Cheatography

Polymer 1.1 Cheat Sheet

by WebPlanet (Pawel) via cheatography.com/25918/cs/7033/

Register and Create

```
// Register element 'my-element' and return constructor
var elemen tConstr = Polyme - r({is: 'my-el eme nt'})
// Register element 'my-el - ement' with factory implem ent - ation
var elemen tConstr =Polymer( { is: 'my-el ement',
  factor yImpl function (v1, v2) {
        this.v1 = v1;
        this.v2 = v2
}});
// Create element
var myElement = new elemen tCo - nstr();
OR
var myElement = docume nt.c re - ate Ele men t(' my- ele ment');
Create element with factory implem ent ation
var myElement = new elemen tCo - nstr(1, 2);
```

Extend existing element

```
// Extend existing input element
var myElement = Polymer({
   is: 'my-in put',
   extends: 'input'
});
```

Data binding

{{prop}}	bind property 'prop', get child custom element notifications
[[prop]]	bind property 'prop', DO NOT get child custom element notifications
{{prop::eventname}}	bind property 'prop' when event 'event-name' occurs

Data binding (cont)

{{obj.prop}}	bind property 'prop' of 'obj' object property
{{!bool}}	negate and bind boolean property 'bool'
{{compute(a, b)}}	execute 'compute(a, b)' and bind
<my-element selected$="{{value}}"> </my-element>	bind 'value' to attribute 'selected'
<my-element selected="{{value}}"> </my-element>	bind 'value' to property 'selected'

Behaviors

```
my-behavior.html
<sc rip t>
myBehavior = {
    listeners: {'click' : 'chang eCo lor'},
    change Color function () {
            thi s.color = 'blue';
        }
}
</s cri pt>
my-ele men t.html
<link rel="im por t" href="m y- b eha vio r.h tml ">
Polymer({
  is: 'my-el ement',
  behaviors: [myBeh avior]
});
```

Register observers

```
Polymer({
        is: 'my-el ement',
        properties: {
            color: String,
            arr: {
                type: Array,
>               value: function () {return [];}
            },
            obj: {
              type: Object
            }
        },
    observers: [
  'obs1(obj.prop2)', 'obj2(color)',
  'obs3(obj.prop2, color)', //Observe mupliple
  'obs4(obj.*) //Observe deep,
  'obs5(arr.splices) //Observe array splices
    ]});
```

Register observers (cont)

Debouncing

```
Polymer({
  is: 'my-el ement',
  properties: {
        prop1: String,
        prop2: String
   },
  observers: ['doSo met hin - g(p rop1, prop2)'],
  doSome thing function () {
    this.d ebounce('doSo met - hing', function () {
            con sol e.l - og( 'called once...');
        }, 300);
   }
});
```

When 'prop1' and 'prop2' are changed within 300ms time frame, 'doSomething' callback is executed once.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Polymer 1.1 Cheat Sheet

by WebPlanet (Pawel) via cheatography.com/25918/cs/7033/

Lifecycle Callbacks

created	when element is registered
ready	when element is registered and local DOM is ready
attached	when element is attached to DOM
detached	when element is removed from DOM
attribute-Changed	when element attribute is changed

Styling

:host	style host element
:host ::content	style distributed content
--custom-property: value	create css property 'custom-property' and give it a 'value'
--mixin-name: {mixin contents}	create mixin 'mixin-name' with contents 'mixin contents'
color: var(---my-color, red)	apply custom property 'my color', set default to red
color: @apply(---mixin-name)	apply mixin 'mixin-name' to a property 'color'

Import styles

style.html

```
<dom-module id=" my- sty le">
        <te mpl ate>
                <st yle>
                        :host {
                                -
--t ext -style: blue;
                        }
```

Import styles (cont)

```
>       </style>
   </template>
</dom-module>
```

my-element.html

```
<link rel="import" href="style.html">
<dom-module id="my-element">
   <style include="my-style"></style>
</dom-module>
```

Style from light DOM

```
<style is="custom-style">
                my- element {
                        --t -
ext -style: red;
                }
</style>
```

Dom repeat

```
<template is="dom-repeat" items=
{{table}} as="myTable">
<div>{{myTable.item}}</div>
</t emp lat e>
...
Polymer ({ ...
          ready: function ()
{this.t able = [{item: 'item1'},
{item: 'item2 '}];}
...});
```

Sort and Filter

```
<template is="dom-repeat" items=
{{table}} filler="myFilter"
sort="mySort">
<div>{{table.item}}</div>
</t emp lat e>
...
Polymer ({ ...
     ready: function ()
{this.t able = [{item: 'item1'},
{item: 'item2 '}];},
```

Sort and Filter (cont)

```
>    myFilter: function (item) {...},
   mySort: function (item) {...}
...});
```

Extending Behaviors

```
<link rel="import"
href="oldbehavior.html">
<sc rip t>
    New Beh avi orImpl = {
   }
  NewBeh avior = [ OldBeh -
avior, NewBeh avi orImpl ]
</s cri pt>
```

Add event listeners

```
// Add event listeners as a
element prototype property
<do m-m odule id=" my- ele men -
t">
<te mpl ate>
     div id=" ele men tID -
1">E lement 1</div>
     div id=" ele men tID -
2">E lement 1</div>
</t emp lat e>
</d om- mod ule>
{
   Polymer({
    is: 'my-el ement',
    listeners: {
               'cl ick.el eme -
ntID1' : 'handl eCl ick1',
               'cl ick.el eme -
ntID2' : 'handl eCl ick2'
               'click' : 'handl -
eClick' // Every click
});
// Adding event listeners on
elements
<el ement on-{event name} = " -
{ha ndl er} " ></ ele men t>
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Polymer 1.1 Cheat Sheet

by WebPlanet (Pawel) via cheatography.com/25918/cs/7033/

Observe mutations

```
this._observer =
Polymer.dom(this).
      observ eNodes(function
(info) {

                                -
con sol e.log ('Added nodes: ' +
info.addedNodes);

                                -
con sol e.log ('Removed nodes:
'+ info.remove dNodes);
});
Polyme r.dom(this).unobse rve -
Nodes(this._ ob server)
```

Property Definition

type	type of a property (String, Array, Boolean, Number, Object)
value	default value
readOnly	set read only property
notify	fire 'property-changed' event, when property is changed. Handy for nested data binding
reflectTo-Attribute	update attribute when property is changed
observer	execute provided Callback when property is changed
computed	compute value based on other property values

Example for *color* and *underline* properties:

properties: {
color: String,
underline: {
type: Boolean,
value: false,
observer: 'applyDecoration',
reflectToAttribute: true,
notify: true
}

Property modifications

this.set('property', value)	set this.property to 'value'
this.set('arr.1', value)	set this.arr[1] to 'value'
this.set('obj.prop1', value)	set this.obj.prop1 to 'value'
this.get('property')	get value of this.property
this.get('obj.prop1')	get value of this.obj.prop1
this.get('arr.1')	get value of this.arr[1]

This API is required to observe properties of Array and Object types.

Polymer API

Dom manipu-lation	Polymer.dom(ele-ment).*method*
Shadow root	this.root
Host element	this
this.$.myid	local DOM element with id = 'myid'
this.$$(s-elector)	"querySelector" in local DOM
contentEl-ement.get-Distribut-edNodes()	return distributed nodes to contentElement
element.g-etDestina-tionInser-tionPo-ints()	return <content> elements for light DOM 'element', where 'element' is distributed to local DOM

Polymer API (cont)

contentEl-ement.get-ContentCh-ildren()	return distributed elements to local DOM that are children of 'contentElement'
.{query | get}Effec-tiveChild-ren()	return effective children Handy for nesting when host element contains <content> tag
.getEffec-tiveChild-Nodes()	same as above for nodes

Conditional Template

```
<template is="dom-if" if="
{{user.isAdmin}}">
          Only admins can see
this.
</t emp lat e>
```

Utility Functions

toggle-Class()	toggle class
toggleAttrib-ute()	toggle attribute
attributeFol-lows()	move attribute from one element to another
classFoll-ows()	move class from one element to another
async()	call asynchronously
fire()	dispatch event
cancelDeb-ouncer()	cancel given debouncer
flushDebo-uncer()	call debouncer task and cancel debouncer
isDebounc-erActive()	return true if active, false otherwise
transform()	tranform element (css transform function)

By **WebPlanet** (Pawel)
cheatography.com/pawel/

Published 6th February, 2016.
Last updated 7th May, 2016.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com