Polymer 1.x

Uh…. Yeah…. Let’s talk about Polymer, then…

André Valgrande andre.valgrande@gofore.com
Fabiano Brito fabiano.brito@gofore.com

LEADIN / GOFORE

What is Polymer?

A library for creating custom elements that behave like standard DOM elements.

Elements can be:

- Instantiated using a constructor or document.createElement.
- Configured using attributes or properties.
- Populated with internal DOM inside each instance.
- Responsive to property and attribute changes.
- Styled with internal defaults or externally.
- Responsive to methods that manipulate its internal state.

Why Polymer?

This is a very biased presentation based on real-life experiences.

Good!

- Creating custom reusable components.
- Components can be easily packed and distributed.

Not so good!

- Creating deep interaction between components (component-based architecture misuse).
- Creating whole web applications.
- Bower is mandatory.

Polymer components can be mixed with other libraries/frameworks (e.g. Angular, React).
… but should you?

Why… indeed?

	stars	forks	contributors	built with	issues	threads
	18.8k	1.8k	120	<0.1%	601	7.2k
	83.6k	35.1k	2095	9.4%	2172	306k
	73.4k	13.8k	1043	1.8%	580	54k
	63.3k	9k	122	0.3%	89	8.5k

Component declaration

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

<dom-module id="element-name">
  <template>
    <style></style>
    <div>{{ greeting }}</div>
  </template>
  <script>
    Polymer({
      is: "element-name",
        properties: {
          greeting: {
            type: String,
            value: "Hello!"
          }
        }
      });
  </script>
</dom-module>
...
<element-name greeting="What's up?"></element-name>
```

- property typing
- supposed to put everything in the distributable html

Data binding

- Automatic, which allows upward (target to host) and downwards (host to target) data flow. Automatic bindings use double curly brackets ({{ }}):

```
<my-input value="{{name}}"></my-input>
```

- One-way, which only allows downwards data flow. Upward data flow is disabled. One-way bindings use double square brackets ([[ ]]).

```
<name-tag name="[[name]]"></name-tag>
```

Scoped styling 1.x

```
<dom-module id="my-toolbar">
  <template>
    <style>
      :host {
        padding: 4px;
        background-color: gray;
      }
      .title {
        color: var(--my-toolbar-title-color, black);
      }
    </style>
    <span class="title">{{title}}</span>
  </template>
  <script>
    Polymer({
      is: 'my-toolbar',
      properties: {
        title: String
      }
    });
  </script>
</dom-module>
```

Styling the host element and passing a style variable

```
<dom-module id="my-element">
  <template>
    <style>
      :host {
        --my-toolbar-title-color: green;
      }
      .warning {
        --my-toolbar-title-color: red;
      }
    </style>

    <my-toolbar title="This one is green."></my-toolbar>
    <my-toolbar title="This one is green too."></my-toolbar>
    <my-toolbar class="warning" title="This one is red."></my-toolbar>

  </template>
  <script>Polymer({ is: 'my-element'});</script>
</dom-module>
```

Helper elements

- dom-repeat

```
<template is="dom-repeat" items="{{employees}}">
  <div># <span>{{index}}</span></div>
  <div>First name: <span>{{item.first}}</span></div>
  <div>Last name: <span>{{item.last}}</span></div>
</template>
```

- dom-if

```
<template is="dom-if" if="{{user.isAdmin}}">
  Only admins will see this.
  <div>{{user.secretAdminStuff}}</div>
</template>
```

Life-cycle methods

```
MyElement = Polymer({

  is: 'my-element',

  created: function() {
    console.log(this.localName + '#' + this.id + ' was created');
  },
  ready: function() {
    console.log(this.localName + '#' + this.id + ' has local DOM initialized');
  },
  attached: function() {
    console.log(this.localName + '#' + this.id + ' was attached');
  },
  detached: function() {
    console.log(this.localName + '#' + this.id + ' was detached');
  },
  attributeChanged: function(name, type) {
    console.log(this.localName + '#' + this.id + ' attribute ' + name + ' was changed to ' + this.getAttribute(name));
  }
});
```

Observers

Simple

```
properties: {
  disabled: {
    type: Boolean,
    observer: '_disabledChanged'
  },
  highlight: {
    type: Boolean,
    observer: '_highlightChanged'
  }
},
_disabledChanged: function(newValue, oldValue) {
  this.toggleClass('disabled', newValue);
  this.set('highlight', true);
},
_highlightChanged: function() {
  this.classList.add('highlight');
  this.async(function() {
    this.classList.remove('highlight');
  }, 300);
}
```

observable change!

Complex

Deep sub-property

```
observers: [
  'userNameChanged(user.name.*)'
],
userNameChanged:
function(changeRecord) {
  console.log('path: ' + changeRecord.path);
  console.log('value: ' +
changeRecord.value);
}
```

Multiple attributes

```
observers: [
  'updateImage(preload, src, size)'
],
updateImage: function(preload, src, size) {
  // ... do work using dependent values
}
```

Computed properties

Virtual properties with values calculated from other properties.

```
Polymer({
  is: 'x-custom',
  properties: {
    first: String,
    last: String,
    fullName: {
      type: String,
      computed: 'computeFullName(first, last)'
    }
  },
  computeFullName: function(first, last) {
    return first + ' ' + last;
  }
});
```

Behaviors

highlight-behavior.html

```
<script>
  HighlightBehavior = {
    properties: {
      isHighlighted: {
        type: Boolean,
        value: false,
        notify: true,
        observer: '_highlightChanged'
      }
    },
    listeners: {
      click: '_toggleHighlight'
    },
    _toggleHighlight: function() {
      this.isHighlighted = !this.isHighlighted;
    },
    _highlightChanged: function(value) {
      this.toggleClass('highlighted', value);
    }
  };
</script>
```

my-element.html

```
<link rel="import" href="highlight-behavior.html">

<script>
  Polymer({
    is: 'my-element',
    behaviors: [HighlightBehavior]
  });
</script>
```

Conclusion

Biased presentation based on our, so far, weird experience with Polymer.

- Despite the interesting concept, doesn't seem to be worth the hassle.
- Restricted support from the community.
- Restricted modules and components, found several dependency issues.
- At this time, we wouldn't choose Polymer for a new project.

Questions? Be careful...

More info

https://www.polymer-project.org/

https://www.webcomponents.org/

https://www.youtube.com/playlist?list=PLOU2XLYxmsII5c3Mgw6fNYCzaWrsM3sMN

https://github.com/Polymer/polymer/wiki/Who%27s-using-Polymer%3F

https://www.youtube.com/playlist?list=PLNYkxOF6rcICc687SxHQRuo9TVNOJelSZ