TRIRIGA Wiki Home
Facilities Management …
Facilities Maintenance
Environmental & Ener…
Real Estate Management
Capital Project Manag…
CAD Integrator-Publis…
IBM TRIRIGA Connect…
IBM TRIRIGA Anywhere
IBM TRIRIGA Applicati…
Release Notes
Media Library
Best Practices
Upgrading
Troubleshooting
UX Framework
UX Articles
UX App Building
Introducing UX
Implementing UX
Extending UX
Implementing UX (…
Extending UX (Poly…
Converting UX to …
Bundling UX (Poly…
Commanding UX (…
UX Perceptive Apps
UX in Foundation To…
UX App Designer Tools
UX Best Practices
UX in Foundation Docs
UX Component Docs
UX Tips & Tricks
UX Videos
UX Archives
New Page
Index
Members
Trash
Tags
Find a Tag
analysis application availability_section best_practices cad change_management changes compare compare_revisions customizations customize database db2 exchange find_available_times gantt_chart gantt_scheduler group memory_footprint modifications modify object_label object_revision operating_system oracle performance platform problem_determination reports reserve reserve_performance revision revisioning single_sign-on snapshot space sql_server sso support system system_performance tags: track_customizations tririga troubleshoot tuning upgrade ux version versioning
Cloud | List

You are in: IBM TRIRIGA > UX Framework > UX App Building > Converting UX to Polymer 3

Converting UX to Polymer 3

Like | Updated July 8, 2019 by Jay.Manaloto | Tags: *None* Add tags

Edit Page Actions

UX Framework | UX App Building | UX in Classic Tools | UX App Designer Tools | UX Best Practices

See the Polymer website at www.polymer-project.org for more about Polymer 3. See the NPM website at www.npmjs.com for more about Node.js. See the Upgrading UX Framework video and Upgrading UX Apps video for more information.

Converting UX to Polymer 3: Keeping up with the latest Polymer standards

STILL HERE? If you're ready for yet another serving, I admire your appetite! In my **previous articles**, we explored the concepts, built a simple UX application, and extended it with new fields, buttons, dialogs, toasts, and ways to manipulate records. This time, we'll discuss how to convert your custom UX apps from the Polymer Version 1 standard to the Polymer Version 3 standard.

- How do we convert from Polymer 1 to Polymer 3?
- How do we get the NPM and TRIRIGA tools?
- How do we use tri-polymer-upgrade?
- What are the details behind the automatic conversion?
- Still confused or curious?

How do we convert from Polymer 1 to Polymer 3?

When the IBM TRIRIGA Application Platform 3.5.0 was released in late 2015, it introduced an MVC-based UX framework for **Polymer**-based applications, also known as the **IBM TRIRIGA UX Framework**. At the time, our UX release was based on **Polymer 1**. Our latest UX release supports the latest **Polymer 3** standard. So our latest UX apps were developed in **Polymer 3**.

To be clear, our UX framework supports **both** Polymer 1 and Polymer 3 UX apps. Both types of UX apps can **coexist** together in the same environment. But if you're interested in converting your own **custom** UX apps from Polymer 1 to Polymer 3, the following tools and processes are available. Be aware that these tools are **not** officially supported at this time.

Notes:

- Before converting a UX view from Polymer 1 to Polymer 3, **remove** the **vulcanized** file if the view is vulcanized. Vulcanized files **cannot** be converted to Polymer 3. Instead, you **must** convert the source component files to Polymer 3 and then use **tri-bundler** to bundle the converted components into a single component file.

How do we get the NPM and TRIRIGA tools?

Contact your IBM TRIRIGA representative or business partner if you cannot access the download location of the **Node.js Package Manager (NPM)** tool. This NPM tool is used to install **several TRIRIGA tools** which allow you to populate the JavaScript (JS) files in your view metadata, preview your JS changes, and sync (deploy) your JS changes with the JS files in your TRIRIGA environment. Be aware that these tools are **not** officially supported at this time.

Download and install the Node/NPM file. For example: **node-v8.12.0-x64.msi**.

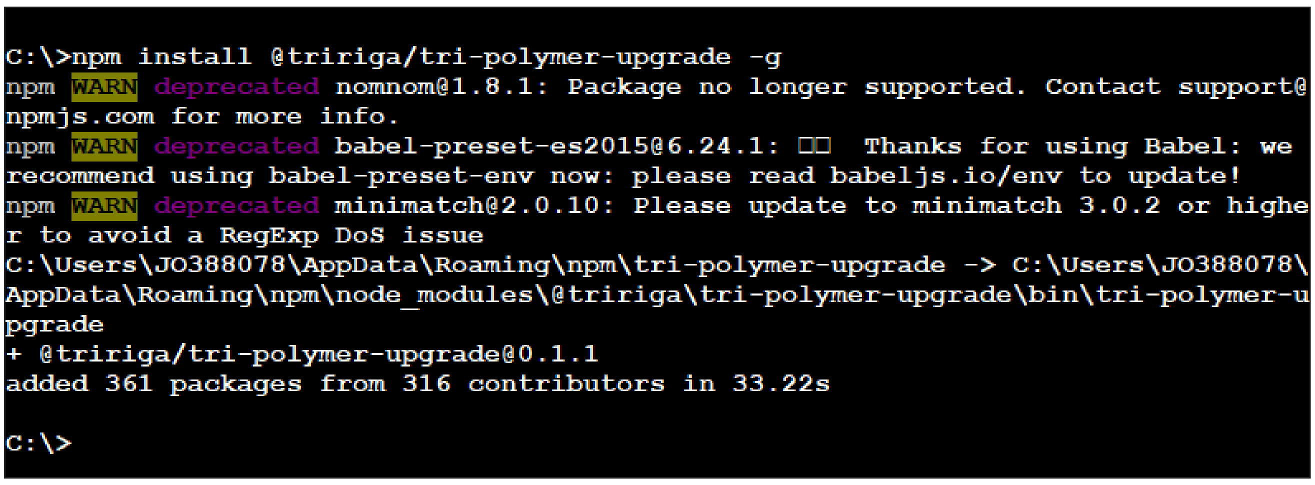
Next, open your command prompt. Run the following NPM command to install the following TRIRIGA tool.

If you see any NPM-related warnings (optional, unsupported, or deprecated), you can ignore them.

TRIRIGA Tool.

Tool	Description
tri-polymer-upgrade	**npm install @tririga/tri-polymer-upgrade -g** This command installs the **tri-polymer-upgrade** tool. **tri-polymer-upgrade** This is a simple tool that automatically converts TRIRIGA UX views, UX components, and UX apps from Polymer 1 to Polymer 3. If you're curious, feel free to check out the **tri-polymer-upgrade** options and details.

NPM > Install TRIRIGA Tool.



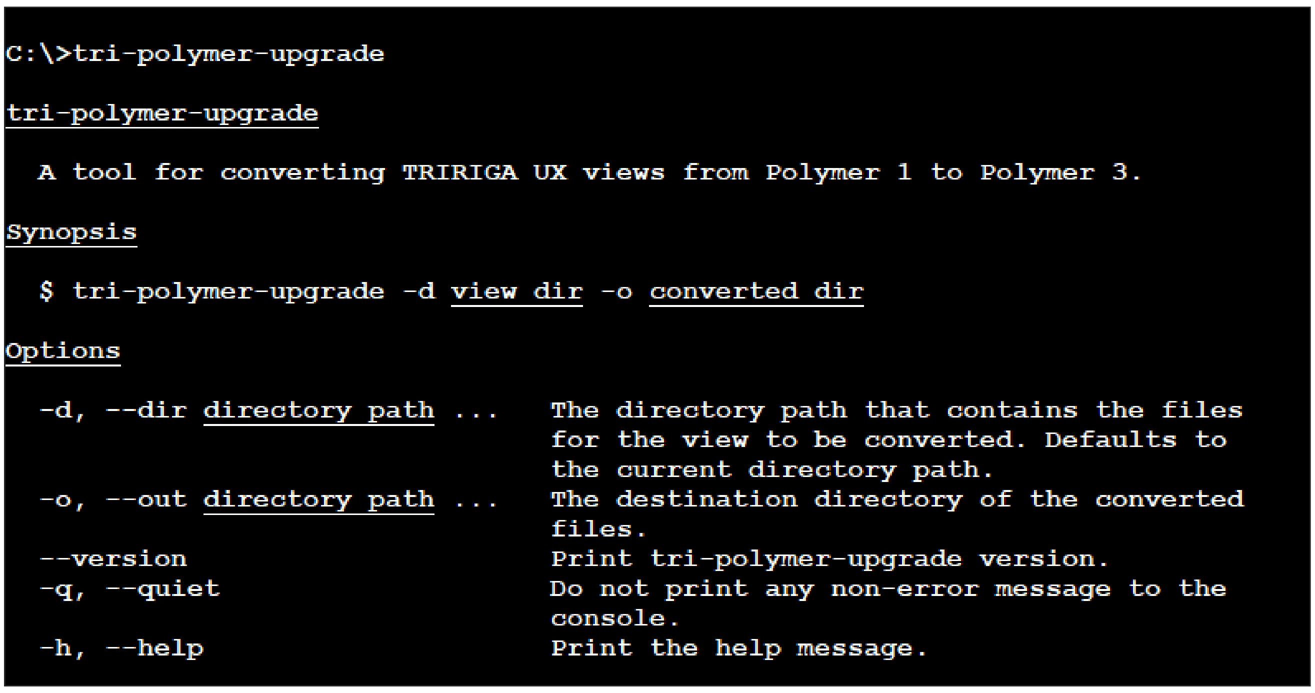
```
C:\>npm install @tririga/tri-polymer-upgrade -g
npm WARN deprecated nomnom@1.8.1: Package no longer supported. Contact support@
npmjs.com for more info.
npm WARN deprecated babel-preset-es2015@6.24.1: □□  Thanks for using Babel: we
recommend using babel-preset-env now: please read babeljs.io/env to update!
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or highe
r to avoid a RegExp DoS issue
C:\Users\JO388078\AppData\Roaming\npm\tri-polymer-upgrade -> C:\Users\JO388078\
AppData\Roaming\npm\node_modules\@tririga\tri-polymer-upgrade\bin\tri-polymer-u
pgrade
+ @tririga/tri-polymer-upgrade@0.1.1
added 361 packages from 316 contributors in 33.22s

C:\>
```

How do we use tri-polymer-upgrade?

Notes:

- During Polymer conversion, it is important that the folder of the source input matches the component name. For example: **/inputPath/triplat-search-input** for **triplat-search-input**.

NPM > tri-polymer-upgrade.



```
C:\>tri-polymer-upgrade

tri-polymer-upgrade

  A tool for converting TRIRIGA UX views from Polymer 1 to Polymer 3.

Synopsis

  $ tri-polymer-upgrade -d view dir -o converted dir

Options

  -d, --dir directory path ...    The directory path that contains the files
                                  for the view to be converted. Defaults to
                                  the current directory path.
  -o, --out directory path ...    The destination directory of the converted
                                  files.
  --version                       Print tri-polymer-upgrade version.
  -q, --quiet                     Do not print any non-error message to the
                                  console.
  -h, --help                      Print the help message.
```

What are the details behind the automatic conversion?

The following details are not arranged or performed in any specific systematic order. The numbering is added for convenience.

1. Fix @apply CSS rule

Before:

@apply(--my-mixin);

After:

@apply --my-mixin;

2. Remove IIFE

An immediately invoked function expression (IIFE) is a JavaScript function that runs as soon as it is defined. For example:

    (function () { statements })();

This conversion step removes any top-level IIFE wrappers. Modules automatically encapsulate their contents.

Before:

(() => { let foo; })();

After:

let foo;

3. Replace content tags

Change <content> insertion points to <slot> elements.
Change <content select="..."> to named slots: <slot name="...">.

Before:

```
<dom-module id="my-el">
    <template>
        ...
        <h2>
            <content select=".title"></content>
        </h2>
        <div>
            <content></content>
        </div>
    </template>
</dom-module>
<script>
    ...
    Polymer.dom(this).querySelectorAll(".title")[0];
    Polymer.dom(this).querySelector(".title");
    this.$$(".title");
    ...
</script>
```

After:

```
<dom-module id="my-el">
    <template>
        ...
        <h2>
            <slot name="title"></slot>
        </h2>
        <div>
            <slot></slot>
        </div>
    </template>
</dom-module>
<script>
    ...
    Polymer.dom(this).querySelectorAll("[slot=title]")[0];
    Polymer.dom(this).querySelector("[slot=title]");
    this.$$("[slot=title]");
    ...
</script>
```

4. Replace CSS content selectors

Before:

```
::content > *
::content > .transition
::content > .transition, ::content > .content
::content [header]
::content > :not(.iron-selected):not(.transition)
#content > ::content:last-child.transition
paper-header-panel > ::content > #mainPanel > #mainContainer
```

After:

```
::slotted(*)
::slotted(.transition)
::slotted(.transition), ::slotted(.content)
::slotted([header])
::slotted(:not(.iron-selected):not(.transition))
#content > ::slotted(:last-child.transition)
*** paper-header-panel > ::slotted(#mainPanel > #mainContainer)
```

*** This last item is not a valid use of slotted pseudo element. In Shadow DOM v1, you cannot select a descendant of a top-level distributed child.

5. Update custom CSS property syntax

Before:

```
color: var(--special-color,--default-color);
```

After:

```
color: var(--special-color, var(--default-color));
```

6. Add slot attribute

Before:

```
<my-el>
        <template is="dom-if">
                <span class="title">Mr. Darcy</span>
        </template>
        <span>Fun at parties.</span>
</my-el>
```

After:

```
<my-el>
        <template is="dom-if">
                <span class="title" slot="title">Mr. Darcy</span>
        </template>
        <span>Fun at parties.</span>
</my-el>
```

7. Export top-level namespace declarations

Before:

```
TriplatSampleBehaviorImpl = {
        properties: {
                ...
        },
        listeners: {
                "event": "_onEvent"
        },
        attached: function() {
                ...
        }
        ...
};
Polymer.TriplatSampleBehavior = [
        TriplatSampleBehaviorImpl,
        Polymer.IronResizableBehavior
];
```

After:

```
export const TriplatSampleBehaviorImpl = {
        properties: {
                ...
        },
        listeners: {
                "event": "_onEvent"
        },
        attached: function() {
                ...
        }
        ...
};
export const TriplatSampleBehavior = [
        TriplatSampleBehaviorImpl,
        IronResizableBehavior
];
```

8. Import non-module JS files

Before:

```
<script src="triplat-multipart-util.js"></script>
```

After:

```
import { importJs } from '../tricore-util/tricore-util.js';
importJs("triplat-multipart-util.js","<packagename>/<filename>");
```

9. Convert HTML imports to JS-module imports

Before:

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

<link rel="import" href="../iron-image/iron-image.html">

<link rel="import" href="../tricore-url/tricore-url.html">

<link rel="import" href="../iron-resizable-behavior/iron-resizable-behavior.html">

<link rel="import" href="triplat-image-orientation-behavior.html">
```

After:

```
import { html } from "../@polymer/polymer/lib/utils/html-tag.js";
import { Polymer } from "../@polymer/polymer/lib/legacy/polymer-fn.js";

import "../@polymer/polymer/polymer-legacy.js";
import "../@polymer/iron-image/iron-image.js";

import "../tricore-url/tricore-url.js";

import { IronResizableBehavior } from "../@polymer/iron-resizable-behavior/iron-resizable-behavior.js";
import { TriplatImageOrientationBehavior, TriplatImageOrientation } from "./triplat-image-orientation-behavior.js";
import { resolveUrl } from "../@polymer/polymer/lib/utils/resolve-url.js";
```

10. Remove namespace from element's behaviors import

Remove Polymer from Polymer.IronOverlayBehaviorImpl.

Before:

```
Polymer({
        is: "triplat-sample",
        behaviors: [
                TriplatImageOrientationBehavior,
                TriplatFileSizeValidationBehavior,
                Polymer.IronResizableBehavior
        ],
        properties : {
                ...
                var propertyDescriptor = Object.getOwnPropertyDescriptor(Polymer.IronOverlayBehaviorImpl, "_focusableNodes");
                ...
        }
});
```

After:

```
import { IronOverlayBehaviorImpl, IronOverlayBehavior } from "../@polymer/iron-overlay-behavior/iron-overlay-behavior.js";

...

Polymer({
        is: "triplat-sample",
        behaviors: [
                TriplatImageOrientationBehavior,
                TriplatFileSizeValidationBehavior,
                IronResizableBehavior
        ],
        properties : {
                ...
                var propertyDescriptor = Object.getOwnPropertyDescriptor(IronOverlayBehaviorImpl, "_focusableNodes");
                ...
        }
});
```

11. Move leading comments to JS module

Before:

```
<!--
@license
IBM Confidential - OCO Source Materials - (C) COPYRIGHT IBM CORP. 2015-2018 - The source code for this program is not
published or otherwise divested of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
-->

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

<!--
Doc for triplat-sample

### Styling

The following custom properties and mixins are also available for styling:
-->

<dom-module id="triplat-sample">

...
```

After:

```
/*
@license
IBM Confidential - OCO Source Materials - (C) COPYRIGHT IBM CORP. 2015-2018 - The source code for this program is not
published or otherwise divested of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
*/

/*
Doc for triplat-sample

### Styling

The following custom properties and mixins are also available for styling:
*/

import { Polymer } from "../@polymer/polymer/lib/legacy/polymer-fn.js";
import "../@polymer/polymer/polymer-legacy.js";

Polymer({

        _template: html`

...
```

12. Update Polymer.dom usage

Before:

```
this._userTemplate = Polymer.dom(this).querySelector('template');
```

After:

```
import { dom } from '../@polymer/polymer/lib/legacy/polymer.dom.js';
...
this._userTemplate = dom(this).querySelector('template');
```

13. Update Polymer.RenderStatus usage

Before:

```
Polymer.RenderStatus.afterNextRender(this, function(){});
```

After:

```
import { afterNextRender } from '../@polymer/polymer/lib/utils/render-status.js';
...
afterNextRender(this, function(){});
```

14. Replace Polymer.isInstance and Polymer.instanceof usage

Before:

```
Polymer.isInstance(element);

if (Polymer.instanceof(this.$.reservationFindRoomPage))
```

After:

```
import { PolymerElement } from '../@polymer/polymer/polymer-element.js';
...
element instanceof PolymerElement

if (this.$.reservationFindRoomPage instanceof PolymerElement)
```

15. Replace Polymer.Base usage

Before:

```
_meta = {value: Polymer.Base.create('iron-meta', {type: 'iconset'})};
Polymer.Base.mixin(computedParams, params);
```

After:

```
import { Base } from '../@polymer/polymer/polymer-legacy.js';

_meta = {value: Base.create('iron-meta', {type: 'iconset'})};
Base.mixin(computedParams, params);
```

16. Replace Polymer.Templatizer usage

Before:

```
Polymer({
    is: "triblock-table-column",
    behaviors: [ Polymer.Templatizer ],
```

After:

```
import {Templatizer} from '../@polymer/polymer/lib/legacy/templatizer-behavior.js';
...
Polymer({
    is: "triblock-table-column",
    behaviors: [ Templatizer ],
```

17. Convert style modules and custom-styles

a. Custom styles

Before:

```
<custom-style>
    <style is="custom-style" include="tristyles-fonts">
        html {
            --ibm-blue-10: rgb(192, 230, 255);
            --ibm-blue-20: rgb(124, 199, 255);
        }
    </style>
</custom-style>
```

After:

```
import { addCustomStyle } from "../tricore-util/tricore-util.js";
...
const customStyle0 = `
<custom-style>
    <style is="custom-style" include="tristyles-fonts">
        html {
            --ibm-blue-10: rgb(192, 230, 255);
            --ibm-blue-20: rgb(124, 199, 255);
        }
    </style>
```

```
</custom-style>
`;
addCustomStyle(customStyle0);
```

b. Style modules

Before:

```
<dom-module id="tristyles-theme">
    <template>
        <style>
            a:not(.tri-disable-theme) {
                text-decoration: none;
                color: var(--tri-primary-color);
            }
        </style>
    </template>
</dom-module>
```

After:

```
import { addDomStyleModule } from "../tricore-util/tricore-util.js";
...
const tristylesFonts = `
<dom-module id="tristyles-theme">
    <template>
        <style>
            a:not(.tri-disable-theme) {
                text-decoration: none;
                color: var(--tri-primary-color);
            }
        </style>
    </template>
</dom-module>
`;
addDomStyleModule(tristylesFonts, "<package-name>/<host-component-filename>");
```

c. Nodes outside an element template

Before:

```
<iron-iconset-svg name="ibm-large" size="128">
    <?xml version="1.0" encoding="utf-8"?>
    <svg version="1.1" id="graph-in-progress">
        <path d="M61,8.009v15.061C85,23.57,104.859,43,105.11,67h15.061C119.919,35,93,8.513,61,8.009z"/>
    </svg>
</iron-iconset-svg>
```

After:

```
import { addDomNodes } from "../tricore-util/tricore-util.js";
...
const domNodesContainer = `
<iron-iconset-svg name="ibm-large" size="128">
    <?xml version="1.0" encoding="utf-8"?>
    <svg version="1.1" id="graph-in-progress">
        <path d="M61,8.009v15.061C85,23.57,104.859,43,105.11,67h15.061C119.919,35,93,8.513,61,8.009z"/>
    </svg>
</iron-iconset-svg>
`;
addDomNodes(domNodesContainer);
```

18. Add tristyles-theme to all elements

Before:

```
<dom-module id="triplat-image">
    <style>
    ...
    </style>
    ...
</dom-module>
```

After:

```
Polymer({
    _template: html`
        <style include="tristyles-theme">
        ...
        </style>
        ...
    `,
    ...
});
```

19. Fix loadResource and resolveUrl method usage

If the component or behavior uses the TriLazyLoadingBehavior and/or calls this.loadResource method or calls this.resolveUrl, the following changes are needed.

a. loadResource

Before:

```
Polymer({
    behaviors: [
        TriLazyLoadingBehavior,
        ...
    ],
    ...
    _onTaskInboxRouteActive: function(e) {
        if (e.detail.active) {
            this.loadResource(this.$.inboxPage, "tripage-task-inbox.html");
        }
    },
    ...
});
```

After:

```
import { getModuleUrl } from "../tricore-util/tricore-util.js";
Polymer({
    behaviors: [
        TriLazyLoadingBehavior,
        ...
    ],
    ...
    _onTaskInboxRouteActive: function(e) {
        if (e.detail.active) {
            this.loadResource(this.$.inboxPage, "tripage-task-inbox.js");
        }
    },
    importMeta: getModuleUrl("<this package name>/<this file name.js>"),
    ...
});
```

b. resolveUrl

Before:

```
this.frameurl = this.resolveUrl("../forge-viewer/ForgeViewer.html");
```

After:

```
import { getModuleUrl } from "../tricore-util/tricore-util.js";
...
this.frameurl = this.resolveUrl("../forge-viewer/ForgeViewer.html");
...
importMeta: getModuleUrl("<this package name>/<this file name.js>"),
...
```

20. Update copyright year

Before:

```
/*
@license
IBM Confidential - OCO Source Materials - (C) COPYRIGHT IBM CORP. 2015-2016 - The source code for this program is not
published or otherwise divested of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
*/

/*
@license
IBM Confidential - OCO Source Materials - (C) COPYRIGHT IBM CORP. 2015 - The source code for this program is not
published or otherwise divested of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
*/
```

After:

```
/*
@license
IBM Confidential - OCO Source Materials - (C) COPYRIGHT IBM CORP. 2015-2018 - The source code for this program is not
published or otherwise divested of its trade secrets, irrespective of what has been deposited with the U.S. Copyright Office.
*/

// both the same
```

21. Fix iron-meta due to removed list property

Because iron-meta removed the list property.

Before:

```
<link rel="import" href="../iron-meta/iron-meta.html">

...

<iron-meta type="iconset" list="{{iconsets}}"></iron-meta>

...

properties: {
    iconsets: {
        type:Array
    }
}
```

After:

```
import { IronMeta } from "../@polymer/iron-meta/iron-meta.js";

...

<iron-meta type="iconset"></iron-meta>

...

properties: {
    iconsets: {
        type:Array,
        value: new IronMeta({ type: "iconset" }).list
    }
}
```

22. Check observers for undefined parameters

Observers need to check for undefined arguments, which was not an issue in Polymer 1.x. A warning will output if the method cannot be found (perhaps in a behavior).

a. Example 1

Before:

```
Polymer({
    ...
    observers:[
        "_handleFieldChange(fieldA, 'abc', fieldB)"
    ]
    ...
    _handleFieldChange: function(fieldA, pLiteral, fieldB) {
        //function code
    }
});
```

After:

```
import { assertParametersAreDefined } from '../tricore-util/tricore-util.js';
...
Polymer({
    ...
    observers:[
        "_handleFieldChange(fieldA, 'abc', fieldB)"
    ]
    ...
    _handleFieldChange: function(fieldA, pLiteral, fieldB) {
        if (!assertParametersAreDefined(arguments)) {
            return;
        }
        //function code
    }
});
```

b. Example 2

Before:

```
<dom-module id="abc-component">
    <template>
        <test-sample param="[[_computeParam(fieldA, FieldB)]]">
    </template>
</dom-module>
...
Polymer({
    ...
    _computeParam: function(fieldA, fieldB) {
        //function code
    }
});
```

After:

```
import { assertParametersAreDefined } from '../tricore-util/tricore-util.js';
...
Polymer({
    _template: html`
        <test-sample param="[[_computeParam(fieldA, FieldB)]]">
    `;
    ...
    _computeParam: function(fieldA, fieldB) {
        if (!assertParametersAreDefined(arguments)) {
            return;
        }
        //function code
    }
});
```

c. Example 3

Before:

```
Polymer({
    ...
    properties: {
        fullName: {
            type: String,
            computed: 'computeFullName(first, last)'
        }
    },
    ...
    computeFullName: function(first, last) {
        //function code
    }
});
```

After:

```
import { assertParametersAreDefined } from '../tricore-util/tricore-util.js';
...
Polymer({
    ...
    properties: {
        fullName: {
            type: String,
            computed: 'computeFullName(first, last)'
        }
    },
    ...
    computeFullName: function(first, last) {
        if (!assertParametersAreDefined(arguments)) {
            return;
        }
        //function code
    }
});
```

d. Example 4

Before:

```
<div>[[_computeValue(first, last)]]"</div>
<div>[[_tt1]] - {{_tt2(height, width)}}</div>

computeValue: function(first, last) {
    //function code
}

_tt2: function(height, width) {
    // function code
}
```

After:

```
import { assertParametersAreDefined } from '../tricore-util/tricore-util.js';
...
<div>[[_computeValue(first, last)]]"</div>
[[_tt1]] - {{_tt2(height, width)}}

computeValue: function(first, last) {
    if (!assertParametersAreDefined(arguments)) {
        return;
    }
    //function code
}

_tt2: function(height, width) {
    if (!assertParametersAreDefined(arguments)) {
        return;
    }
    // function code
}
```

23. Convert <input is="iron-input"> type-extension

Polymer 2.0+ doesn't support type-extension elements, for example, <input is="iron-input">. Refactor type-extension elements as wrapper elements. Wrap existing type-extension elements. Add import of iron-input, if missing. Properties on <iron-input> include: bind-value, allowedPattern, autoValidate, validator, and invalid.

Before:

```
<input id="[[_inputId]]"
    placeholder="[[_computePlaceHolder(placeholder)]]"
    aria-labelledby="[[_labelId]]"
    is="iron-input"
    bind-value="{{_searchValue}}"
    disabled$="[[disabled]]"
    aria-label$="{{label}}"
    on-tap="_onInputTapped"
    allowed-pattern="[[allowedPattern]]"
    validator="[[validator]]"
    invalid="{{invalid}}"
    class="input-element"
    id$="[[_inputId]]"
    maxlength$="[[maxlength]]">
```

After:

```
import "../@polymer/iron-input/iron-input.js";
...
<style>
    input {
        @apply --paper-input-container-shared-input-style;
    }
</style>

<iron-input id="[[_inputId]]"
    bind-value="{{_searchValue}}"
    allowed-pattern="[[allowedPattern]]"
    validator="[[validator]]"
    invalid="{{invalid}}"
    class="input-element"
    id$="[[_inputId]]"
    maxlength$="[[maxlength]]">
    <input
        placeholder="[[_computePlaceHolder(placeholder)]]"
        aria-labelledby="[[_labelId]]"
        disabled$="[[disabled]]"
        aria-label$="{{label}}"
        on-tap="_onInputTapped">
</iron-input>
```

24. Convert paper-menu to paper-listbox

After Polymer 2.x, paper-menu does not exist, but it can be replaced with paper-listbox. However, there are some style mixins that are not available. **Note**: Be careful not to convert paper-menu-button.

Before:

```
<link rel="import" href="../paper-menu/paper-menu.html">
...
<template>
...
<style>
    paper-menu, paper-menu-button {
        --paper-menu-button-color: gold;
    }
    paper-menu {
        --paper-menu-background-color: var(--triblock-app-layout-mobile-drawer-background-color, --tri-primary-color-70);
        --paper-menu-color: var(--triblock-app-layout-mobile-drawer-color, --tri-header-color);
        --paper-menu-selected-item: {
            opacity: 1;
            font-weight: normal;
        };
        --paper-menu-focused-item: {
            background-color: var(--triblock-app-layout-mobile-drawer-hover-background-color, --tri-primary-color-80);
        };
        --paper-menu-focused-item-after: {
            background-color: var(--triblock-app-layout-mobile-drawer-background-color, --tri-primary-color-70);
        };
        --paper-menu: {
            padding: 0 0;
        };
    }
</style>
<paper-menu id="mobileMenu" selected="1">
    ...
</paper-menu>
</template>
...
<script>
    ...
    this.$$("paper-menu").selected = "2";
    var menu = Polymer.dom(root).querySelector("paper-menu");
    var menu2 = Polymer.dom(root).querySelectorAll("paper-menu");
    ...
</script>
```

After:

```
import "../@polymer/paper-listbox/paper-listbox.js";
...
// template
<style>
    paper-listbox, paper-menu-button {
        --paper-menu-button-color: gold;
    }
    paper-listbox {
        --paper-listbox-background-color: var(--triblock-app-layout-mobile-drawer-background-color, var(--tri-primary-color-70));
        --paper-listbox-color: var(--triblock-app-layout-mobile-drawer-color, var(--tri-header-color));
        --paper-listbox-selected-item: {
            opacity: 1;
            font-weight: normal;
        };
        --paper-listbox-focused-item: {
            background-color: var(--triblock-app-layout-mobile-drawer-hover-background-color, var(--tri-primary-color-80));
        };
        --paper-listbox-focused-item-after: {
            background-color: var(--triblock-app-layout-mobile-drawer-background-color, var(--tri-primary-color-70));
        };
        --paper-listbox: {
            padding: 0 0;
        };
    }
</style>
<paper-listbox id="mobileMenu" selected="1">
    ...
</paper-listbox>
...
// js
    ...
    this.$$("paper-listbox").selected = "2";
    var menu = dom(root).querySelector("paper-listbox");
    var menu2 = dom(root).querySelectorAll("paper-listbox");
```

25. Convert Nodelist to Array for dom querySelectorAll

Before:

```
var bannerButtonElements = dom(this).querySelectorAll("triblock-banner-button");
```

After:

```
var bannerButtonElements = Array.from(dom(this).querySelectorAll("triblock-banner-button"));
```

26. Fix the usage of triplat-word-highlight

Polymer 2.0+ doesn't support type-extension elements, for example, <input is="iron-input">. The triplat-word-highlight is an extension of the span element, but in Polymer 3, the component was changed to no longer be an extension. All components that are upgraded to Polymer 3 and also use the triplat-word-highlight must be changed as described bellow.

Before:

```
<span class="child-text" is="triplat-word-highlight" value="{{child.value}}" search-value="{{value}}"></span>
```

After:

```
<triplat-word-highlight class="child-text" value="{{child.value}}" search-value="{{value}}"></triplat-word-highlight>
```

27. Replace Polymer.CaseMap.camelToDashCase() with camelToDashCase()

Also replace Polymer.CaseMap.dashToCamelCase() with dashToCamelCase().

Before:

```
this.listen(this, Polymer.CaseMap.camelToDashCase(property) + "-changed", "_propagate");
let camelCaseProperty = Polymer.CaseMap.dashToCamelCase(property);
```

After:

```
import { camelToDashCase, dashToCamelCase } from "../@polymer/polymer/lib/utils/case-map.js";
...
this.listen(this, camelToDashCase(property) + "-changed", "_propagate");
let camelCaseProperty = dashToCamelCase(property);
```

28. Add missing imports for behaviors and exports

This conversion step is also for -import files, since we are moving the imports out of the -import file to the original file.

a. Example 1

Before:

```
    behaviors : [
        IronResizableBehavior
    ],
```

After:

```
import { IronResizableBehavior } from "../@polymer/iron-resizable-behavior/iron-resizable-behavior.js";
...
    behaviors : [
        IronResizableBehavior
    ],
```

b. Example 2

Before:

```
    this.myTaskDS.updateRecord(task._id, TriPlatDs.RefreshType.SERVER, "actions", "activate");
```

After:

```
import { TriPlatDs } from "../triplat-ds/triplat-ds.js";
...
    this.myTaskDS.updateRecord(task._id, TriPlatDs.RefreshType.SERVER, "actions", "activate");
```

29. Move property initializations that use this.$ or this.$$ to ready callback

Move property initializations that use "this.$" or "this.$$" (Polymer.dom(this.root).querySelector()) to "ready" callback.

Before:

```
    _searchFieldEl: {
        type: Object,
        value: function() {
            return this.$.searchField;
        }
    },
```

After:

```
    _searchFieldEl: {
        type: Object
    },
...
ready: function() {
    this.set("_searchFieldEl", this.$.searchField);
}
```

30. Convert customStyle to updateStyles

Because the customStyle instance property was removed, use updateStyles instead.

Before:

```
this.customStyle['--tri-datetime-picker-calendar-position-top'] = _top + " !important";
        this.customStyle['--tri-datetime-picker-calendar-position-left'] = _left + " !important";
        this.customStyle['--tri-datetime-picker-calendar-position'] = _position + " !important";
```

After:

```
this.updateStyles({
        '--tri-datetime-picker-calendar-position-top': _top + " !important",
        '--tri-datetime-picker-calendar-position-left': _left + " !important",
        '--tri-datetime-picker-calendar-position': _position + " !important"
});
```

31. Add web-animations-next-lite.min.js import

Add web-animations-next-lite.min.js import if the component uses neon-animation or paper-dropdown-menu.

a. neon-animation

Before:

```
<link rel="import" href="../neon-animation/neon-animation-runner-behavior.html">
<link rel="import" href="../neon-animation/animations/slide-from-left-animation.html">
<link rel="import" href="../neon-animation/animations/slide-from-right-animation.html">
<link rel="import" href="../neon-animation/animations/fade-in-animation.html">
```

After:

```
import { NeonAnimationRunnerBehaviorImpl, NeonAnimationRunnerBehavior } from "../@polymer/neon-animation/neon-animation-runner-behavior.js";
import "../@polymer/neon-animation/animations/slide-from-left-animation.js";
import "../@polymer/neon-animation/animations/slide-from-right-animation.js";
import "../@polymer/neon-animation/animations/fade-in-animation.js";
...
const importJsPromise = importJs(["../web-animations-js/web-animations-next-lite.min.js"], "triplat-date-picker/triplat-calendar.js");
```

b. paper-dropdown-menu

Before:

```
<link rel="import" href="../paper-dropdown-menu/paper-dropdown-menu.html">
```

After:

```
import "../../@polymer/paper-dropdown-menu/paper-dropdown-menu.js";
...
const importJsPromise = importJs(["../web-animations-js/web-animations-next-lite.min.js"], "triplat-date-picker/triplat-calendar.js");
```

32. Fix :host property selectors and fix :host in :host-context selectors

Fix ":host" property selectors by wrapping them with (), and fix the mixing of ":host" in ":host-context" selectors.

Before:

```
        :host[small-screen-width] .search {
                padding: 15px;
        }
        :host[small-screen-width]:not([opened]) .duration {
                margin-bottom: 10px;
        }
        :host.narrow {
                flex: var(--triblock-table-column-flex, 0.5);
        }
        :host.wide {
                flex: var(--triblock-table-column-flex, 2);
        }
        :host-context([dir="ltr"]):not([small-screen-width]):not(:last-of-type) {
        }
        :host-context([dir="ltr"]):not([small-screen-width]):not(:last-of-type) .request {
        }
```

After:

```
        :host([small-screen-width]) .search {
                padding: 15px;
        }
        :host([small-screen-width]:not([opened])) .duration {
                margin-bottom: 10px;
        }
        :host(.narrow) {
                flex: var(--triblock-table-column-flex, 0.5);
        }
        :host(.wide) {
                flex: var(--triblock-table-column-flex, 2);
```

```
    }
    :host-context([dir="ltr"]):host(:not([small-screen-width]):not(:last-of-type)) {
    }
    :host-context([dir="ltr"]):host(:not([small-screen-width]):not(:last-of-type)) .request {
    }
```

33. Replace :host-context([dir="rtl"]) with :host([dir="rtl"]) and add tricore-dir-behavior

Be sure to replace for dir="ltr" and dir="rtl".

Before:

```
    :host-context([dir="ltr"]) .header-count {
        margin-left: 7px
    }
    :host-context([dir="rtl"]):host(:not([small-screen-width]):last-of-type) {
        border-radius: 10px 0 0 10px;
    }
    :host-context([dir="ltr"]):host(:not([small-screen-width]):not(:last-of-type)) .request {
    }
```

After:

```
import { TricoreDirBehavior } from "../tricore-dir-behavior/tricore-dir-behavior.js";
...
    behaviors: [
        TricoreDirBehavior
    ],
    ...
    :host([dir="ltr"]) .header-count {
        margin-left: 7px
    }
    :host([dir="rtl"]:not([small-screen-width]):last-of-type) {
        border-radius: 10px 0 0 10px;
    }
    :host([dir="ltr"]:not([small-screen-width]):not(:last-of-type)) .request {
    }
```

34. Remove "id" in the listener, if there is an "id" specified, leaving only the event

For example, if "id" = "ajax".

Before:

```
    listeners: {
        "ajax.response": "_onResponse",
        "ajax.error": "_onError"
    },
```

After:

```
    listeners: {
        "response": "_onResponse",
        "error": "_onError"
    },
```

35. Replace Polymer.dom(someComponent).flush() with just flush()

Only /triplat-graphic/triplat-graphic-pin-tooltip and /triplat-query/triplat-query-scroll-page uses it.

Before:

```
Polymer.dom(this.$.tooltip).flush();
```

After:

```
import { dom, flush } from "../@polymer/polymer/lib/legacy/polymer.dom.js";
...
flush();
```

Still confused or curious?

If you have any questions about UX that weren't answered in this article, feel free to reach out to your IBM TRIRIGA representative or business partner. Or if you want, I'll go ask the team.

Next >

Comments (1) | Versions (9) | **Attachments (2)** | About

1-2 of 2 Previous | Next

image300.png	March 28, 2019	22 KB	Edit | Delete
image301.png	June 12, 2019	15 KB	Edit | Delete

Add an attachment

Show 10 | 25 | 50 items per page Previous | Next

Feed for this page | Feed for these attachments