Polymer.js Cheat Sheet

by Jonathan Beri (jonathanberi) via cheatography.com/10955/cs/1315/

Using elements

<script src="platform.js"></script>

Load the polyfills

<link rel="import" href="x-foo.html">

Import components used on the page

<x-foo></x-foo>

Declare the component by its tag

<x-foo attr color="blue"></x-foo>

Published properties as attributes

<x-foo>My foo</x-foo>

My foo is part of the light dom

<x-foo><x-bar></x-bar></x-foo>

Composing elements. *<x-bar>* is part of the light dom

document.querySelector('x-foo').customMethod();

Accessing methods

document.querySelector('x-foo').color = 'red';

Accessing properties

window.addEventListener('polymer-ready', function(e) { document.querySelector('x-foo').barProperty = 'baz';});

Wait for the 'polymer-ready' event before interacting with the element

Element declaration

```
<polymer-element name="tag-name"
constructor="TagName">
<template>
<!-- shadow DOM here -->
</template>
<script>
Polymer('tag-name');
</script>
</polymer-element>
```

Element lifecycle methods

created()

an instance of the element is created

ready()

The <polymer-element> has been fully prepared (e.g. Shadow DOM created, property observers setup, event listeners attached, etc.)

attached()

an instance was inserted into the document

domReady()

Called when the element's initial set of children are guaranteed to exist. (after **attached()**)

detached()

an instance was removed from the document

attributeChanged()

an attribute was added, removed, or updated

```
Polymer('tag-name', {
...
});
```

Expressions

foo, foo.bar.baz

Identifiers & paths. These values are treated as relative to the local model, extracted, observed for changes and cause the expression to be re-evaluated if one or more has changed.

!

Logical not operator

+foo, -bar

Converted to Number. Or converted to Number, then negated.

foo + bar, foo - bar, foo * bar

Supported: <, >, <=, >=, ==, !=, ===, !==

foo && bar || baz

Logical comparators

a ? b : c

Ternary operator

(a + b) * (c + d)

Grouping (parenthesis)

numbers, strings, null, undefined

Literal values. Escaped strings and non-decimal numbers are not supported.

By **Jonathan Beri** (jonathanberi)
cheatography.com/jonathanberi/
www.jonathanberi.com

Published 20th October, 2013.
Last updated 11th May, 2016.
Page 1 of 3.

Expressions (cont)

[foo, 1], {id: 1, foo: bar}

Array & Object initializers

foo: bar.baz; bat: boo > 2;

Labeled statements

Firing custom events

this.fire('ouch', {msg: 'That hurt!'});

// fire(inType, inDetail, inToNode)

document.querySelector('ouch-button').addEventListener('ouch', function(e) { console.log(e.type, e.detail.msg); // "ouch" "That hurt!" });

Listening outside the element

<ouch-button on-ouch="{{ myMethod }}"></ouch-button>

Using on-* handlers within another Polymer element

Change watcher

propertyNameChanged

When the value of a watched property changes, the appropriate change handler is automatically invoked.

```
<polymer-element name="g-cool"
attributes="better best">
<script>
Polymer('g-cool', {
betterChanged: function(inOldValue) {
},
bestChanged: function(inOldValue) {
}
});
</script>
</polymer-element>
```

Event mapping

<template><input on-keypress="{{ keypressHandler }}"></input></template>

on-keypress declaration maps the standard DOM "keypress" event to the keypressHandler method defined on the element

buttonClick: function(event, detail, sender) { ... }

on-* handler

event

inEvent is the standard event object.

detail

inDetail: A convenience form of inEvent.detail.

sender

A reference to the node that declared the handler. This is often different from inEvent.target (the lowest node that received the event) and inEvent.currentTarget (the component processing the event), so Polymer provides it directly.

Element attributes

<polymer-element name="tag-name"><polymer-element>

Name for the custom element. Requires a "-".

<polymer-element attributes="color"><polymer-element>

Published properties

<polymer-element extends="other-element"><polymer-element>

Extend other elements

<polymer-element noscript><polymer-element>

For simple elements that don't need to call Polymer().

Element attributes (cont)

<polymer-element lightdom><polymer-element>

For simpler elements that don’t require the features of Shadow DOM, use the lightdom attribute to control how the element stamps out DOM.

<polymer-element constructor="TagName"><polymer-element>

The name of the constructor to put on the global object. Allows users to create instances of your element using the new operator (e.g. var tagName = new TagName()).

Automatic node finding

this.$.nameInput.value

<template><input type="text" id="nameInput"></template>

Extending elements

<polymer-element name="p-el" extends="p-2"></polymer-element>

A Polymer element can extend another element by using the extends attribute. The parent’s properties and methods are inherited by the child element and data-bound. You can override any attribute or method

this.super();

Calls the parent's method

<link rel="import" href="x-foo.html">

Import the file with the extended element if not in the same file

By **Jonathan Beri** (jonathanberi)
cheatography.com/jonathanberi/
www.jonathanberi.com

Template syntax

<template>{{ owner }}</template>

You can bind properties in your component using declarative data binding and the “double-mustache” syntax ({{}}).

<template repeat="{{ user,i in users }}"></template>

repeats one instance for each item in the array 'users'

<template bind="{{ foo as bar }}"></template>

Named scopes are useful for referencing a model value from an “outer” model “scope”.

<template if="{{ conditionalValue }}"></template>

Binds if and only if conditionalValue is truthy.

<template repeat if="{{ conditionalValue }}"></template>

Repeat if and only if conditionalValue is truthy.

<template bind ref="myTemplate"></template>

When creating an instance, the content of this template will be ignored, and the content of #myTemplate is used instead.

<content select="h2"></content>

When a tag is rendered, the content of the shadow host is projected into the spot that the <content> element appears.

<input type="text" value="this value is inserted once: [[obj.value]]">

double brackets ([[]]) be used in place of {{}} to setup a one-time binding. The binding becomes inactive after Polymer sets its value for the first time.

```
<polymer-element name="tk-element-databinding">
<template>{{owner}}</strong></template>
<script>
Polymer('tk-element-databinding', {
owner: 'Daniel',
users: [1,2,3]
});
</script>
</polymer-element>
```