Polymer workshop assignments

# What we’ll build

We're going to develop a Polymer app that does the following:

* Add new contacts to the list
* Create contact list and save it in localStorage
* Show contact list using Polymer Material design components
* Delete contact list from the app and from localStorage
* Update contact list by editing the selected contact

# Setup and Installation

### Pre-requisite:

1. Download and install editor: Visual studio code

<https://code.visualstudio.com/download>

1. Install node js (npm is already integrated in the same package) <https://nodejs.org/en/download/>
2. Install git: <https://git-scm.com/downloads>
3. Install bower from terminal (cmd):

npm install -g bower

1. Install polymer cli from terminal (cmd):

npm install -g polymer-cli

### Download polymer app from github:

1. Navigate to your working folder
2. Clone app by using this command line in terminal:

git clone <https://github.com/rmacho/polymer-workshop.git>

1. Jump into the app folder and run bower install:

cd polymer-workshop

bower install

1. Run the application:

polymer serve -o

Your application is running and ready to use!

# Assignments

assignment 1 (run application and fire event to add new contact):

1. Run application
2. Go to your browser and open the app page
3. Fill in the contact name and click on ‘ ADD CONTACT’.

Nothing is happening after hitting the button! To make this work you need to send an event and inform the parent component that a new contact has been added

1. Open **‘/src/components/add-contact.html’** file
2. In **addContact()** function, add this code line after **‘//Fire event when contact is added’:**

this.dispatchEvent(new CustomEvent('newContactAdded',

{bubble: true, composed: true}));

1. Fill the contact name again and try to add the contact into your list

Now it’s working! You are now able to add new contacts.

For the next step you will add a new input field to add a phone number and save it

1. Open **‘/src/components/add-contact.html’** file
2. Add this piece of code under the first **<paper-input>** tag

<paper-input

id="contactPhone"

label="[[\_labelPhone]]"

auto-validate

pattern="^[0-9]\*$"

error-message="Numbers only!">

</paper-input>

1. In **addContact()** function, replace the if-statement with this one:

if(this.$.contactName.value && this.$.contactPhone.value) {

contactList.push({

name: this.$.contactName.value,

phone: this.$.contactPhone.value

});

1. Go to your browser and refresh the app page. Now you see the new field for phone number. Fill in a new contact with a phone number and click on ‘ADD CONTACT’ after that.

A new contact has been added with phone number. Good job!

#### Assignment 2 (create custom component for contact list):

We already created an empty component for contact list. You can find it in ‘**src/components/show-contact-list.htm**l’

1. Open ‘**src/polymer-workshop-app.html’**
2. Move the following imports to **show-contact-list.html**

<link rel="import" href="../../../bower\_components/iron-list/iron-list.html">

<link rel="import" href="../../../bower\_components/paper-item/paper-item.html">

<link rel="import" href="../../../bower\_components/paper-item/paper-item-body.html">

<link rel="import" href="../../../bower\_components/iron-icon/iron-icon.html">

<link rel="import" href="../../../bower\_components/iron-icons/iron-icons.html">

1. Go back to **polymer-workshop-app.html** and move the following html code to **show-contact-list.html**

<iron-list items="[[contactListItems]]" as="listItem">

<template>

<paper-item>

<paper-item-body two-line>

<div>[[listItem.name]]</div>

<div secondary><iron-icon icon="settings-phone"></iron-icon> [[listItem.phone]]</div>

</paper-item-body>

</paper-item>

</template>

</iron-list>

1. Create a ‘**list**’ property for your web component: list is an Array
2. Replace the **contactListItems** with the list property in your html
3. Import **show-contact-list.html** component in your polymer-workshop-app.html
4. Call and use **<show-contact-list>** in the same file. Put the element in the contact list “dom-if” statement.
5. Pass ‘**contactListItems**’ to the show-contact-list element by using the ‘**list**’ property you already created.  
   You can pass values to component properties like so:

<sub-component property='[[customvalue]]'>

1. Check if the app is still working after the changes.

#### Assignment 3 (Advanced: create edit contact component)

For this assignment you need to create a new custom web component for editing the contact.

1. Create **edit-contact.html** component, this component should have the following fields:
   1. Input field to edit the name
   2. Input field to edit the phone number
   3. Button to save the changes
2. In **show-contact-list.html**:
   1. Import **edit-contact.html**
   2. Create a new ‘Edit’ button for each contact in the list
   3. Use paper-dialog element to show the edit fields for the selected contact. Something like this:

<paper-dialog …>

<edit-contact …></edit-contact>

</paperdialog>

* 1. Create two private functions to show and close the dialog

1. Make it happen!