Web & Mobile Graphics in Niagara 4

Logan Byam and Tony Richards

Software Engineers @ Tridium

TRIDIUM

Niagara Web Technologies

More options than ever

JavaScript

But do I *need* this JavaScript stuff?

Maybe not...

Servlets

```
1  public void doGet(HttpServletRequest req, HttpServletResponse res)
2    throws IOException
3  {
4    res.getWriter().write(
5      "<html><body>" +
6      "  Hello, Niagara Summit!" +
7      "</body></html>");
8  }
```

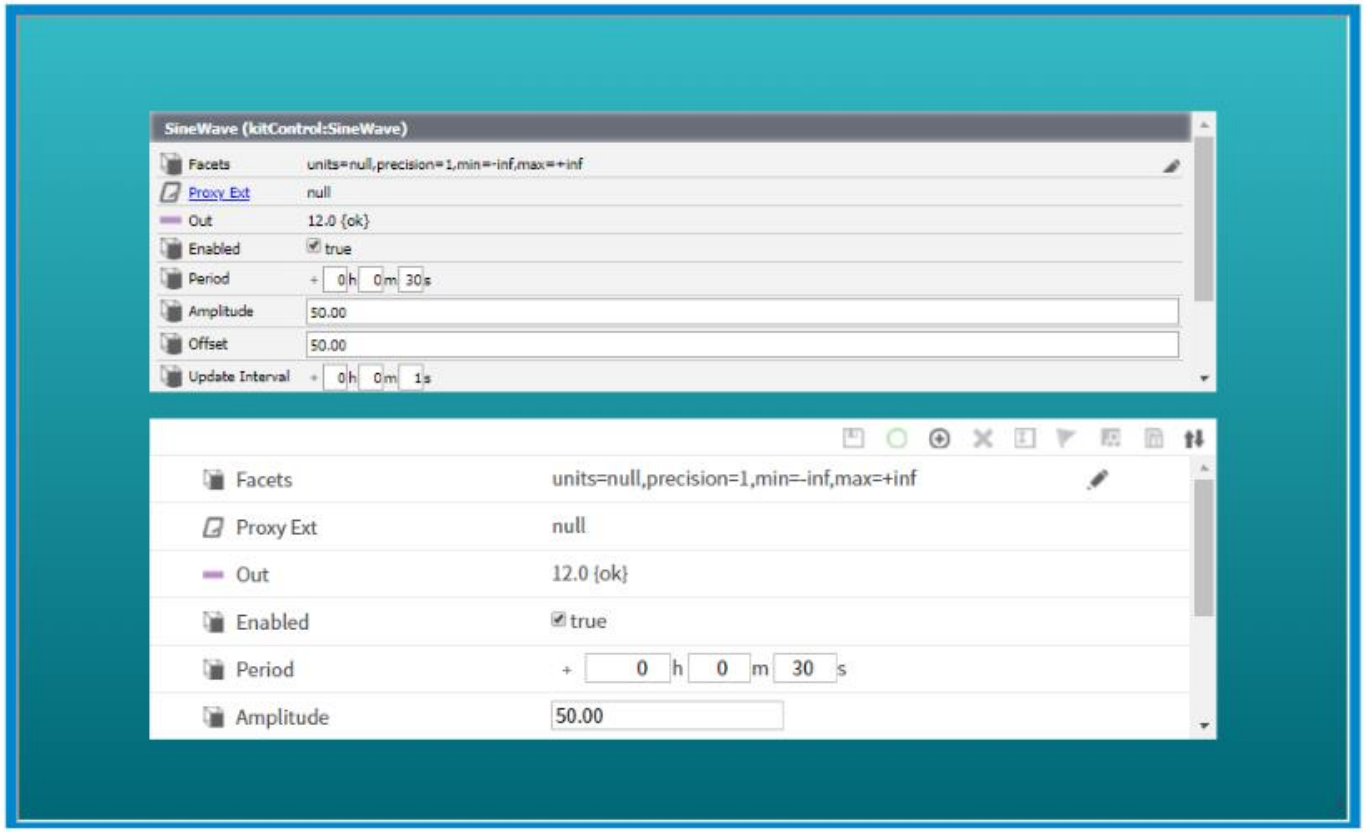
BServletView

```
1  public void doGet(WebOp webOp)
2    throws IOException
3  {
4    webOp.getWriter().write(
5      "<html><body>" +
6      "  You're looking at a " + webOp.get().getType() + "." +
7      "</body></html>");
8  }
```

Apache Velocity

```
1  <html><body>
2    You're looking at a: $util.get().getType().
3  </body></html>
```

HxPx


SineWave (kitControl:SineWave)
Facets
units=null,precision=1,min=-inf,max=+inf
Proxy Ext
null
Out
12.0 {ok}
Enabled
true
Period
0 h 0 m 30 s
Amplitude
50.00
Offset
50.00
Update Interval
0 h 0 m 1 s
Facets
units=null,precision=1,min=-inf,max=+inf
Proxy Ext
null
Out
12.0 {ok}
Enabled
true
Period
0 h 0 m 30 s
Amplitude
50.00

Why JavaScript?

- Single page apps
- Real time response
- Build on existing components

BajaScript

is an extensible JavaScript data API

that interacts with a Niagara station.

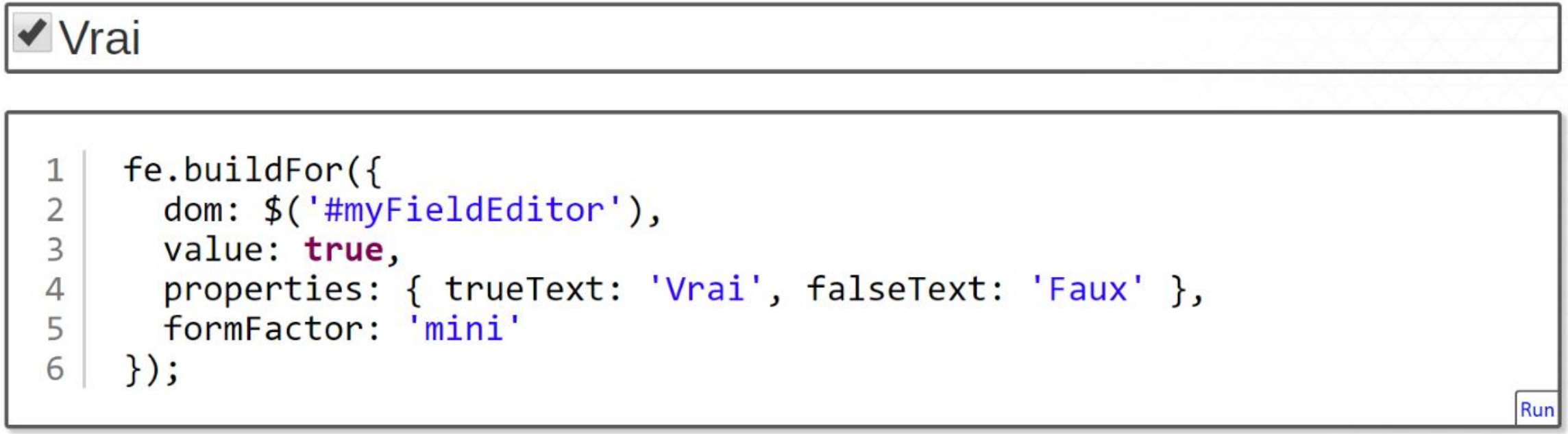
```
1  var sub = new baja.Subscriber();
2  sub.attach('changed', function (prop) {
3    if (prop.getName() === 'out') {
4      console.log('Out value is: %c%s', 'color: green', this.get(prop));
5    }
6  });
7
8  baja.Ord.make('station:|slot:/SineWave').get({ subscriber: sub });
```

Run

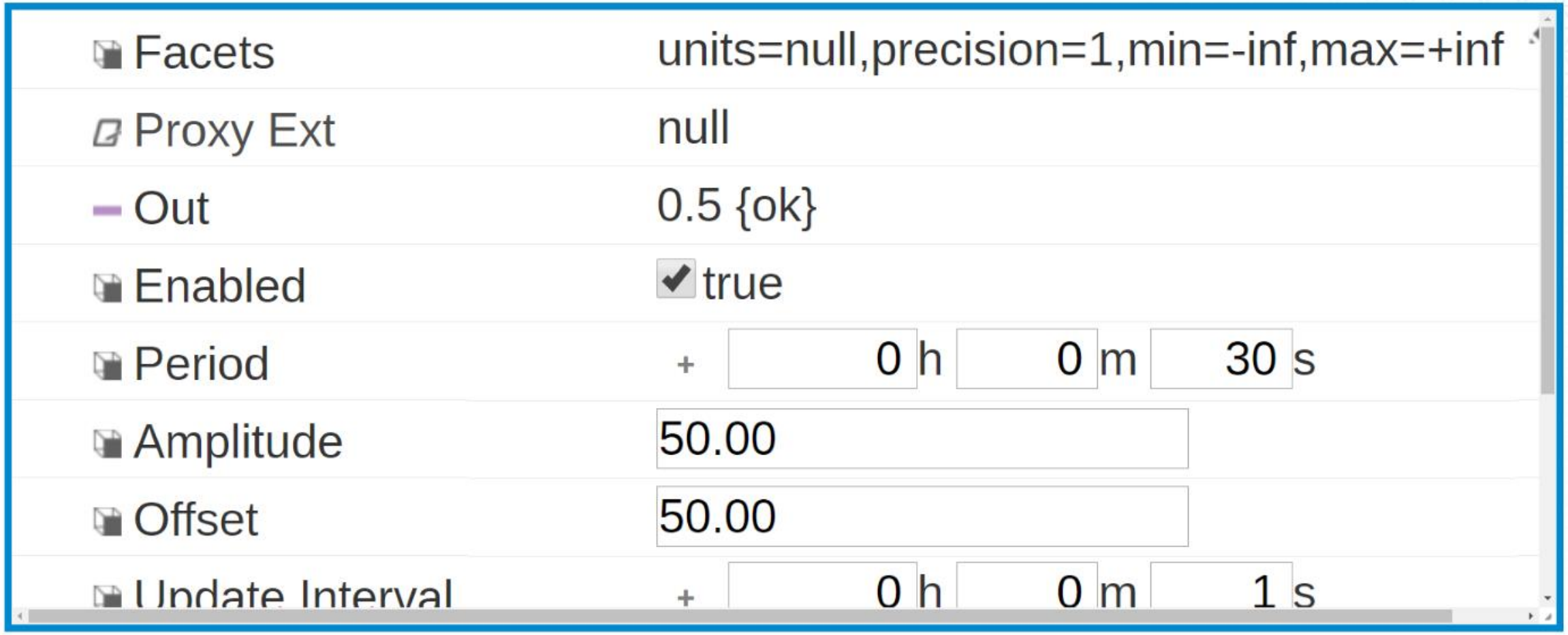
bajaux

is an HTML5 widget framework

that integrates with Niagara.



bajaux Views



Open-Source software

Out value is 27.6 {ok}!

```
1  require(['react', 'reactDom'], function (React, ReactDOM) {
2    var sub = new baja.Subscriber();
3    sub.attach('changed', function (prop) {
4      ReactDOM.render(
5        <span>Out value is { this.getDisplay('out') }!</span>,
6        document.getElementById('myReactComponent')
7      );
8    });
9    baja.Ord.make('station:|slot:/SineWave').get({ subscriber: sub });
10 });
```

Run

Getting Started With JavaScript

grunt-init-niagara

Create a brand new web module with smart defaults.

- Q&A-style configuration
- **bajaux** views
- Niagara integration
- Agent registrations

https://github.com/tridium/grunt-init-niagara

grunt-niagara

- Linting
- Testing
- Documentation
- RequireJS Optimization

v2.0 coming soon

https://github.com/tridium/grunt-niagara

Documentation

I know.

It's actually pretty decent:

Help -> Doc Developer -> Developer Guide -> User Interface

Help -> Doc Developer -> Development Tools

Automated Testing

Why is automated testing important?

- Ensure your code works
- Ensure it's protected from changing dependencies
- Ensure you can safely improve it
- Yes, you need to do it
- For real
- But you've got everything you need

```java
@Test
public class BMyClassTest extends BTestNg {

  @Test
  public void myFooMethodShouldReturnBar() {
    assertEquals(BMyClass.foo(), "bar");
  }
}
```

```
gradlew jar testModule -a -t
```

```
test.exe -watch myModule:MyTestModule
```

```
grunt watch --testOnly=myModule
```

```
describe('myModule', function () {
  describe('.foo()', function () {
    it('does something', function () {
      expect(myModule.foo()).toBe('bar');
    });
  });
});
```

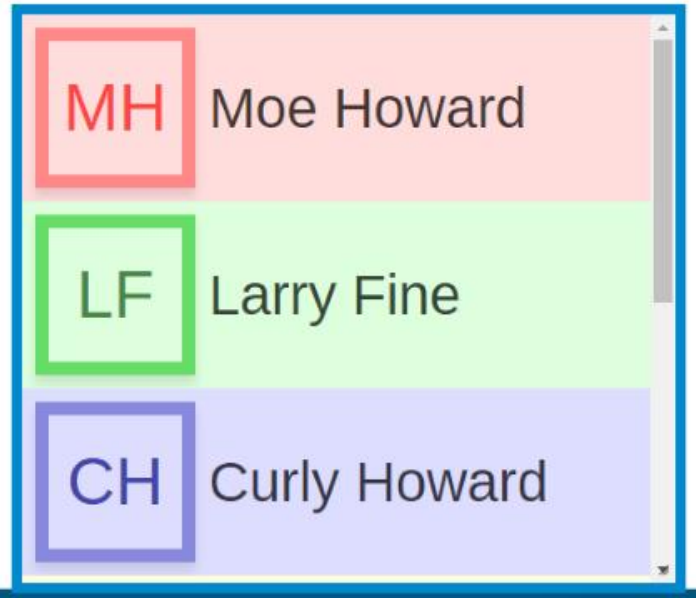
Run

Responsive Web Design

Tools and Enhancements in 4.6

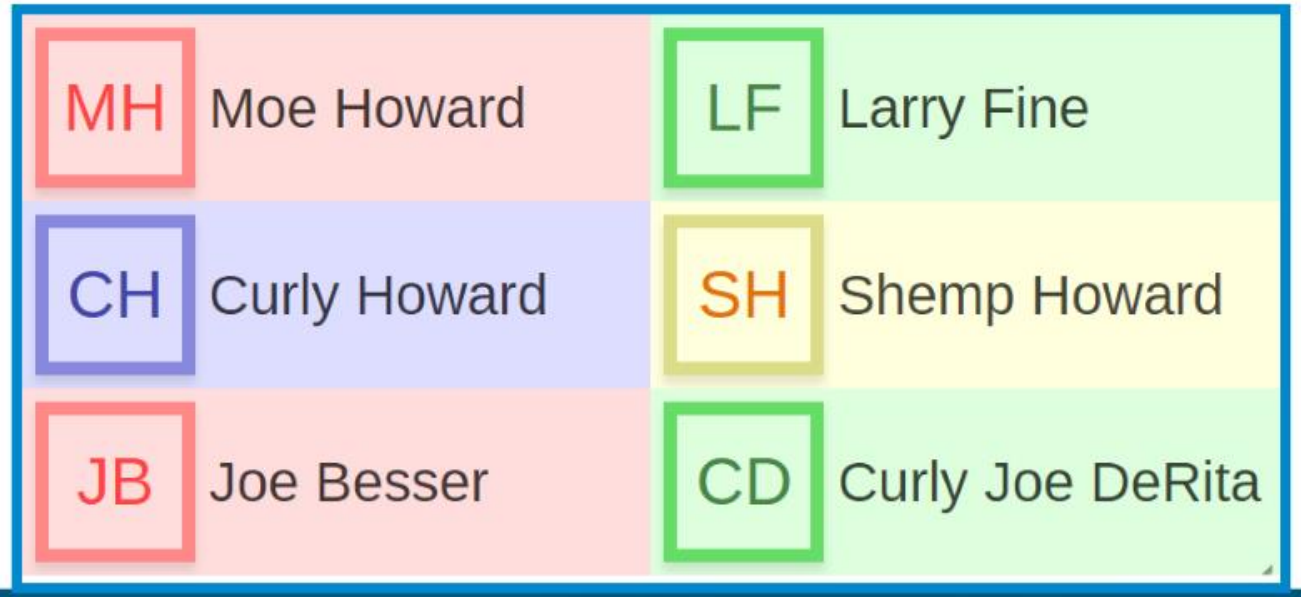
bajaux/mixin/responsiveMixIn

- Widgets lay out in response to their container size
- Browsers, Workbench, and Px
- Works where media queries don't



bajaux/mixin/responsiveMixIn

- Widgets lay out in response to their container size
- Browsers, Workbench, and Px
- Works where media queries don't



bajaux/mixin/responsiveMixIn

- Widgets lay out in response to their container size
- Browsers, Workbench, and Px
- Works where media queries don't

```
responsiveMixIn(this, {
  'tablet-portrait-up': { minWidth: 700 },
  'desktop-up': { minWidth: 1400 }
});
```

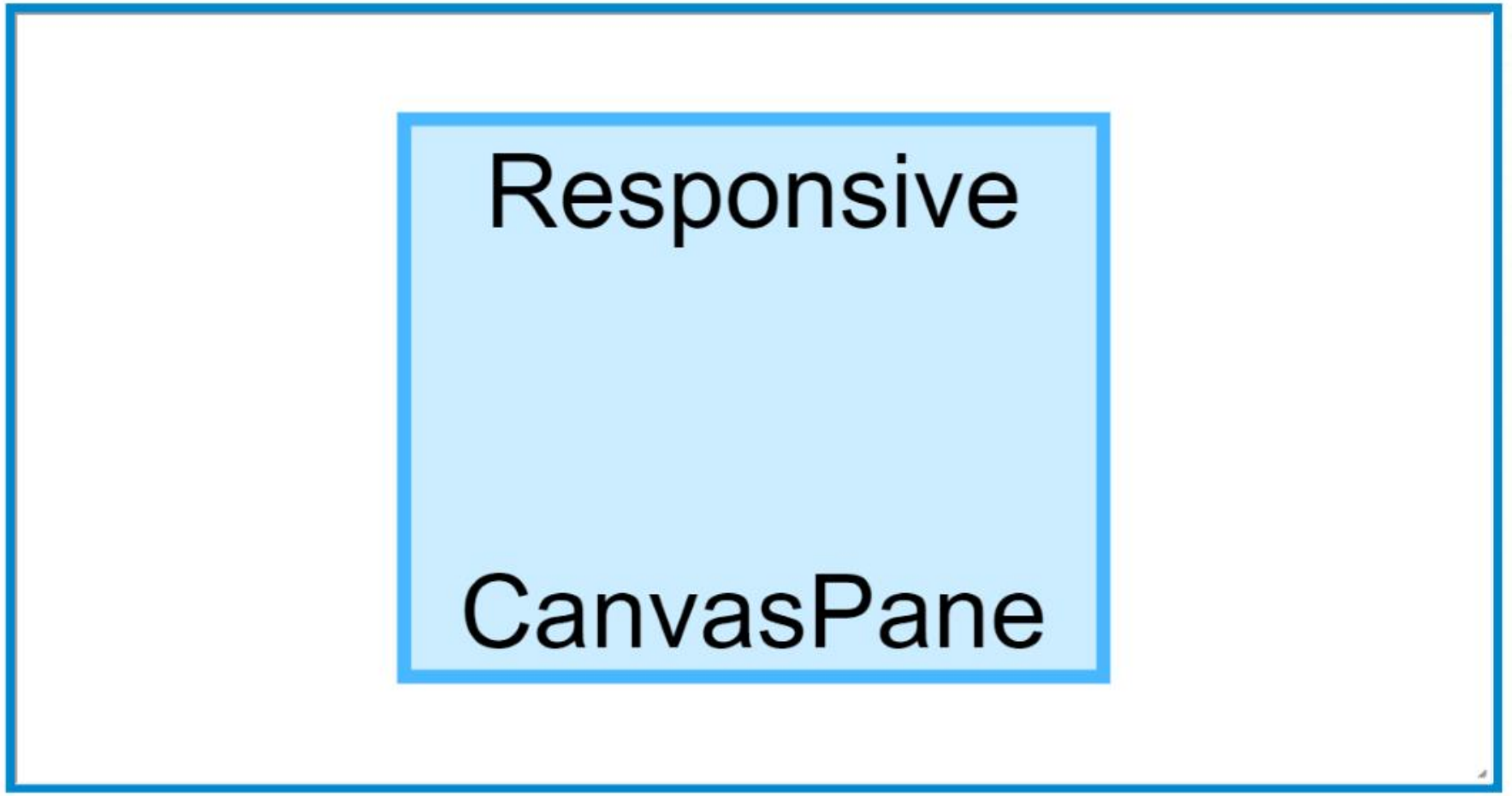
Run

```
.ResponsiveWidget .info-cell {
  flex-basis: 100%;
}
.ResponsiveWidget.tablet-portrait-up .info-cell {
  flex-basis: 50%;
}
.ResponsiveWidget.desktop-up .info-cell {
  flex-basis: 33.3%;
}
```

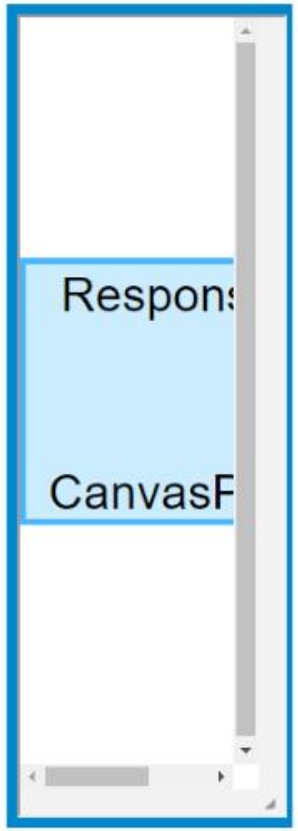
Responsive Px

Enhancements in Niagara 4.6

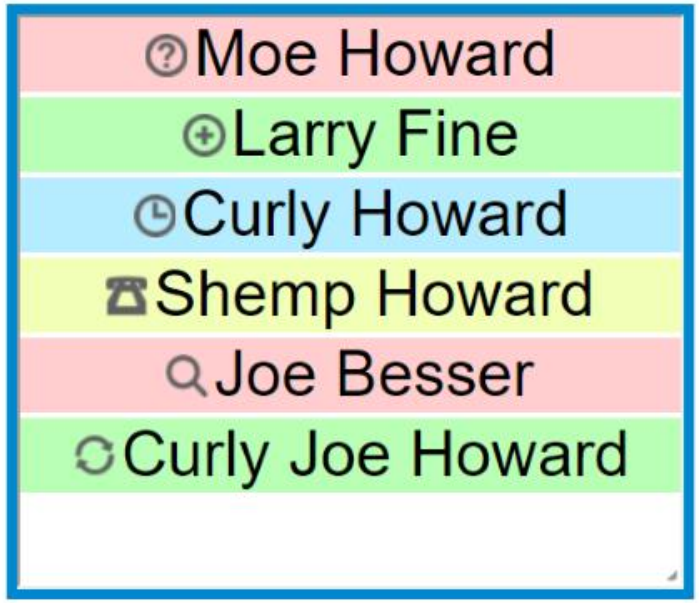
CanvasPane scaling



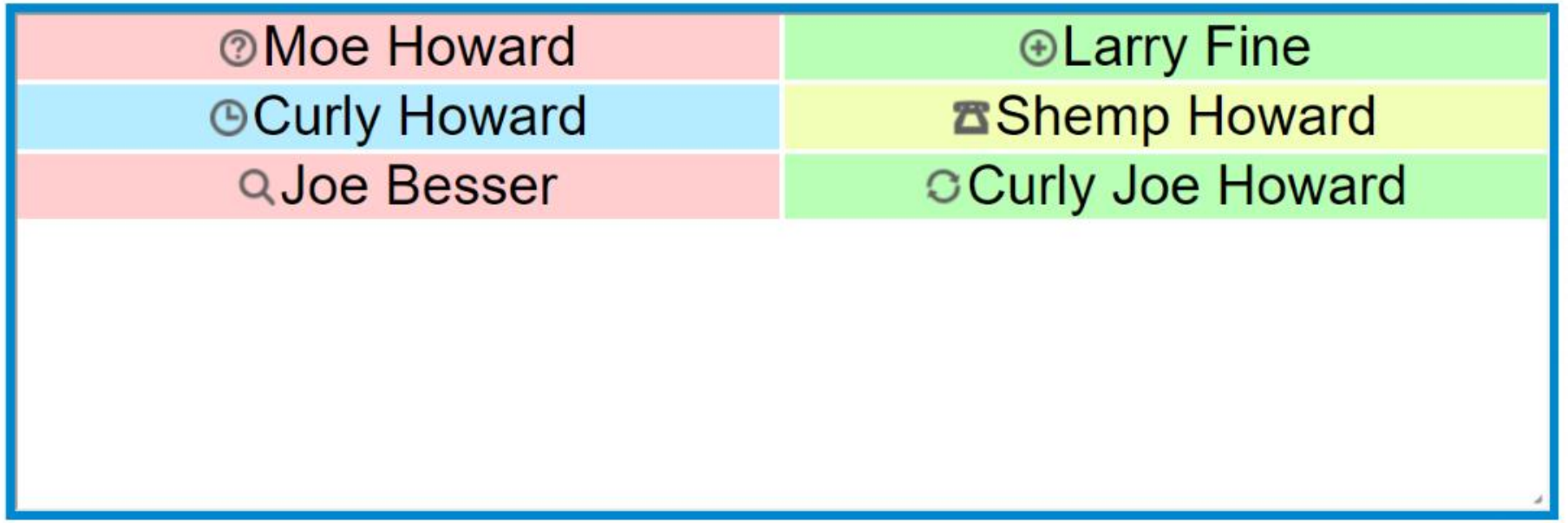
CanvasPane scaling



FlowPane and ResponsivePane



FlowPane and ResponsivePane



Abstract Manager Framework

Making Managers easy to create, simple to use

Why an Abstract Manager Framework?

- Easy to create user-friendly views
- Manage large sets of data and points
- Consistent, simple user experience
- Minimal coding

Why an Abstract Manager Framework?

- Need rows and columns of Data
- Live data points in a Station

Abstract Manager Framework

- What are Managers?
- Why would I want to develop a Manager?
- How do I develop a manager?
- One Caveat
- Further Reading

What are Managers?

Examples:

- Bacnet Device Manager
- User Manager
- Email Account Manager

Why Would I Want to Develop a Manager?

Driver Developer and other Application Developer

Collection of "things" Need to be managed by the user May be converted into Niagara Points

Components of a Manager

- Manager
- Data Model
- Learn Model (optional)

Diving Deeper

- **Manager**
- Data Model
- Learn Model

Manager Lifecycle Events

- `constructor()`
- `makeModel()`

Manager Constructor

```
var MyManager = function MyManager (params) {
  Manager.call(this, {
    moduleName: 'myModule',
    keyName: 'MyUxManager' // Typically the name of the type
  });
};
```

Run

Manager makeModel()

```
1  MyManager.prototype.makeModel = function (component) {
2    return MyManagerModel.make(component);
3  };
```

Run

Diving Deeper

- Manager implementation
- **Data Model implementation**
- Learn Model implementation

Data Model: Extend MgrModel

```
1  define(['nmodule/webEditors/rc/wb/mgr/model/MgrModel', function (
2            MgrModel) {
3    var MyMgrModel = function MyMgrModel() {
4      MgrModel.apply(this, arguments);
5    };
6    MyMgrModel.prototype = Object.create(MgrModel.prototype);
7    MyMgrModel.prototype.constructor = MyMgrModel;
8  });
```

Run

Data Model

```
1  MyManagerModel.make = function (component) {
2    return MgrTypeInfo.make([
3      'control:BooleanWritable', 'control:NumericWritable'
4    ])
5    .then(function (newTypes) {
6      return new MyManagerModel({
7        columns: makeColumns(),
8        componentSource: component,
9        newTypes: newTypes
10     });
11   });
12 };
```

Run

MgrColumn

- NameMgrColumn
- PropertyMgrColumn
- PathMgrColumn
- TypeMgrColumn
- IconMgrColumn
- PropertyPathMgrColumn

MgrColumn methods

- `constructor`
- `getValueFor(row)`
- `buildCell(row, dom)`

MgrColumn constructor

```
var MyManagerColumn = function (name) {
  Column.apply(this, [name, {
    displayName: lex.get({
    key: 'mymodule.' + name,
    def: textUtils.toFriendly(name)
    })
  }]);
  this.setHidable(false);
};
```

Run

MgrColumn getValueFor

```
1  MyManagerColumn.prototype.getValueFor = function (row) {
2    var component = row.getSubject();
3    return getSomeValueFromTheSubject(component);
4  };
```

Run

MgrColumn buildCell

```
1  MyManagerColumn.prototype.buildCell = function (row, dom) {
2    var value = this.getValueFor(row),
3      displayText = getDisplayTextForValue(value);
4
5    dom.text(displayText);
6  };
```

Run

makeColumns

```
1  function makeColumns() {
2    return [
3      new NameMgrColumn({ flags: Column.flags.EDITABLE }),
4      new PropertyMgrColumn('fullName', { flags: Column.flags.EDITABLE }),
5      new PropertyMgrColumn('email', { flags: Column.flags.EDITABLE })
6      new MyManagerColumn('column1'),
7      new MyManagerColumn('column2')
8    ];
9  }
```

Run

Example Manager

Name	Type	Exts	Status
MIDIIN2 (2- Launchpad Pro)Tx	Midi Device		{ok}
PC Keyboard	Pc Keyboard Midi Device		{ok}

New Edit

Diving Deeper

- Manager implementation
- Data Model implementation
- **Learn Model implementation**

Discovery Process

Discovery Job -> Learn Model -> Add -> Data Model

Manager discovery support

- `makeLearnModel()`
- `doDiscover()`
- `getTypesForDiscoverySubject()`
- `getProposedValuesFromDiscovery()`

and optionally

- `isExisting()`

MgrLearn mixin

```
1  define([...
2    'nmodule/webEditors/rc/wb/mgr/MgrLearn'], function (
3    ...,
4    MgrLearn) {
5
6
7  var MyManager = function MyManager() {
8    Manager.call(...);
9    MgrLearn(this);
10 };
11 ...
```

Run

Learn Model

- Is or derives from **`TreeTableModel`**
- Requires columns and root node.

Manager makeLearnModel()

```
1  MyManager.prototype.makeLearnModel = function () {
2    return MyLearnModel.make();
3  };
```

Run

and...

```
1  MyLearnModel.make = function () {
2    return TreeTableModel.make({
3      columns: createColumns(),  // return an array of Columns
4      node: createRootNode()     // return a TreeNode instance for the roo
5    });
6  };
```

Run

doDiscover()

```
1  MyManager.prototype.doDiscover = function () {
2    var that = this,
3      pointExt = this.getPointExtension();
4
5    return that.showDiscoveryConfigurationDialog()
6      .then(function (config) {
7        return pointExt.invoke({
8          slot: 'discoverPoints', value: config
9        });
10     })
11     .then(function (ord) {
12       return that.setJob(ord);
13     });
14 };
```

Run

Add jobcomplete handler

```
1  var MyManager = function MyManager (params) {
2    var that = this;
3    Manager.call(...);
4
5    // Add jobcomplete event handler
6    that.on('jobcomplete', function (job) {
7      that.updateLearnTable(job)
8        .catch(baja.error);
9    });
10 };
```

Run

Discovery Job completion handler

```
1  MyManager.prototype.updateLearnTable = function(job) {
2    var that = this;
3
4    return that.resolveLeafChildren(job)
5      .then(function (leafChildren) {
6        var treeNodes = leafChildren.map(leafToTreeNode),
7            learnModel = that.getLearnModel();
8
9        return overwriteAllRows(learnModel, nodes);
10     });
11 };
```

Run

Example Manager With Discovery

N Discovery — Success

Discovered — 10 objects

Dev Name	Vendor	Description	Version	Max Receivers
Gervill	OpenJDK	Software MIDI Synthesizer	1.0	-1
2- Launchpad Pro	Unknown vendor	No details available	10.0	0

Database — 2 objects

Name	Type	Exts	Status
MIDIIN2 (2- Launchpad Pro)Tx	Midi Device		{ok}

New | Edit | Discover | Cancel | Add | Match

One Caveat

Note: this API is still in incubation and may change in future versions of Niagara.

Please, give us feedback!

Further Reading

In the Niagara Developer Documentation, there's a tutorial that covers this topic in more details than what we can cover here.

That tutorial includes more details about the Manager Model and integrating a Point Manager, Device Manager, etc. with your new Manager.