Polymer

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

Polymer.js is a JavaScript library created by Google that allows reusing the HTML elements for building applications with components.

This tutorial covers most of the topics required for a basic understanding of Polymer.js and to get a feel of how it works.

Audience

This tutorial is designed for software programmers who want to learn the basics of Polymer.js and its programming concepts in a simple and easy way. This tutorial will give you enough understanding on the components of Polymer.js with suitable examples.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of HTML, CSS, JavaScript, Document Object Model (DOM) and any text editor. As we are going to develop web-based application using Polymer.js, it will be good the readers have understanding on how internet and web-based applications work.

Copyright & Disclaimer

Table of Contents

1. POLYMER – OVERVIEW

Polymer.js is a JavaScript library created by Google that allows reusing the HTML elements for building applications with components.

Polymer is an open-source JavaScript library developed by Google developers and was initially released on May 27, 2015. The stable release is 1.7.0 and it was released on September 29, 2016.

Why Use Polymer.js?

- It allows to create our own custom elements easily using the HTML, CSS, and JavaScript for adding interactions to the element.
- It is created by Google that provides cross-browser compatible applications along with the web components.
- It provides both one-way and two-way data binding.
- It provides Polymer command line interface for managing the projects from simple components to complicated web applications.

Features of Polymer.js

- It is a JavaScript library built on top of the web standards APIs that allow building custom HTML elements.
- It provides the polyfills (web component specifications) for creating our own customized and reusable elements.
- It uses the web component standards for the creation of reusable widgets in web documents and web applications.
- It uses Google material design for the development of hybrid mobile application.
- It distributes the custom elements across the network and the users can use these elements with the help of HTML Imports.

2. POLYMER – INSTALLATION

It's easy to configure Polymer in your system. Following are the two ways to install Polymer.

- The Polymer CLI (Command Line Interface)
- The Bower

Installing Polymer Using Polymer CLI

Step 1: Install Polymer using the following npm command.

```
npm install -g polymer-cli@next
```

Step 2: Check the successful installation and version using the following command.

```
polymer --version
```

If it has installed successfully, then it will show the version as:

```
0.18.0-pre.13.
```

Step 3: Create a directory with the name of your choice and switch to that directory.

```
mkdir polymer-js
cd polymer-js
```

Step 4: To initialize your project, run the following command in your *polymer-js*directory:

```
polymer init
```

After executing the above command, it will show something like this:

```
C:\polymer-js>polymer init
? Which starter template would you like to use?
  1) polymer-1-element - A simple Polymer 1.0 element template
  2) polymer-2-element - A simple Polymer 2.0 element template
  3) polymer-1-application - A simple Polymer 1.0 application template
  4) polymer-2-application - A simple Polymer 2.0 application
  5) polymer-1-starter-kit - A Polymer 1.x starter application template, with
navigation and "PRPL pattern" loading
```

```
  6) polymer-2-starter-kit - A Polymer 2.x starter application template, with
navigation and "PRPL pattern" loading

  7) shop - The "Shop" Progressive Web App demo

  Answer: 4
```

Step 5: Select the polymer-2-application from the above given options. Now, start your project using the following command.

```
 polymer serve
```

Installing Polymer Using Bower

Step 1: To start from scratch using Bower method, install the Bower using the following command.

```
npm install -g bower
```

Step 2: Install the Polymer using the following command.

```
npm install -g polymer-cli@next
```

Step 3: Check the successful installation and version of Polymer, using the following command.

```
polymer --version
```

If it has installed successfully, then it will show the version as:

```
0.18.0-pre.13.
```

Step 4: To install the latest Polymer 2.0 RC release from bower, use the following command.

```
bower install Polymer/polymer#^2.0.0-rc.3
```

Step 5: Create a **index.html** file and add the following code in the <head> tag.

```
<script src="/bower_components/webcomponentsjs/webcomponents-loader.js"></script>
// it loads the polyfills

<link rel="import" href="/bower_components/polymer/polymer.html"> // it import
Polymer
```

Step 6: Start your project using the following command.

```
polymer serve
```

Building for Deployment

To build your project for deployment, **polymer build** command is an easier way, which will minify, compile, or bundle your code depending on the command line flags.

To create a universal build that works on all browsers, use the following command.

```
polymer build --js-compile
```

The above command will build the project to build/default and you can start this directory, using the following command.

```
polymer serve build/default
```

Polymer 2.0 uses ES6 and HTML Custom Elements. For best practice, it is always good to use ES6 to browsers with full ES6 support and compile ES5 to old browsers that don't support ES6. The following table shows the best strategy for your project.

Strategy	Easiest for cross-browser support	Most optimal for WC v1 performance
Server	Any server works, including static ones	Differential serving required
Deployed Code	ES5 transpiled	ES6
Polyfill Loader	webcomponents-es5-loader.js	webcomponents-loader.js

3. POLYMER – ELEMENTS

Polymer elements are a set of visual and non-visual elements designed to work with the layout, user interaction, selection, and scaffolding applications. These include everything from a simple button to a dialog box with neat visual effects. The following table shows different types of polymer elements.

Sr. No.	Types & Description
1	**app elements** The app elements are useful when building entire applications.
2	**iron elements** These are the basic building blocks for creating an application.
3	**paper elements** The paper elements are a set of UI components designed to implement Google's material design guidelines.
4	**google Web components** The google Web component are a stock of web components for Google APIs & services.
5	**gold elements** The gold elements are built for e-commerce-specific use cases.
6	**neon elements** It is used for implementing animated transitions for Polymer elements using web animations.
7	**platinum elements** The platinum elements provide features to turn your web page into a true webapp.
8	**molecules elements** The molecule element helps to develop an application easily and is used to connect a group of plugins to the Polymer application.

App Elements

The app elements are useful while building an entire application. The following table lists different types of polymer elements. These components support features such as routing, data storage, and more.

Sr. No.	Types & Description
1	**app-layout** The app-layout elements are useful for building high-quality, responsive layouts just with markup.
2	**app-route** These are for managing the routing within an app, mapping URLs to views.

Polymer - App Layout

The app-layout elements are comprised of components such as toolbars, drawers, and headers. These are used for building high-quality, responsive layouts just with markup. Some of the elements are listed in the following table.

Sr. No.	Elements & Description
1	**app-box** This element works as container and has scroll effects, visual effects based on the scroll position.
2	**app-drawer** This is a navigation drawer which will slide in and out from left or right.
3	**app-drawer-layout** This will position the app-drawer and other content.
4	**app-grid** This is used for creating responsive and fluid grid layouts using custom properties.
5	**app-header** This element works at the top of the screen as a container for app-toolbars and has scroll effects, visual effects based on the scroll position.

6	**app-header-layout** This element acts as a cover that positions the app-header and other content.
7	**app-scrollpos-control** This element is used for saving and restoring the scroll position when multiple pages share the same document scroller.
8	**app-toolbar** It is a horizontal toolbar which contains items that can be used for labeling, navigating, searching, and other actions.

Example

To use app-layout elements you have to move into the project directory using the following command in the command prompt.

```
bower install PolymerElements/app-layout --save
```

The above command will install the app-layout elements in **bower_components** folder. Then, you have to import the file using <link> tag in your **index.htmlfile**.

```
<link rel="import" href="/bower_components/app-layout/app-layout.html">
<base href="https://user-content-dot-custom-
elements.appspot.com/PolymerElements/app-layout/v1.0.1/app-layout/">
<script src="../webcomponentsjs/webcomponents-lite.min.js"></script>
<link rel="import" href="app-header/app-header.html">
<link rel="import" href="app-toolbar/app-toolbar.html">
<link rel="import" href="app-box/app-box.html">
<link rel="import" href="demo/sample-content.html">
<link rel="import" href="../iron-icons/iron-icons.html">
 <style is="custom-style">
  html, body {
    margin: 0;
    font-family: 'Roboto', 'Noto', sans-serif;
    -webkit-font-smoothing: antialiased;
    background: #f1f1f1;
    max-height: 368px;
  }
  app-toolbar {
    background-color: #4285f4;
```

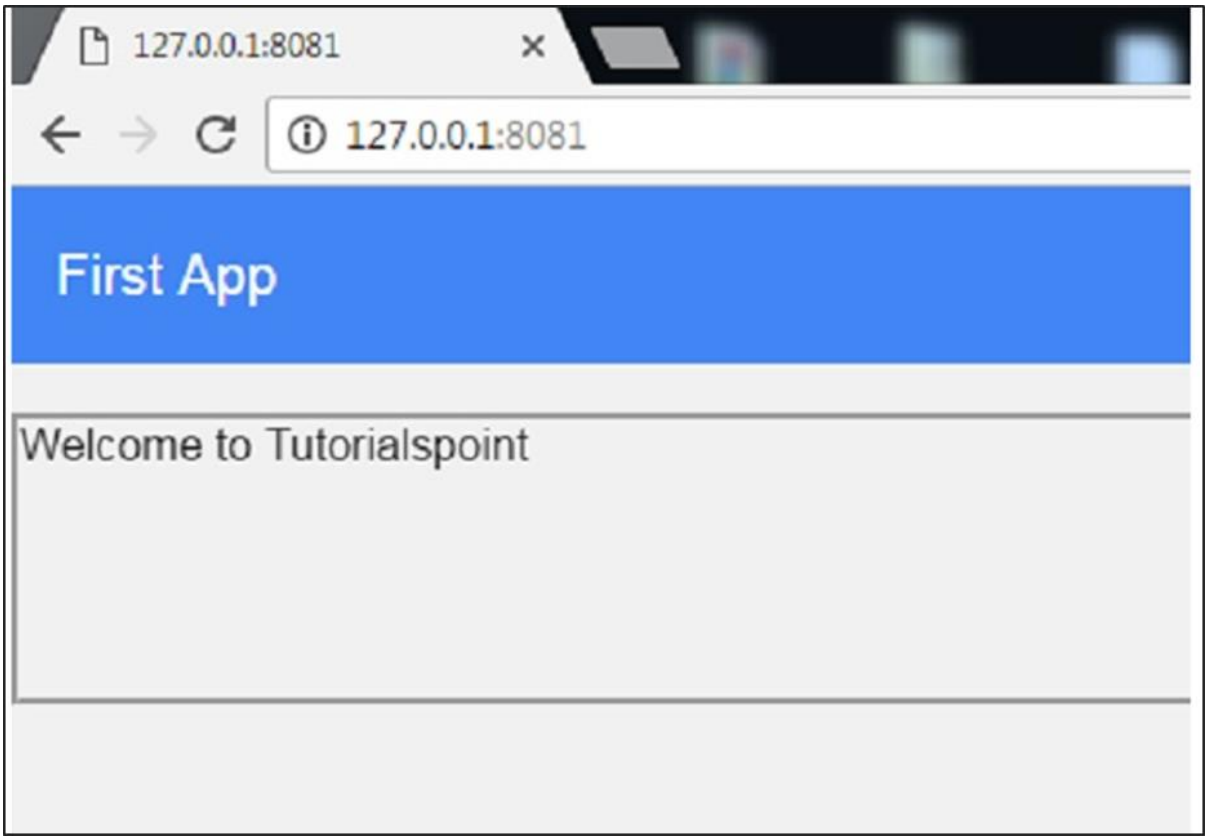
```
    color: #fff;
  }

  paper-icon-button + [main-title] {
    margin-left: 24px;
  }
  paper-progress {
    display: block;
    width: 100%;
    --paper-progress-active-color: rgba(255, 255, 255, 0.5);
    --paper-progress-container-color: transparent;
  }
  app-header {
    @apply(--layout-fixed-top);
    color: #fff;
    --app-header-background-rear-layer: {
      background-color: green;
    };
  }

  sample-content {
    padding-top: 64px;
  }
 </style>
<app-header reveals>
  <app-toolbar>
   <div main-title>First App</div>
 </app-toolbar>
</app-header>
<sample-content></sample-content><br>
<app-box style="height: 100px;border-style: groove;">
 <div main-title>Welcome to Tutorialspoint</div>
</app-box>
```

Output

Refresh the Polymer server and following will be the output.



Polymer - App Route

The app-route element utilizes an object, which describes a state about the current route, via the route property. It will determine the state using pattern property and determines some data related to the route, and a tail that contains the rest of the route as shown in the following code.

```
<app-location route="{{route}}"></app-location>
<app-route
    route="{{route}}"
    pattern="/:page"
    data="{{routeData}}"
    tail="{{subroute}}">
```

```
</app-route>
```

```
<app-route
    route="{{subroute}}"
    pattern="/:id"
    data="{{subrouteData}}">
</app-route>
```

The explanation of the above instances is listed in the following table.

Sr. No.	Field & Description
1	**app-location** It is an element that provides synchronization between the browser location bar and the state of an app, and produces a route value.
2	**Pattern** The **route.path** property is matched by comparing it to the **pattern** property.
3	**app-route** It sets the data property with an object, whose properties correspond to the parameters in **pattern** property. It is responsive to bi-directional changes to the data objects they produce.
4	**Tail** It represents the remaining part of the route state, after the pattern has been applied to a matching route.

Hashes vs Paths

The portion of the URL pathname is used by an app-location route with the help of backend server. The app-location can be configured to use the hash part using the following attribute.

```
<app-location route="{{route}}" use-hash-as-path></app-location>
```

Iron Elements

The iron elements are used to create an application and are also called as **core** elements, when matched with the "Developer Preview" version.

The following table lists different types of iron elements:

Sr. No.	Type & Description
1	**iron-a11y-keys** These elements are used to process keyboard commands using cross-browser interface.
2	**iron-ajax** The iron-ajax elements are useful in making the ajax calls.
3	**iron-collapse** The iron-collapse elements are used to collapse a content. To show or hide the content use **opened** or **toggle()**.
4	**iron-image** These elements are used for displaying a single image.
5	**iron-dropdown** These are low-level elements used to reveal the hidden dropdown content.
6	**iron-flex-layout** These elements are useful to provide CSS flexible box layout.
7	**iron-form** The iron-form element is an HTML element used to validate and submit any custom elements.
8	**iron-icon** These elements are used to display a single icon.
9	**iron-swipeable-container** It is a container that allows swapping of its nested children, i.e. native or custom elements.

Iron A11y Keys

The **<iron-a11y-keys>** element is used to process keyboard commands using cross-browser interface.

The **keys** attribute indicates by what combination of keys the event will be triggered. It accepts modifier keys such as Shift, Control, Alt, Meta and some common keyboard keys such

as a-z, 0-9, F1-F12, Page Up, Page Down, Left Arrow, Right Arrow, Home, End, Escape, Space, Tab, and Enter.

All the keys should be shortened and should be in lowercase. For example, Right Arrow is for right, Page Up is for pageup, Control is for ctrl, Escape is for esc, F5 is for f5, etc.

Example

To use the iron-a11y-keys element, navigate to your project folder in the command prompt and use the following command.

```
bower install PolymerElements/iron-a11y-keys --save
```

The above command installs the iron-a11y-keys element in **bower_components** folder. Next, import the iron-a11y-keys file in your index.html using the following command.

```
<link rel="import" href="/bower_components/iron-a11y-keys/iron-a11y-keys.html">
```

The following example demonstrates the use of iron-a11y-keys element.

```
<!DOCTYPE html>
<html>
  <head>
    <base href="http://polygit.org/components/">
    <meta name="viewport" content="width=device-width">
    <title>iron-a11y-keys</title>
    <link rel="import" href="polymer/polymer.html">
    <link rel="import" href="iron-a11y-keys/iron-a11y-keys.html">
    <link rel="import" href="paper-input/paper-input.html">
  </head>

  <body>
    <demo-keys></demo-keys>
    <dom-module id="demo-keys">

    <template>
      <iron-a11y-keys target="[[_target]]" keys="shift+enter enter esc pageup pagedown
up down left right space" on-keys-pressed="_onKeyPress"></iron-a11y-keys>

      <h4>Press any of the below keys and check console.</h4>
      <p>shift + enter, enter, esc, pageup, pagedown, up, down, left, right, space.<p>
```

```
      <paper-input type="text" value="{{_value::input}}" id="input">
    </template>

    <script>
        Polymer({
          is: 'demo-keys',
          properties: {
            _value: {
              type: String
            },
            _target: {
              value: function() {
                return this.$.input;
              }
            }
          },
          _onKeyPress: function(e) {
            e.detail.keyboardEvent.preventDefault();
            console.log(e.detail.combo);
          }         });
    </script>
    </dom-module>
  </body>
</html>
```

Output

To run the application, navigate to your project directory and run the following command.

```
polymer serve
```

Now, open the browser and navigate to http://127.0.0.1:8081/. Following will be the output.

Iron Ajax

The <iron-ajax> element is useful in making ajax calls.

```
<iron-ajax
    auto
    url="https://www.googleapis.com/youtube/v3/search"
    params='{"part":"snippet", "q":"polymer", "key": "YOUTUBE_API_KEY", "type":
"video"}'
    handle-as="json"
    on-response="handleResponse"
    debounce-duration="500">
</iron-ajax>
```

When you set **auto** to *true*, a request is made by an element when its properties **url**, **params**, or **body** are changed. As you can see, the component has several attributes available:

- **url**: An attribute where you place the url to the API endpoint.
- **params**: An attribute where you can pass the JSON with the request parameters.
- **handle-as**: Specifies what data must be stored in the response property. By default, the data is stored in the json format.

- **on-response**: An attribute which can tell the iron-ajax component by what method the response is handled.

Changing multiple attributes sequentially causes automatically generated requests to be debounced.

You can call **generateRequest** on the element to trigger a request explicitly.

Example

In the following example, we have a link to the **iron-ajax** and **paper-input** components used from the CDN.

```
<!DOCTYPE html>
<html>
  <head>
    <title>iron-ajax</title>
    <link rel="import" href="https://cdn.rawgit.com/download/polymer-
cdn/1.0.1/lib/paper-input/paper-input.html">
    <link rel="import" href="https://cdn.rawgit.com/download/polymer-
cdn/1.0.1/lib/iron-ajax/iron-ajax.html">
  </head>

  <body>
    <dom-module id="data-bind">

    <template>
      <paper-input label="Write something..." id="input" value="{{q::input}}"
autofocus>
      </paper-input>
      <p>You typed: <b>{{q}}</b></p>
      <p>Echo: <b>{{echo}}</b></p>

      <iron-ajax auto url="http://dict.leo.org/dictQuery/m-
query/conf/ende/query.conf/strlist.json" params="{'q': q}" handle-as="json" on-
response="_handleResponse" debounce-duration="500"></iron-ajax>
    </template>

    <script>
      (function () {
        Polymer({
```

```
          is: 'data-bind',

          properties: {
            echo: {
              type: String,
              value: 'waiting ...',
              reflectToAttribute: true
            }
          },

          _handleResponse: function(event, request) {
             var response = request.response;
             this.echo = JSON.stringify(response);
          }
        });
      })();
    </script>
    </dom-module>

    <data-bind></data-bind>
  </body>
</html>
```

Output

To run the application, navigate to your project directory and run the following command:

```
polymer serve
```

Now open the browser and navigate to http://127.0.0.1:8081/. Following will be the output.

Iron Collapse

The <iron-collapse> elements are used to collapse a content. To show or hide the content, use opened or toggle().

The max-height/max-width of the collapsible element is automatically adjusted by the iron-collapse element.

The custom properties and mixins to be used for styling is as follows:

- **--iron-collapse-transition-duration**: It is the duration of the animation transition. The default value is 300ms.

Example

To implement iron-collapse element, navigate to your project folder in the command prompt and use the following commands:

```
bower install PolymerElements/iron-collapse --save
bower install PolymerElements/paper-toggle-button --save
```

The above command installs both the elements in **bower_components** folder. Then you have to import both the files in your index.html file as shown below:

```
<link rel="import" href="iron-collapse/iron-collapse.html">
<link rel="import" href="paper-toggle-button/paper-toggle-button.html">
```

The following example demonstrates the use of iron-collapse element:

```
<!DOCTYPE html>
<html>
  <head>
```

```
    <title>iron-collapse</title>
    <base href="http://polygit.org/polymer+:master/components/">
    <link rel="import" href="polymer/polymer.html">
    <link rel="import" href="paper-toggle-button/paper-toggle-button.html">
    <link rel="import" href="iron-collapse/iron-collapse.html">

    <style>
      #coll {
        display: flex;
        width: 500px;
      }
      demo-collapse{
        border: 2px solid LightGrey;
        width: 50%;
      }
    </style>
  </head>

  <body>
    <h3>Iron Collapse Example</h3>
    <dom-module id="demo-collapse">
    <template>
      <paper-toggle-button checked="{{opened}}">Collapse this</paper-toggle-button>
      <iron-collapse opened="[[opened]]">
        <div><p>This is polymerjs iron-collapse.</p></div>
      </iron-collapse>
    </template>
    </dom-module>

    <script>
      Polymer({
        is: 'demo-collapse',
        });
    </script>
```

```
    <div id="coll">
      <demo-collapse></demo-collapse>
    </div>
  </body>
</html>
```

Output

To run the application, navigate to your project directory and run with the following command:

```
polymer serve
```

Now open the browser and navigate to http://127.0.0.1:8081/. Following will be the output.

When you click the toggle button, following will be the output.

Iron Image

The <iron-image> is an image-displaying element. The image is displayed with the useful sizing and preload options.

The sizing option either **crops** (cover) or **letterboxes** (contain) the image to fit within its specified size whereas the preload option blocks the image from getting rendered. Meanwhile, you can either use the element's CSS background-color as the placeholder or you can prefer a data-URI. The **fade** option fades out the image/color of the placeholder after rendering the image.

The <iron-image> element is similar to <img scr="...."> tag as shown in the following code snippet:

```
<iron-image src="http://lorempixel.com/600/600"></iron-image>
```

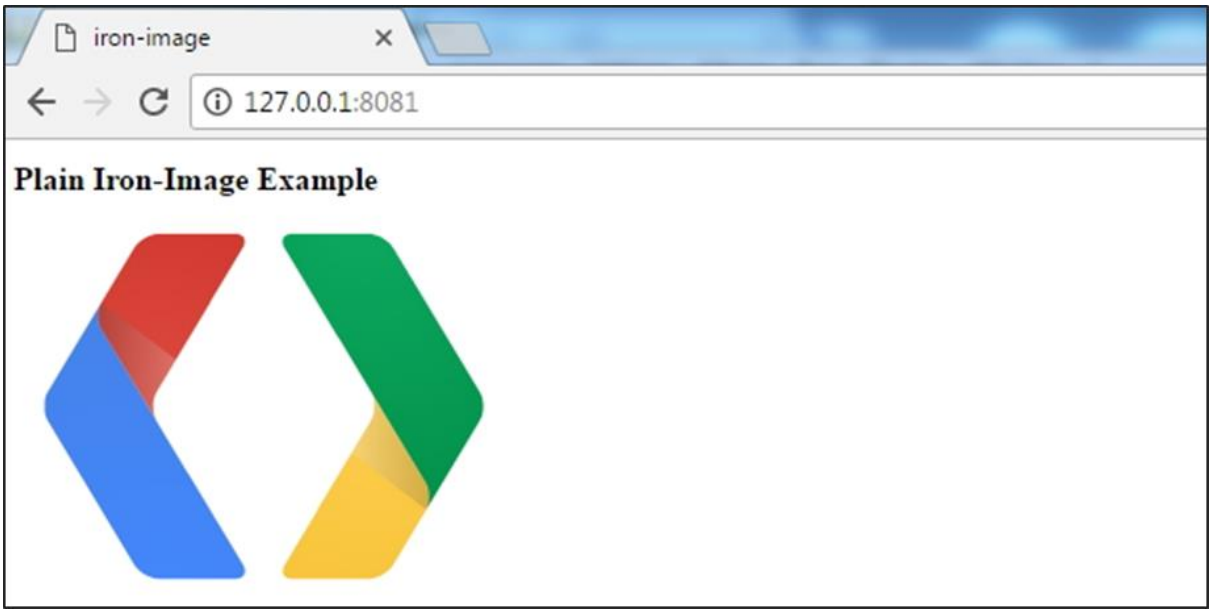
Example

To use the iron-image element, install the iron-image element using bower and import it in your index.html file. The following code displays a plain image.

```
<!DOCTYPE html>
<html>
  <head>
    <title>iron-image</title>
    <base href="https://polygit.org/components/">
    <script src="webcomponentsjs/webcomponents-lite.js"></script>
    <link rel="import" href="iron-image/iron-image.html">
  </head>
```

```
  <body>
    <h1>Iron-Image Example</h1>
    <iron-image src= "http://www.tcgreenmedia.com/wp-
content/uploads/2014/07/google-developers-v01-510x380.png" alt="iron-image"
></iron-image>
  </body>
</html>
```

You will get the output as shown in the following screenshot.



Use the option sizing = "cover"

```
<!DOCTYPE html>
<html>
  <head>
    <title>iron-image</title>
    <base href="https://polygit.org/components/">
    <script src="webcomponentsjs/webcomponents-lite.js"></script>
    <link rel="import" href="iron-image/iron-image.html">

    <style>
      #sizing-cover {
```

```
      width: 140px;
      height: 140px;
      background: LightGrey;
      margin-left: 20px;
    }
    </style>
  </head>

  <body>
    <h1>Example using sizing = "cover"</h1>
    <iron-image src= "http://www.tcgreenmedia.com/wp-
content/uploads/2014/07/google-developers-v01-510x380.png" sizing="cover"
id="sizing-cover" alt="iron-image" ></iron-image>
  </body>
</html>
```

You will get the output as shown in the following screenshot.

Use the option sizing = "contain"

```
<!DOCTYPE html>
<html>
  <head>
    <title>iron-image</title>
    <base href="https://polygit.org/components/">
    <script src="webcomponentsjs/webcomponents-lite.js"></script>
    <link rel="import" href="iron-image/iron-image.html">
```

```
    <style>
      #sizing-contain {
      width: 140px;
      height: 140px;
      background: #ddd;
       margin-left: 20px;
    }
    </style>
  </head>
    <body>
    <h1>Example using sizing = "contain"</h1>
    <iron-image src= "http://www.tcgreenmedia.com/wp-
content/uploads/2014/07/google-developers-v01-510x380.png" sizing="contain"
id="sizing-contain" alt="iron-image" ></iron-image>
  </body>
</html>
```

You will get the output as shown in the following screenshot.

The following code displays the grey background as well as base-64 encoded placeholder until the image is loaded.

```
<iron-image style="width:800px; height:600px; background-color: grey;"
placeholder="..."
  sizing="cover" preload src="http://lorempixel.com/600/600"></iron-image>
```

The following code fades out the image after the image is rendered.

```
<iron-image style="width:800px; height:600px; background-color: grey;"
  sizing="cover" preload fade src="http://lorempixel.com/600/600"></iron-image>
```

The following table shows the custom properties of <iron-image> element.

Sr. No.	Custom properties & Description	Default
1	**--iron-image-placeholder**: It is a mixin to be used to style for #placeholder.	{}
2	**--iron-image-width**: Use this property to set the width of the image, wrapped by the iron-image.	auto
3	**--iron-image-height**: Use this property to set the height of the image, wrapped by the iron-image.	auto

Iron Dropdown

The <iron-dropdown> element is used to reveal the hidden dropdown content - **.dropdown-content**. The implementation of elements that uses iron-dropdown may include comboboxes, menubuttons, selects, etc.

The <iron-dropdown> element displays the attributes where the **.dropdown-trigger** is configured relative to the position of .dropdown-content.

Example

To implement iron-dropdown element, navigate to your project folder in the command prompt and use the following command.

```
bower install PolymerElements/iron-dropdown --save
```

The following example demonstrates the use of iron-dropdown element:

```
<!DOCTYPE html>
<html>
  <head>
    <title>iron-dropdown</title>
    <base href="http://polygit.org/components/">
```

```
    <script src="webcomponentsjs/webcomponents-lite.min.js"></script>
    <link rel="import" href="iron-dropdown/demo/x-select.html">
    <link rel="import" href="paper-input/paper-input.html">
    <link rel="import" href="paper-button/paper-button.html">

    <style>
      .dropdown-trigger{
            background-color: DarkCyan ;
            border-radius: 4px;
            color: white;
      }
      .dropdown-content {
            background-color: white;
            line-height: 15px;
            border-radius: 5px;
            box-shadow: 0px 2px 5px #ccc;
            padding: 20px;
      }
    </style>
  </head>

  <body>
    <h3>Iron-dropdown Example</h3>
    <x-select>
      <paper-button class="dropdown-trigger">Click Me !</paper-button>
      <div class="dropdown-content">
        <p>Hello !!! <br/>
        <p>This is an iron-dropdown <br/> example of Polymerjs.</p>
      </div>
    </x-select>
  </body>
</html>
```

As shown in the example, the class named .dropdown-content will be hidden till you call an open method on an element.

Output

To run the application, navigate to your project directory and run the following command:

```
polymer serve
```

Now open the browser and navigate to http://127.0.0.1:8081/. When you click the **Click me** button, a dropdown will be displayed.

Iron Flex Layout

The <iron-flex-layout> element is useful for providing CSS flexible box layout and is also called as **flexbox**.

The flexbox can be used in two different ways:

- **Layout classes:** It is a stylesheet providing certain rules of class-based flexbox where the layout properties are directly defined in the markup.
- **Custom CSS mixins:** It is a stylesheet containing custom CSS mixins, which we can apply inside a CSS rule with the help of @apply function.

Example

To use the iron-flex-layout element, navigate to your project folder in the command prompt and use the following command:

```
bower install PolymerElements/iron-flex-layout --save
```

The above command installs the iron-flex-layout element in bower_components folder. Next, import the iron-flex-layout file in your index.html using the following command:

```
<link rel="import" href="/bower_components/iron-flex-layout/iron-flex-layout.html">
```

The following example demonstrates the use of iron-flex-layout element:

```
<!DOCTYPE html>
<html>
  <head>
    <title>iron-flex-layout</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="bower_components/webcomponentsjs/webcomponents-lite.min.js"></script>
    <link rel="import" href="bower_components/iron-flex-layout/classes/iron-flex-
layout.html">

    <style>
      body {
            font-weight: 300;
      }
      div {
            border: 2px solid DarkCyan ;
            background-color: white;
      }
      .layout {
            margin-bottom: 25px;
            background-color: DarkCyan ;
      }
      p {
            margin: 10px;
      }
      </style>

  </head>
  <body>
    <h2>Iron-Flex-Layout</h2>
      <div class="horizontal layout">
        <div><p>Delta</p></div>
        <div><p>Epsilon (flex)</p></div>
        <div><p>Zeta</p></div>
      </div>
  </body>
```

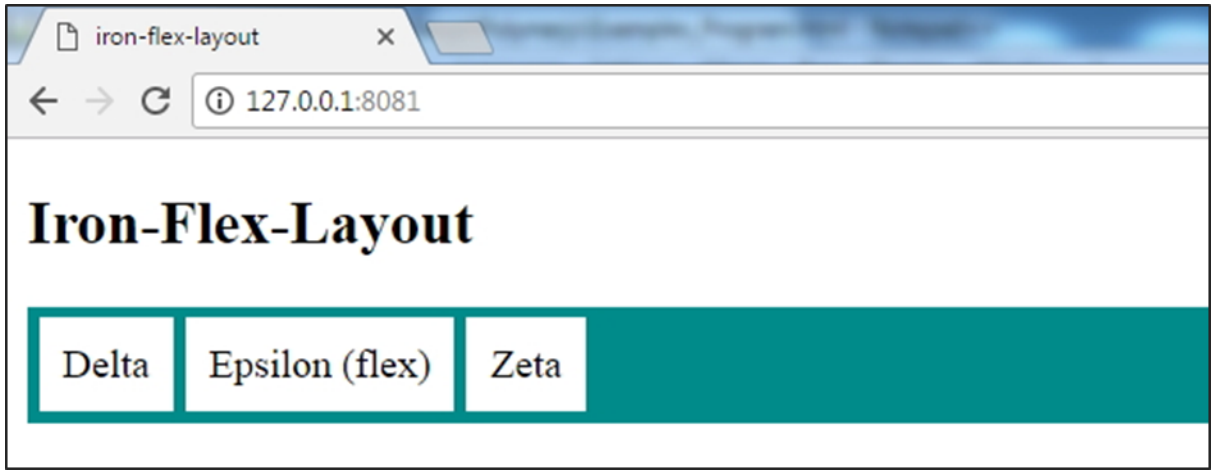
```
</html>
```

Output

To run the application, navigate to your project directory and run the following command:

```
polymer serve
```

Now open the browser and navigate to http://127.0.0.1:8081/. Following will be the output.



Iron Form

The <iron-form> is an HTML <form> element used to validate and submit any custom elements and native elements. Both the **get** and **post** methods are supported and iron-ajax element is used to submit the data to the action URL.

By default, the native button element submits the following form:

```
<form is="iron-form" id="form" method="post" action="/form/handler">
  <paper-input name="password" label="Password"></paper-input>
  <input name="address">
  ...
</form>
```