build custom element

Polymer 1.x	Polymer({})
Polymer 2.0	class MyElement extends Polymer.Element {}

API

this.shadowRoot	get shadow DOM
this.$.container	access identifier #container
<template is="-dom-bind">	use Polymer bindings without defining a new custom element
$$(selector)	Returns the first node in this element's local DOM that matches selector

instance methods https://www.polymer-project.org/1.0/docs/devguide/instance-methods

Instance Methods

fire(type, [detail], [options])	Fires a custom event
async(method, [wait])	Calls method asynchronously
debounce(jobName, callback, [wait])	Call debounce to collapse multiple requests for a named task into one invocation
toggleAttribute(name, bool, [node])	toggles the named boolean attribute
this.transform('rotateX(90deg)', this.$.myDiv)	Applies a CSS transform to the specified node, or host element if no node is specified
resolveUrl(url)	returns a path relative to the current document

instance methods https://www.polymer-project.org/1.0/docs/devguide/instance-methods

property list for polymer base

https://www.polymer-project.org/1.0/docs/api/Polymer.Base

Polymer.Templatizer behavior

```
// Get a template from somewhere, e.g. light DOM
var template = Polyme r.d om( thi s).q ue ryS ele - cto r(' tem pla te');
// Prepare the template
this.t emp lat ize (te mpl ate);
// Instance the template with an initial data model
var instance = this.s tam p({ myProp: 'initi - al'});
// Insert the instance's DOM somewhere, e.g. light DOM
Polyme r.d om( thi s).a pp end Chi ld( ins tan - ce.r oot);
// Changing a property on the instance will propagate to bindings
// in the template
instan ce.m yProp = 'new value';
```

The Polymer.Templatizer behavior adds methods to generate instances of templates that are each managed by an anonymous Polymer.Base instance.

https://www.polymer-project.org/1.0/docs/api/Polymer.Templatizer

Lifecycle

created

ready

attached

detached

attributeChanged

Property name to attribute name mapping

Attribute names are converted to lowercase property names

Attribute names with dashes are converted to camelCase property names

Properties https://www.polymer-project.org/1.0/docs/devguide/properties

Custom CSS properties

```
.title {
        color: var(-- my- too lba r-t itl e-c -
olor);
 }
```

value of the property will inherit down to the toolbar where it is used if defined

https://www.polymer-project.org/1.0/docs/devguide/styling

Declared properties

computed	The value is interpreted as a method name and argument list
observer	The value is interpreted as a method name to be invoked when the property value changes

Configuring default property values

```
Polymer({
   is: 'x-cus tom',
    pro per ties: {
        mode: {
            type: String,
            value: 'auto'
       },
        data: {
            type: Object,
            notify: true,
            value: function() { return {}; }
       }
   }
});
```

Style

:host ::content div	must have a selector to the left of the ::content pseudo-element
--my-toolbar-title-color	These custom properties can be defined and will propogate down the composed DOM tree
color: var(--my--toolbar-title-color, blue)	include a default value in the var() function

Custom CSS mixins

```
// example
<do m-m odule id=" my- too lba r">
    <te mpl ate>
        <st yle>
            :host {
                pad ding: 4px;
                bac kgr oun d-c olor: gray;
           apply/ a mixin /
                @ap ply (-- my- too lba r-t heme);
            }
            .title {
                @ap ply (-- my- too lba r-t itl e-
t heme);
            }
        </s tyl e>
        <span class="title">{{title}}</span>
    </t emp lat e>
   ...
</d om- mod ule>
// example usage
<do m-m odule id=" my- ele men t">
    <te mpl ate>
        <st yle>
```

Custom CSS mixins (cont)

```
           / Apply custom theme to toolbars /
            :host {
                --m y-t ool bar -theme: {
                    bac kgr oun d-c olor: green;
                    bor der -ra dius: 4px;
                    border: 1px solid gray;
                };
                --m y-t ool bar -ti tle -theme: {
                    color: green;
                };
            }
        Make only toolbars with the .warning
class red and bold /
            .wa rning {
                --m y-t ool bar -ti tle -theme: {
                    color: red;
                    fon t-w eight: bold;
                };
            }
        </s tyl e>
        <my -to olbar title= "This one is
green." > </ my- too lba r>
        <my -to olbar title= "This one is green
too.">< /my -to olb ar>
        <my -to olbar class= " war nin g"
title= "This one is red.">< /my -to olb ar>
    </t emp lat e>
    <sc rip t>
        Pol ymer({ is: 'my-el eme nt'});
    </s cri pt>
</d om- mod ule>
```