The `template` tag contains the content of the element. For now, it's static content.

## Add element-defined styles

To add element-defined styles, edit the element in `web/index.html`.

→ Add the following code to declare style:

```
<polymer-element name="hello-world" noscript>
  <template>
    <style>
      :host {
        text-align: center;
      }

      h3 {
        text-decoration: underline;
      }
    </style>
    <h3>Hello World!</h3>
  </template>
</polymer-element>
```

→ Run in Dartium

You should see centered and underlined *"Hello World!"*.

→ Try to affect the style of the element adding other styles outside of it.

```
<html>
  <head>
    <style>
      hello-world {
        text-align: right;
      }
      h3 {
        color: red;
      }
    </style>
  </head>
```

Key information:

- `:host` refers to the custom element itself and has the lowest specificity. This allows users to override your styling from the outside.
- Polymer creates *ShadowDOM* from the topmost of your definition, so styles defined internally to your element are scoped to your element. There's no need to worry about duplicating an id from the outside or using a styling rule that's too broad.

## Externalize element

Create a new file `web/hello.html`.

→ Move the element declaration from `web/index.html` to `web/hello.html`:

```
<!-- import polymer-element's definition -->
<link rel="import" href="packages/polymer/polymer.html">

<polymer-element name="hello-world" noscript>
  <template>
    <style>
      /* ... */
    </style>
    <h3>Hello World!</h3>
  </template>
</polymer-element>
```

→ Import the element with an HTML Import.

```
<html>
  <head>
    <!-- ... -->

    <link rel="import" href="hello.html">

    <script type="application/dart">export 'package:polymer/init.dart';</script>
  </head>
  <body>
    <!-- Previous hello-world element declaration is removed -->

    <!-- ... -->

    <div>
      TO DO: Put the UI widgets here.
      <!-- Use the hello-world custom element -->
      <hello-world></hello-world>
      <hello-world></hello-world>
    </div>
  </body>
</html>
```

Key information:

- HTML Imports are a way to include and reuse HTML documents in other HTML documents.
- For HTML imports use the `import` relation on a standard `<link>` tag.

## Add custom behavior

Create a new file `web/hello.dart`.

→ Add the following code:

```
import 'package:polymer/polymer.dart';

@CustomTag('hello-world')
class HelloWorld extends PolymerElement {
  String name = "You";

  HelloWorld.created(): super.created();
}
```

→ In `web/hello.html`, remove the `noscript` attribute.
→ Add the `script` tag specifying the URL to `hello.dart`.
→ Add the binding to the field `name` of `HelloWorld` class:

```
<link rel="import" href="packages/polymer/polymer.html">

<polymer-element name="hello-world">
  <template>
    ...
    <h3>Hello {{name}}!</h3>
  </template>
  <script type="application/dart" src="hello.dart"></script>
</polymer-element>
```

→ Run in Dartium

You should see *"Hello You!"*.

Key information:

- The `script` tag specifies the path to the underlying dart script.
- Any Polymer element class extends `PolymerElement`.
- `CustomTag` specifies the name of the element.
- The `super.created()` constructor must be called in the custom element constructor.
- `{{name}}` is a Polymer expression. It is bound to the `name` field in `HelloWorld` class.

## Add custom attribute

Attributes are a great way for users of your element to configure it, declaratively.

→ In `web/hello.dart`, add the `@published` annotation to the `name` field:

```
class HelloWorld extends PolymerElement {
  @published
  String name = "You";
  // ...
}
```

→ In `web/index.html`, take advantage of the new attribute:

```
<html>
  <body>
    <!-- ... -->

    <div>
      TO DO: Put the UI widgets here.
      <hello-world name="John"></hello-world>
      <hello-world name="Paul"></hello-world>
    </div>
  </body>
</html>
```

→ Run in Dartium

You should see *"Hello John!"* and *"Hello Paul!"*.

Key information:

- `@published` means that is is an public attribute.
- `@published` also means that it is observable, so it allows to uses live two-way data binding to synchronize DOM nodes and object models.

## Learn more

- Polymer.dart - Creating Elements
- Polymer project
- A Guide to Styling Elements
- Custom elements
- Shadow DOM
- HTML Templates
- HTML Imports

## Problems?

Check your code against the files in s3_element (diff).

**Home | < Previous | Next >**

© 2015 GitHub, Inc. Terms Privacy Security Contact Status API Training Shop Blog About