International
JavaScript
Conference

How to Re-Architect a JavaScript Class System

Olga Petrova, @tyoushe
Developer Advocate, Sencha

Evolution of Standards and Technologies

Olga Petrova, @tyoushe

Sencha

React
JavaScript library

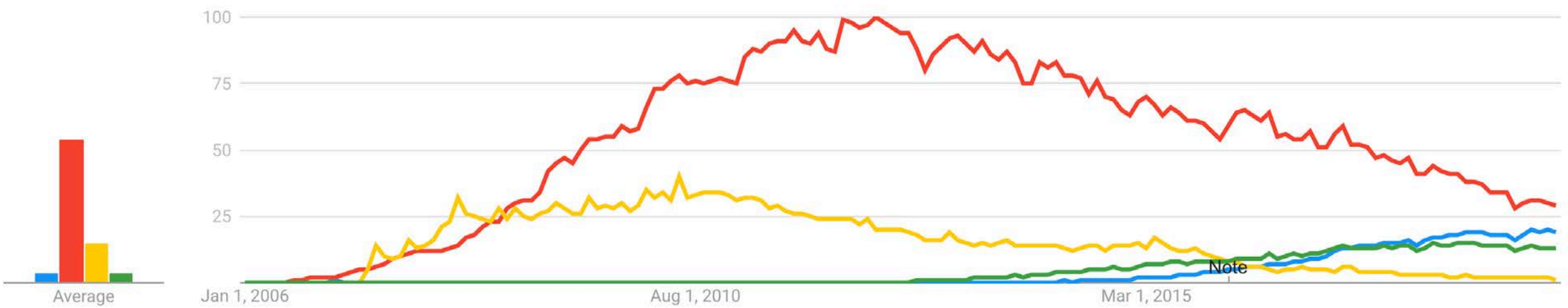
jQuery
Software

Microsoft Silverlight
Web browser extension

Angular
Web framework

Worldwide | 1/1/06 - 5/2/19 | Computers & Electronics | Web Search

Interest over time



Grunt
Topic

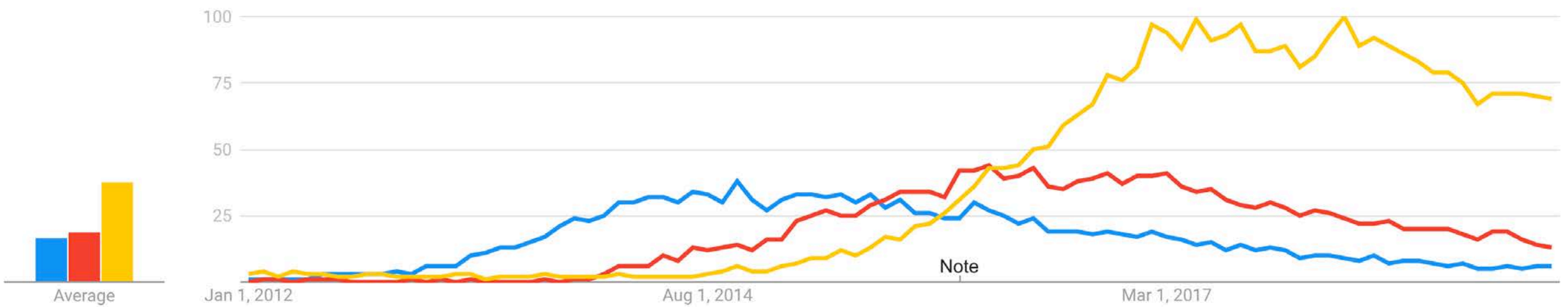
gulp.js
Topic

Webpack
Topic

+ Add comparison

Worldwide | 1/1/12 - 5/6/19 | Computers & Electronics | Web Search

Interest over time



What Makes Technology Future-Proof?

Olga Petrova, @tyoushe

Sencha

Inheritance via Prototyping

```
function MyComponent() {}
MyComponent.prototype.init = function() {
  //...
}

function MyChildComponent() {}
MyChildComponent.prototype = Object.create(new MyComponent());
MyChildComponent.prototype.destroy = function() {
  //...
}

var cmp = new MyChildComponent();
cmp.init();
cmp.destroy();
```

Ext JS

Olga Petrova, @tyoushe

Sencha

Ext JS Class System

- ✓ Inheritance
- ✓ Config properties
- ✓ Dependencies
- ✓ Mixins
- ✓ Overrides
- ✓ Build tool

Ext JS Class

```
Ext.define('My.own.Component', {
    extend: 'Ext.Component',
    requires: ['My.Component.Dependency'],

    myProperty: true,

    myMethod: function() {
        //...
        this.callParent(arguments);
    },

    statics: {
        staticMethod: function() {
            //...
        }
    }
});
```

Ext JS Config Properties

```
Ext.define('My.own.Component', {
  extend: 'Ext.Component‘,

  config: {
      myConfig: 'My Text'
  },

  applyMyConfig: function(newTitle, oldTitle) {
      //implement validation
      return newTitle;
  },

  updateMyConfig: function(newTitle, oldTitle) {
      //implement side effects
  }
});
```

Ext JS Overrides

```
var MyComponent = new Ext.Component({});

Ext.override(MyComponent, {

    myProperty: 'My Text',

    myMethod: function () {
        //...
    }

});
```

Ext JS Mixins

```
Ext.define('Mixin', {

    processData: function(data) { /**/ }

});

Ext.define('MyComponent', {

    mixins: {
        myMixin: 'Mixin'
    },

    processData: function(data) {
        //...
        this.mixins.myMixin.processData.call(this);
    }
});
```

10K	2M	7.2M	500K
CUSTOMERS WORLDWIDE	SENCHA DEVS WORLDWIDE	PRODUCT DOWNLOADS	ACTIVE FORUM MEMBERS

60% of Fortune 100 Companies Rely on Sencha

ECMAScript6 Class system

Olga Petrova, @tyoushe

Sencha


CUPERTINO

ECMAScript6 Class Systems

- ✓ Inheritance
- ✓ Dependencies
- ✓ Build tool
- ✗ Config properties
- ✗ Mixins
- ✗ Overrides


SOON

Decorator pattern is a design pattern that allows behavior to be added to an individual object, dynamically, without affecting the behavior of other objects from the same class

JavaScript Decorators

Olga Petrova, @tyoushe

Sencha

@wrap Method

```
class C {
  @wrap(f) method() { }
}
```

=>

```
class C {
  method() { }
}
C.prototype.method = f(C.prototype.method);
```

@wrap Class

```
@wrap(f)
class C { }
```

=>

```
class C { }
C = f(C);
```

JavaScript Class Fields

Olga Petrova, @tyoushe

Sencha

Class Fields

```
class MyComponent {

  myField = 42;

}
```

ECMAScript 6 to Ext JS Class Systems

- ✓ Inheritance
- ✓ Dependencies
- ✓ Build tool
- ✓ Config properties @define for class & @config for class fields
- ✓ Mixins @define for class
- ✓ Overrides @override for class

Class

```
@define({
  prototype: {
    myProperty: "My text"
  },

  static: {
    myStaticProperty: 42
  }
})

class MyOwnComponent extends Component {
//
}
```

Configs

```
@define
class MyOwnComponent extends Component {

  @config('nullify')
  myConfig = 42;

  applyMyConfig (value, oldValue) {
      //validation
      return value;
  }

  updateMyConfig (value, oldValue) {
      //side effects
  }
}
```

Mixins

```
@define
class Mixin extends Base {
   processData (data) {...}
}

@define({
   mixins: [ Mixin ]
})
class MyComponent extends Component {

   @junction
   processData (data) {
       super.processData(data);
   }
}
```

Overrides

```
class MyComponent extends Component {}

@override(MyComponent)
class MyComponentOverride {

   myProperty = 'My Text';

   myMethod () {
      //...
   }
}
```

Life-cycle Methods

```
class MyComponent extends Component {
    ctor () {
        // do constructor-like things
    }

    setup() {
        // finalize configuration
    }

    dtor () {
        // do destructor-like things
    }
}
```

Lessons Learned

Follow Standards if You Can

Olga Petrova, @tyoushe

Sencha

Estimate Effort on Re-write

Encapsulate

Keep your Eyes on New Standards

Olga Petrova, @tyoushe

Sencha

Start to Adapt Earlier

Keep Decorator/Adapter/Facade Patterns in Mind

Olga Petrova, @tyoushe

Sencha