# JavaScript in APEX

-

# Tips & tricks

# Let’s warm up

```
> '5' - 3
2

> '5' + 3
'53'

> '5' - '4'
1
```

```
> [] + []
""

> {} + []
0

> [] + {}
"[object Object]"

> {} + {}
NaN
```

## MENNO HOOGENDIJK

APEX Consultant

ORACLE ACE

mennooo

mennooo

menn.ooo

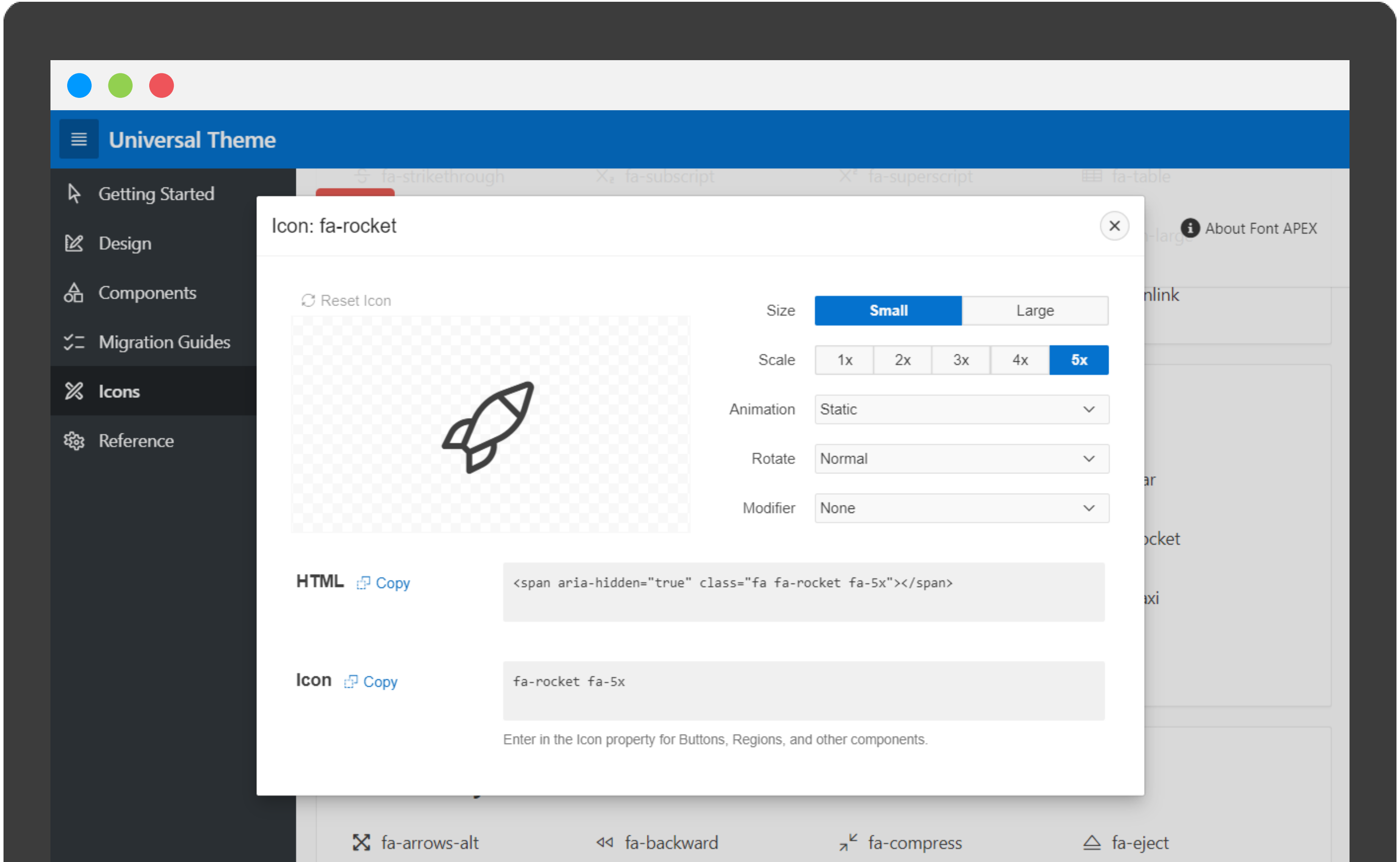
over 160

... colleagues work in the Netherlands

over 40

... colleagues work in the Caribbean

# Oracle APEX 18.2



Icon: fa-rocket

Size: Small

Scale: 5x

Animation: Static

Rotate: Normal

Modifier: None

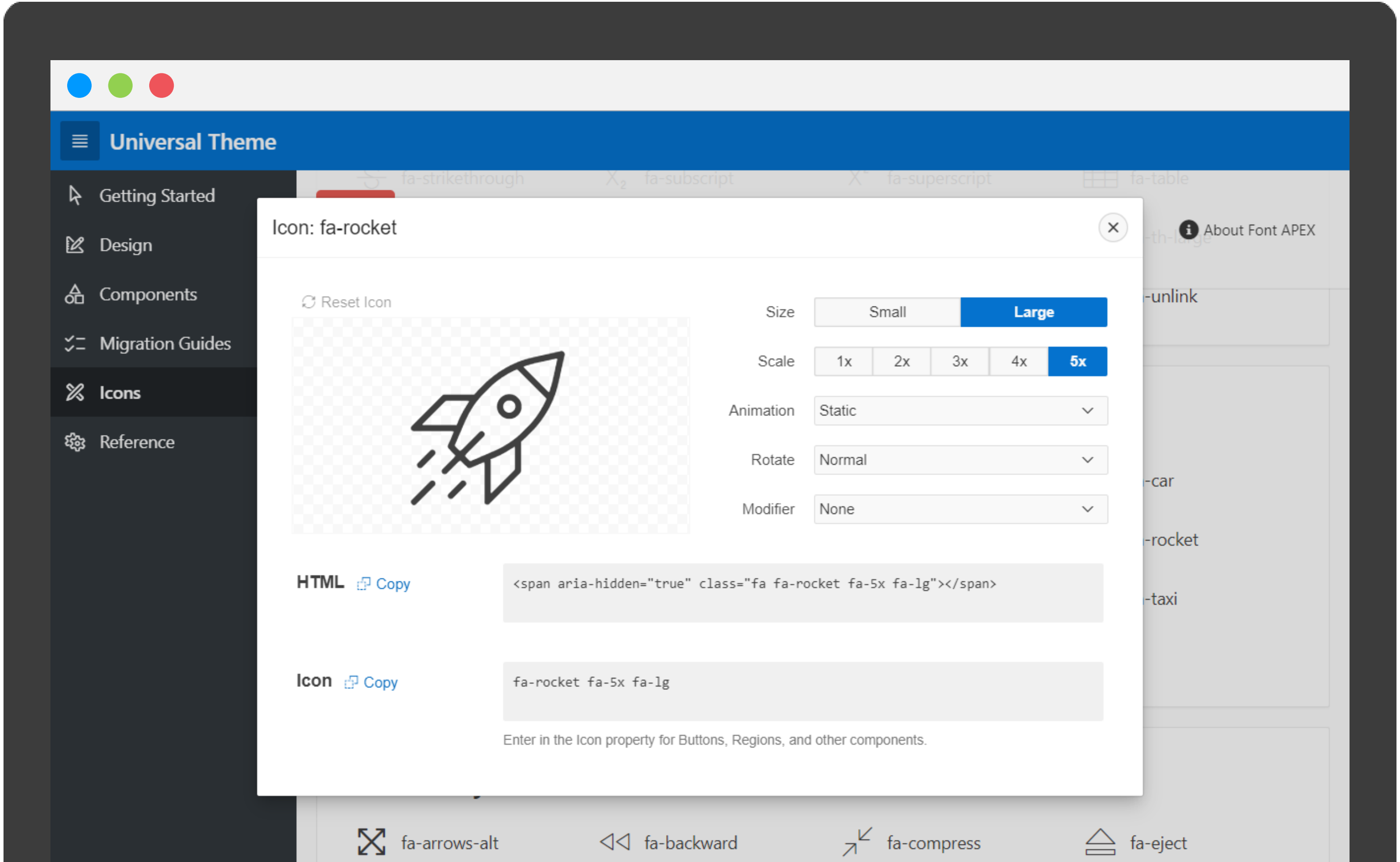
HTML

```
<span aria-hidden="true" class="fa fa-rocket fa-5x"></span>
```

Icon

```
fa-rocket fa-5x
```

Enter in the Icon property for Buttons, Regions, and other components.



Icon: fa-rocket

Size: Large

Scale: 5x

Animation: Static

Rotate: Normal

Modifier: None

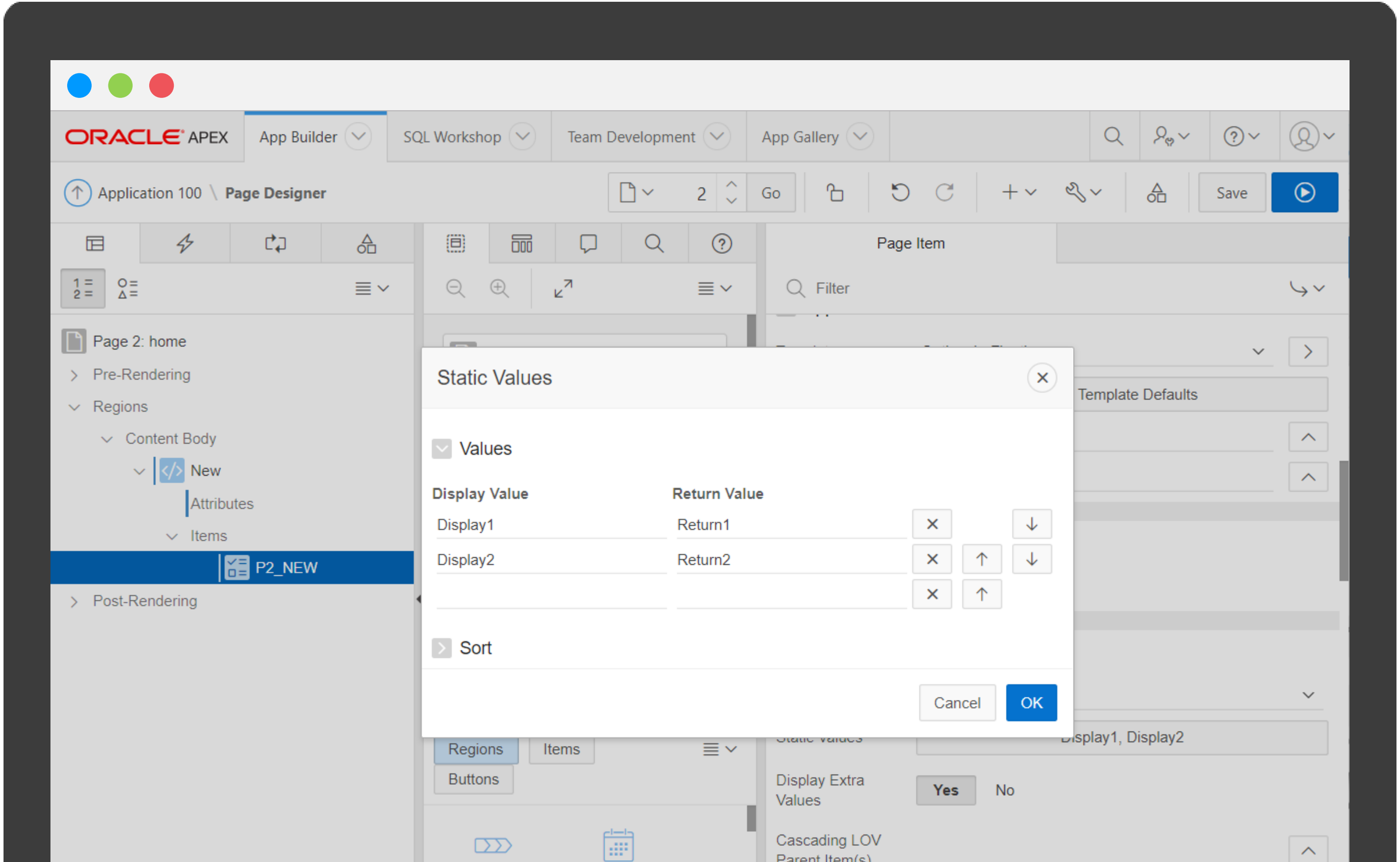
HTML

```
<span aria-hidden="true" class="fa fa-rocket fa-5x fa-lg"></span>
```

Icon

```
fa-rocket fa-5x fa-lg
```

Enter in the Icon property for Buttons, Regions, and other components.



Static Values

Values

| Display Value | Return Value |
|---|---|
| Display1 | Return1 |
| Display2 | Return2 |
| | |

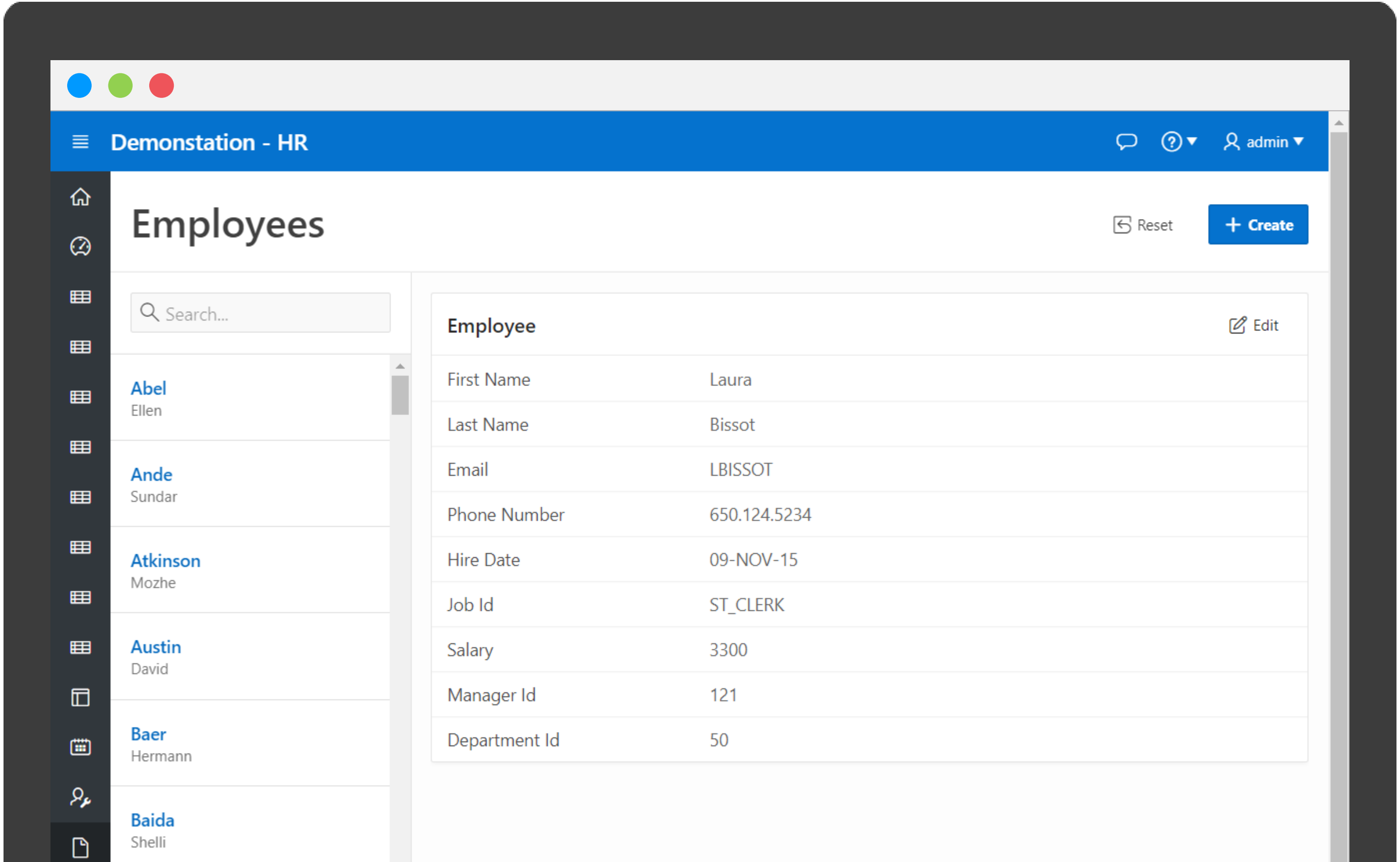
Sort

Cancel OK

Static Values: Display1, Display2

Display Extra Values: Yes No

Cascading LOV Parent Item(s)



Abel
Ellen

Ande
Sundar

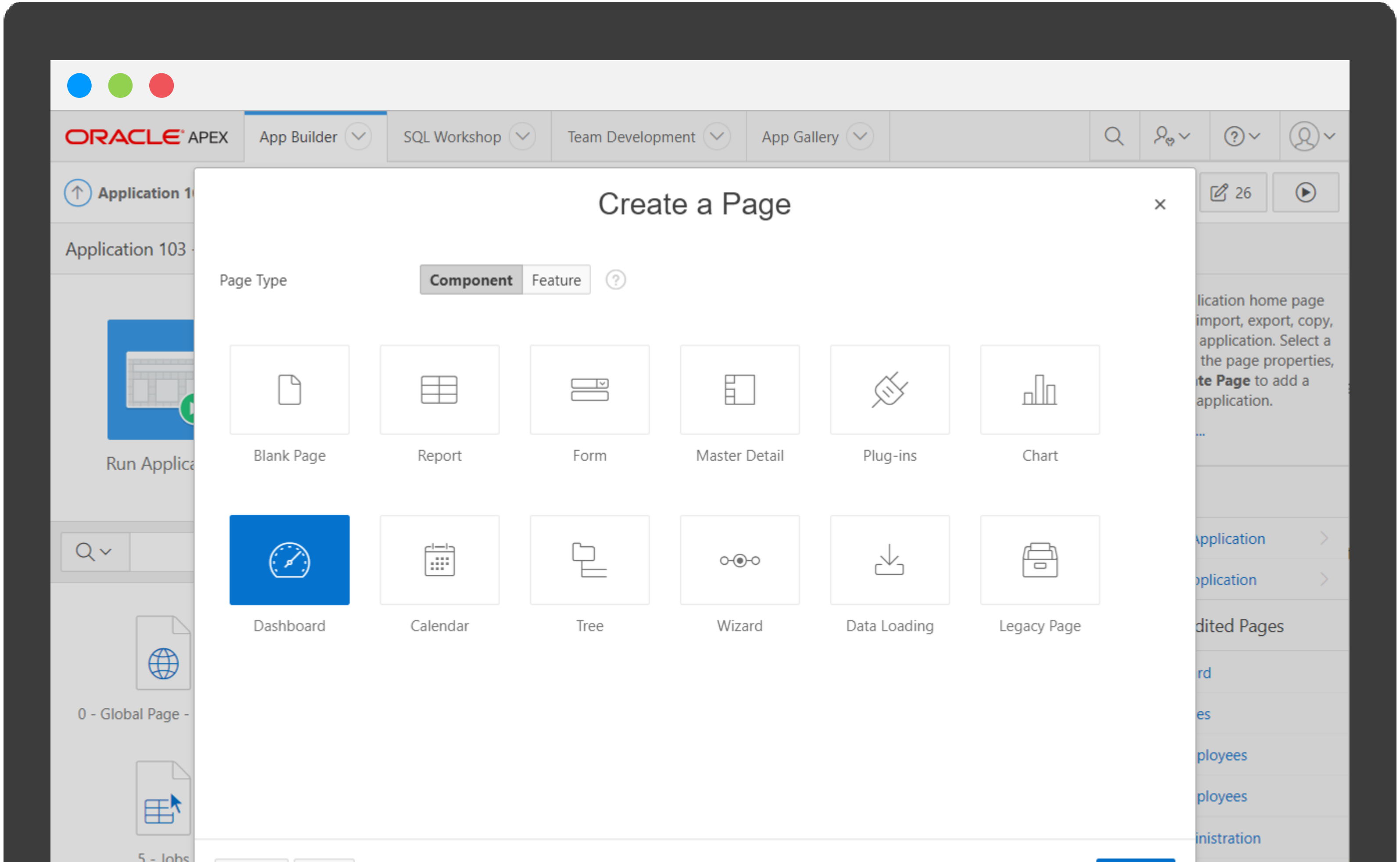
Atkinson
Mozhe

Austin
David

Baer
Hermann

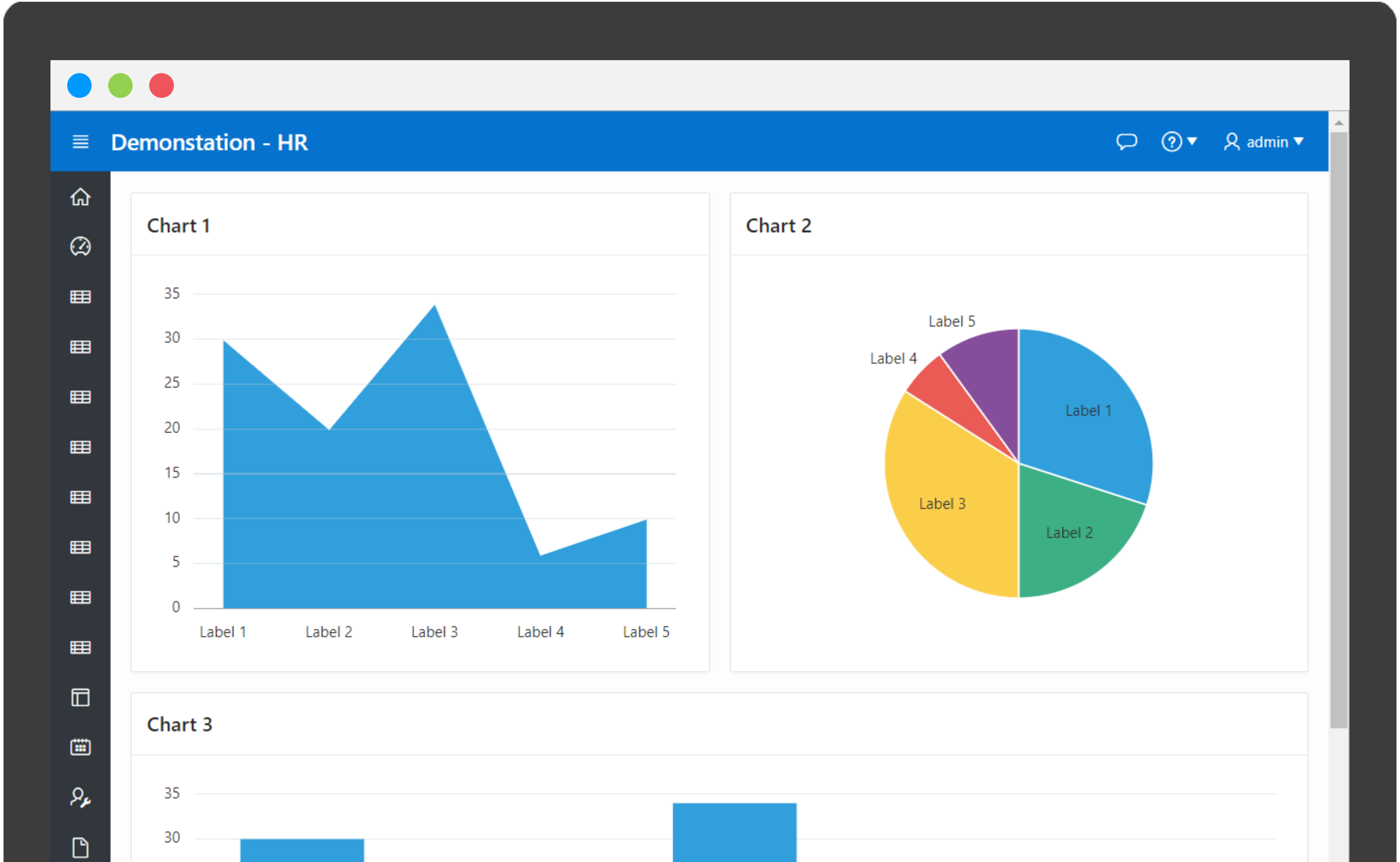
Baida
Shelli

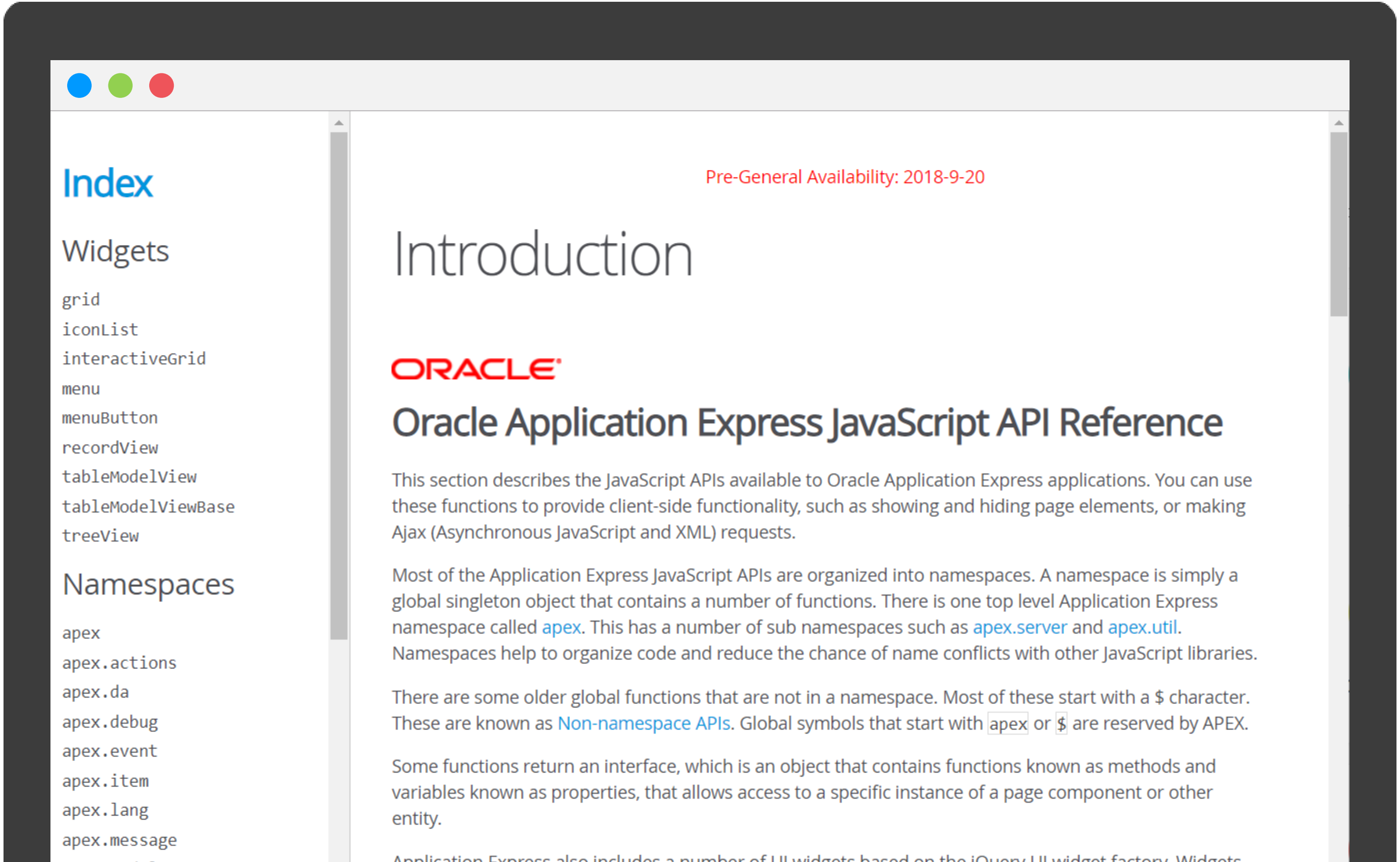
| | |
|---|---|
| First Name | Laura |
| Last Name | Bissot |
| Email | LBISSOT |
| Phone Number | 650.124.5234 |
| Hire Date | 09-NOV-15 |
| Job Id | ST_CLERK |
| Salary | 3300 |
| Manager Id | 121 |
| Department Id | 50 |



Create a Page

Page Type: Component


Demonstation - HR
admin
Chart 1
35
30
25
20
15
10
5
0
Label 1
Label 2
Label 3
Label 4
Label 5
Chart 2
Label 5
Label 4
Label 1
Label 3
Label 2
Chart 3
35
30



# Index

## Widgets

grid
iconList
interactiveGrid
menu
menuButton
recordView
tableModelView
tableModelViewBase
treeView

## Namespaces

apex
apex.actions
apex.da
apex.debug
apex.event
apex.item
apex.lang
apex.message

Pre-General Availability: 2018-9-20

# Introduction

ORACLE

## Oracle Application Express JavaScript API Reference

This section describes the JavaScript APIs available to Oracle Application Express applications. You can use these functions to provide client-side functionality, such as showing and hiding page elements, or making Ajax (Asynchronous JavaScript and XML) requests.

Most of the Application Express JavaScript APIs are organized into namespaces. A namespace is simply a global singleton object that contains a number of functions. There is one top level Application Express namespace called apex. This has a number of sub namespaces such as apex.server and apex.util. Namespaces help to organize code and reduce the chance of name conflicts with other JavaScript libraries.

There are some older global functions that are not in a namespace. Most of these start with a $ character. These are known as Non-namespace APIs. Global symbols that start with `apex` or `$` are reserved by APEX.

Some functions return an interface, which is an object that contains functions known as methods and variables known as properties, that allows access to a specific instance of a page component or other entity.

Application Express also includes a number of UI widgets based on the jQuery UI widget factory. Widgets

# What is still undocumented

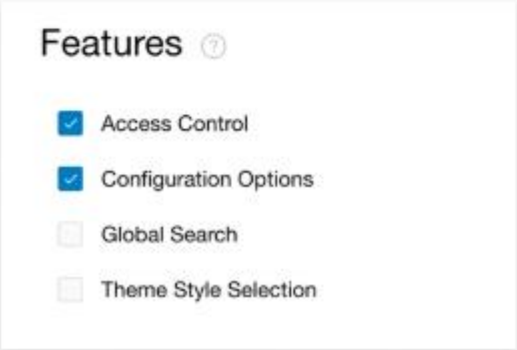
| Widgets |
|---|
| apexTabs |
| collapsible |
| combobox |
| interactiveReport |
| popup |
| popupLOV |

| More widgets |
|---|
| recordView |
| splitter |
| stickyWidget |
| toolbar |

# What is still undocumented

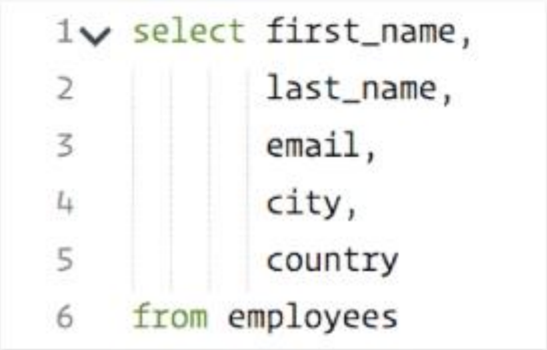
| Namespaces |
|---|
| clipboard |
| locale |
| storage (but in old doc) |
| theme |
| widget.util |

# APEX has many features

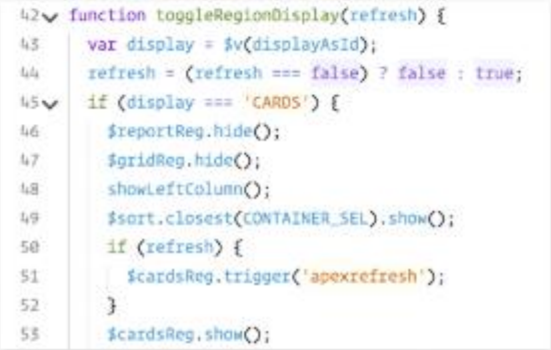


## Features

- ☑ Access Control
- ☑ Configuration Options
- ☐ Global Search
- ☐ Theme Style Selection



```sql
select first_name,
       last_name,
       email,
       city,
       country
from employees
```



```javascript
function toggleRegionDisplay(refresh) {
  var display = $v(displayAsId);
  refresh = (refresh === false) ? false : true;
  if (display === 'CARDS') {
    $reportReg.hide();
    $gridReg.hide();
    showLeftColumn();
    $sort.closest(CONTAINER_SEL).show();
    if (refresh) {
      $cardsReg.trigger('apexrefresh');
    }
    $cardsReg.show();
```

No Code

Low Code

Full Control

# It’s a balance

Declarative features

Easy to learn

# Non-declarative things

Adding checkboxes to a tree

Hiding the search field in the interactive Grid

Customize the rich text editor toolbar

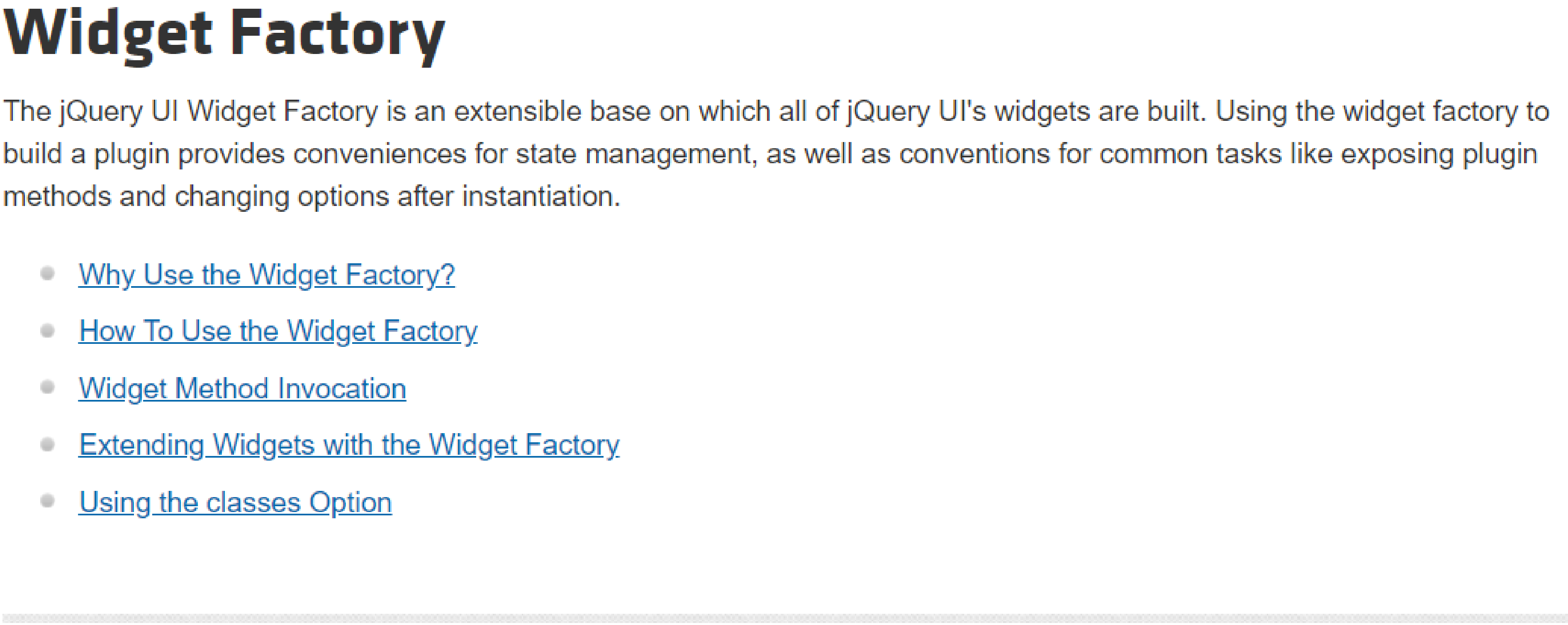
Showing week numbers in the calendar

These are all widgets

# Tips for working with widgets

# The widget factory

https://learn.jquery.com/jquery-ui/widget-factory/



# Widget Factory

The jQuery UI Widget Factory is an extensible base on which all of jQuery UI's widgets are built. Using the widget factory to build a plugin provides conveniences for state management, as well as conventions for common tasks like exposing plugin methods and changing options after instantiation.

- Why Use the Widget Factory?
- How To Use the Widget Factory
- Widget Method Invocation
- Extending Widgets with the Widget Factory
- Using the classes Option

# Getting all widget instances on a page

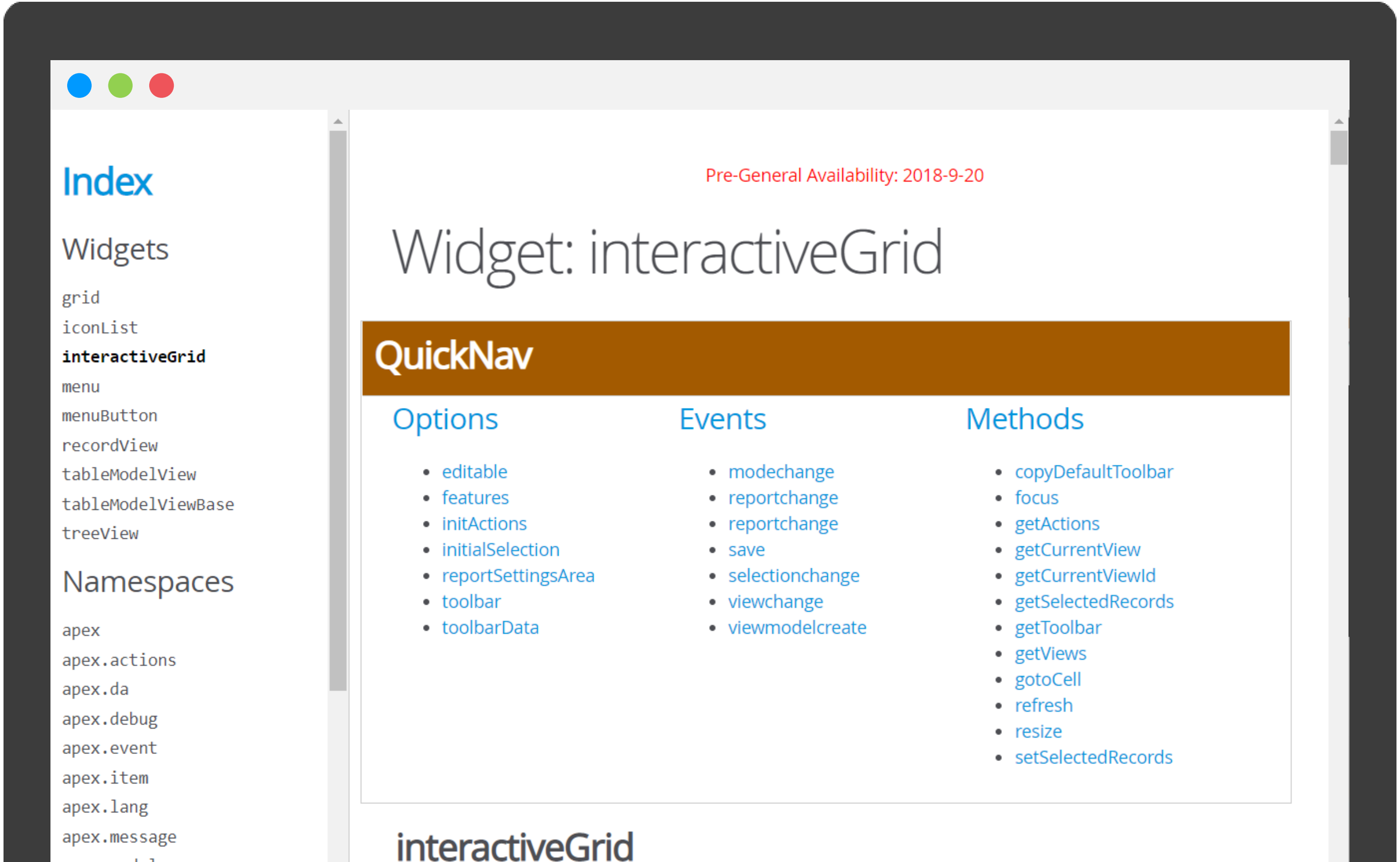
Widgets in jQuery UI have their own selectors. These selectors allow the developer to select widgets by type.

```
$(':<namespace>-<widget name>');

//Examples
$(':apex-interactivegrid');
$(':ui-datepicker');
```

# Widgets are placed under a namespace

| Namespace | Purpose | Widgets |
|---|---|---|
| $.ui | Native jQuery UI widgets | datepicker, dialog, … |
| $.apex | Oracle APEX Widgets | treeView, menu, interactiveGrid, … |
| $.fullCalendar | The calendar region widget | fullCalendar |
| $.oj | Oracle JET widgets | ojChart, ojGantt, … |
| $.demo | Our own widgets | none |



Index

Widgets

- grid
- iconList
- **interactiveGrid**
- menu
- menuButton
- recordView
- tableModelView
- tableModelViewBase
- treeView

Namespaces

- apex
- apex.actions
- apex.da
- apex.debug
- apex.event
- apex.item
- apex.lang
- apex.message

Pre-General Availability: 2018-9-20

# Widget: interactiveGrid

## QuickNav

### Options

- editable
- features
- initActions
- initialSelection
- reportSettingsArea
- toolbar
- toolbarData

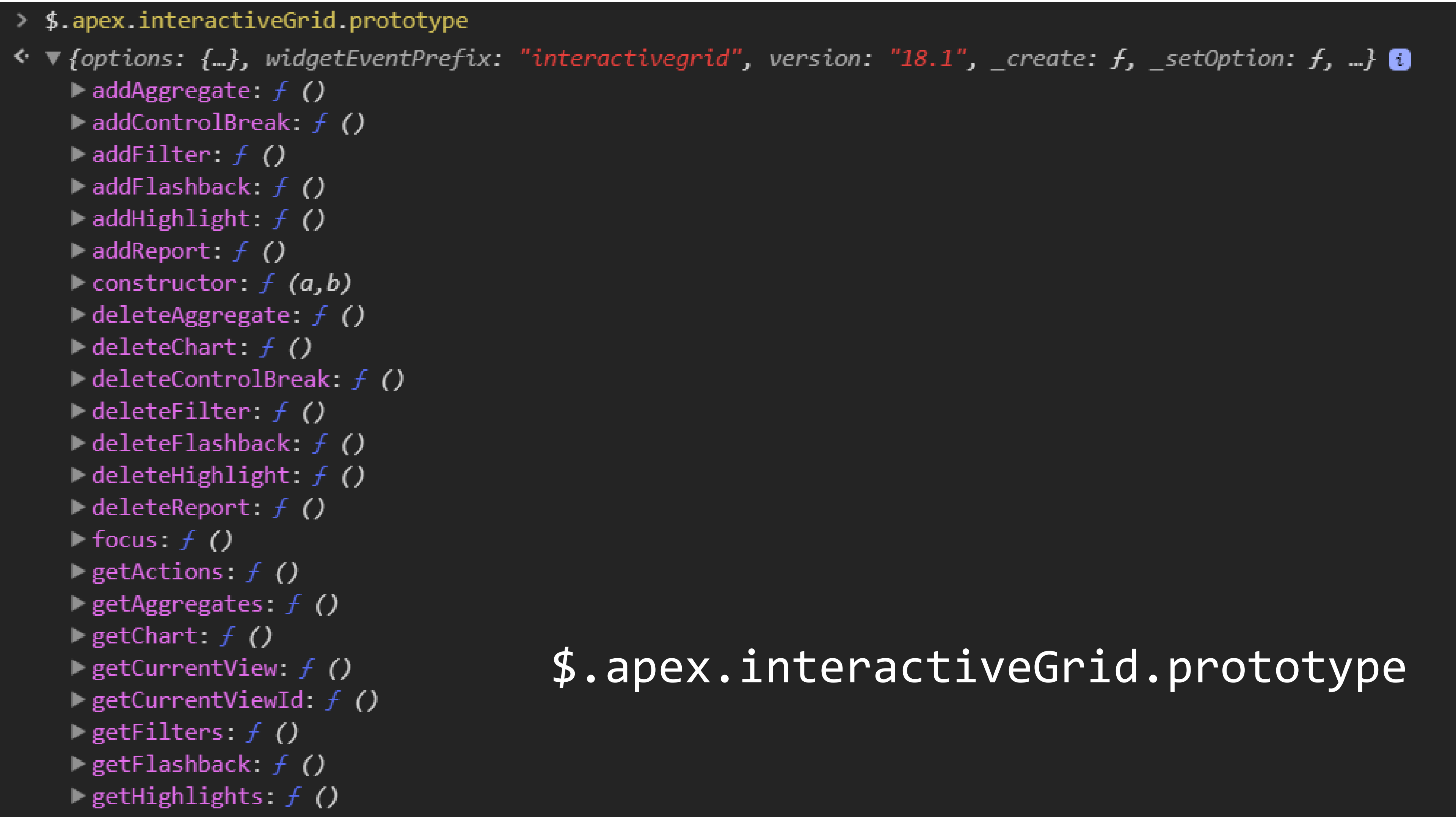
### Events

- modechange
- reportchange
- reportchange
- save
- selectionchange
- viewchange
- viewmodelcreate

### Methods

- copyDefaultToolbar
- focus
- getActions
- getCurrentView
- getCurrentViewId
- getSelectedRecords
- getToolbar
- getViews
- gotoCell
- refresh
- resize
- setSelectedRecords

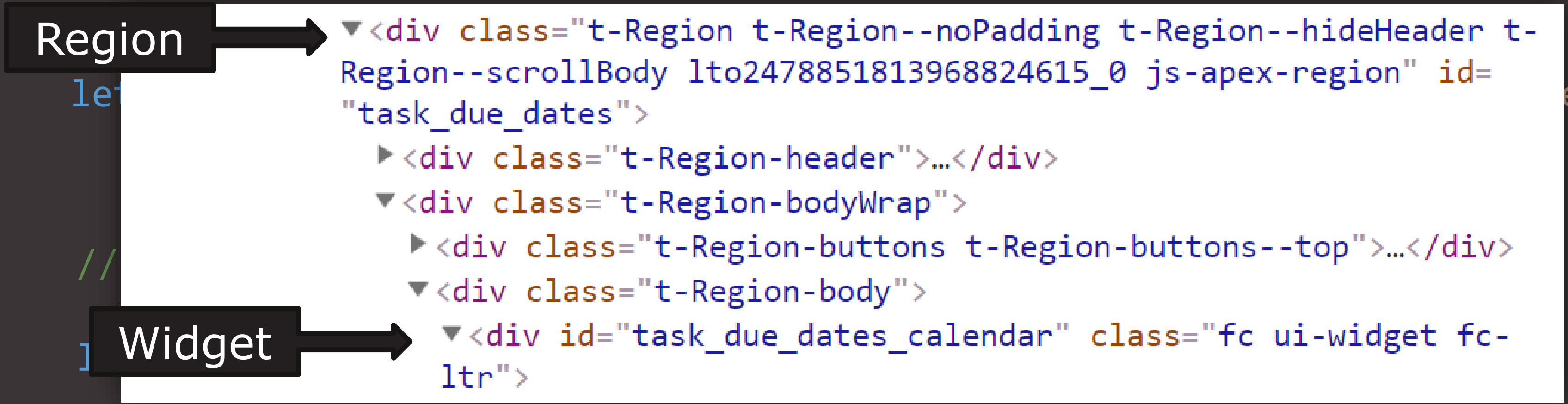
## interactiveGrid



```
> $.apex.interactiveGrid.prototype
< ▼{options: {…}, widgetEventPrefix: "interactivegrid", version: "18.1", _create: ƒ, _setOption: ƒ, …}
    ▶addAggregate: ƒ ()
    ▶addControlBreak: ƒ ()
    ▶addFilter: ƒ ()
    ▶addFlashback: ƒ ()
    ▶addHighlight: ƒ ()
    ▶addReport: ƒ ()
    ▶constructor: ƒ (a,b)
    ▶deleteAggregate: ƒ ()
    ▶deleteChart: ƒ ()
    ▶deleteControlBreak: ƒ ()
    ▶deleteFilter: ƒ ()
    ▶deleteFlashback: ƒ ()
    ▶deleteHighlight: ƒ ()
    ▶deleteReport: ƒ ()
    ▶focus: ƒ ()
    ▶getActions: ƒ ()
    ▶getAggregates: ƒ ()
    ▶getChart: ƒ ()
    ▶getCurrentView: ƒ ()
    ▶getCurrentViewId: ƒ ()
    ▶getFilters: ƒ ()
    ▶getFlashback: ƒ ()
    ▶getHighlights: ƒ ()
```

$.apex.interactiveGrid.prototype

# Using a region widget method



```
▼<div class="t-Region t-Region--noPadding t-Region--hideHeader t-Region--scrollBody lto2478851813968824615_0 js-apex-region" id="task_due_dates">
  ▶<div class="t-Region-header">…</div>
  ▼<div class="t-Region-bodyWrap">
    ▶<div class="t-Region-buttons t-Region-buttons--top">…</div>
    ▼<div class="t-Region-body">
      ▼<div id="task_due_dates_calendar" class="fc ui-widget fc-ltr">
```

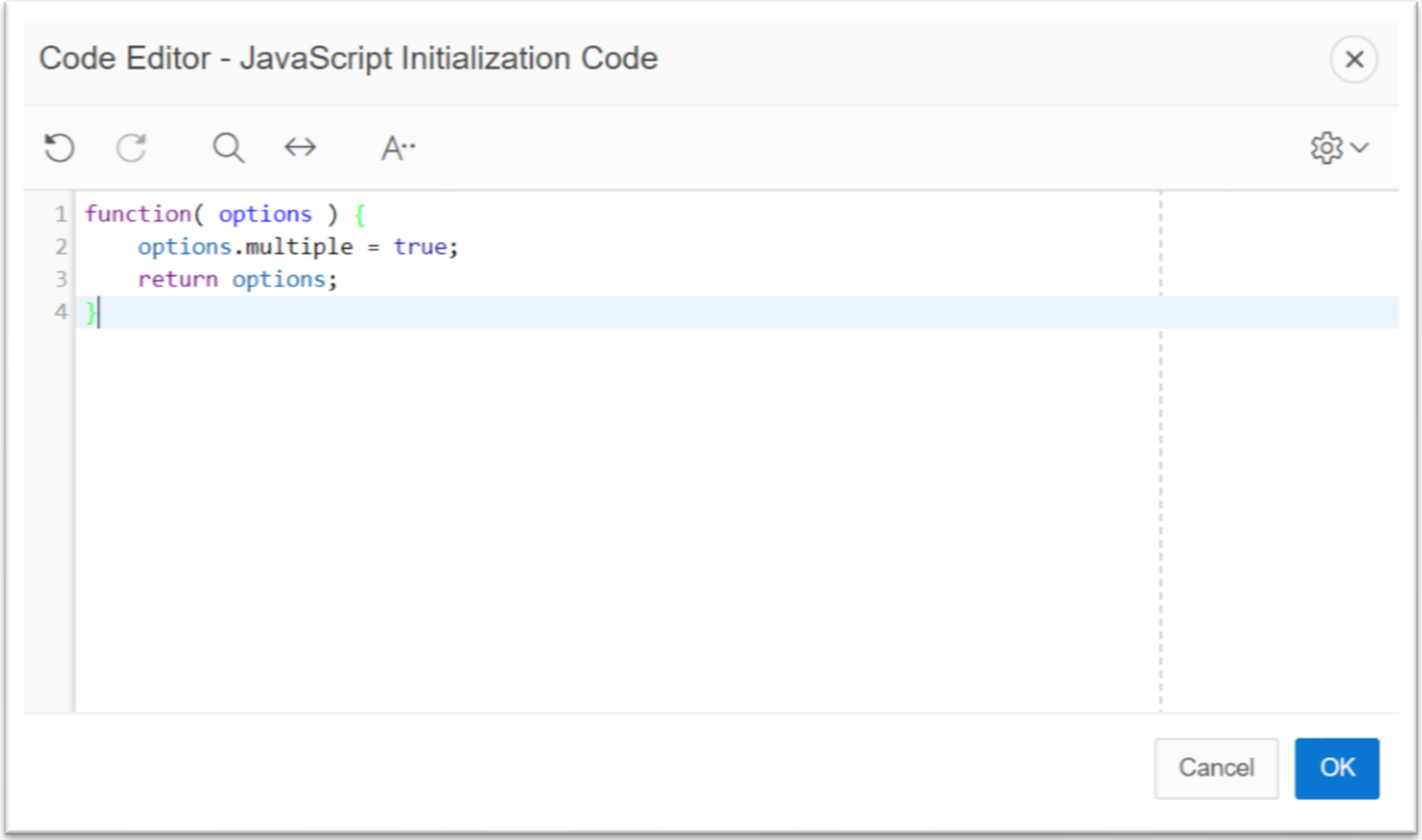
# Using a page item widget

```
$('#P1_DATE').datepicker('getDate');
// returns: Thu Jun 14 2018 00:00:00 GMT+0200 (Central
Europe Daylight Time)
```

# When to use them

- ✓ Changing options
- ✓ Executing methods
- ✓ Acting on events

# Changing widget options on initialization



```
function( options ) {
    options.multiple = true;
    return options;
}
```

## Changing options

The options are part of the widget's state, so we can set options after initialization as well.

```
apex.region('mychart').call('option', 'orientation', 'horizontal');
```

# Acting on events

Event names always start with the widget name in lowercase

```
$('#emp').on('interactivegridcreate', function (event, data) {});

$('#emp').on('interactivegridselectionchange', function (event, data) {
  console.log(data);
});
```

All widgets have a **create** event

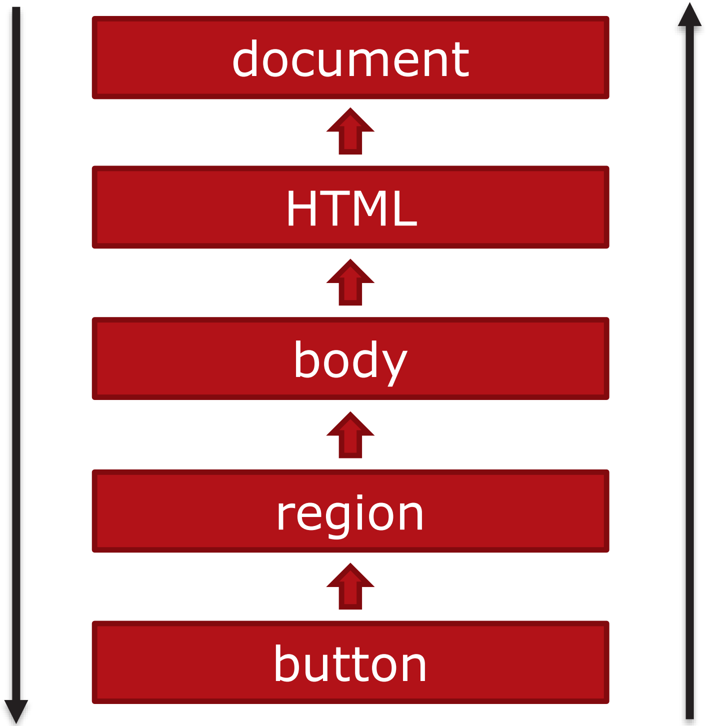
# Advanced usage

## Extending a widget

```
$.widget('apex.interactiveGrid', $.apex.interactiveGrid, {
  refresh: function () {
    alert('Hello, I should refresh now..');
    this._super();
  }
});
```

# Tips for working with events

# Event capturing & bubbling



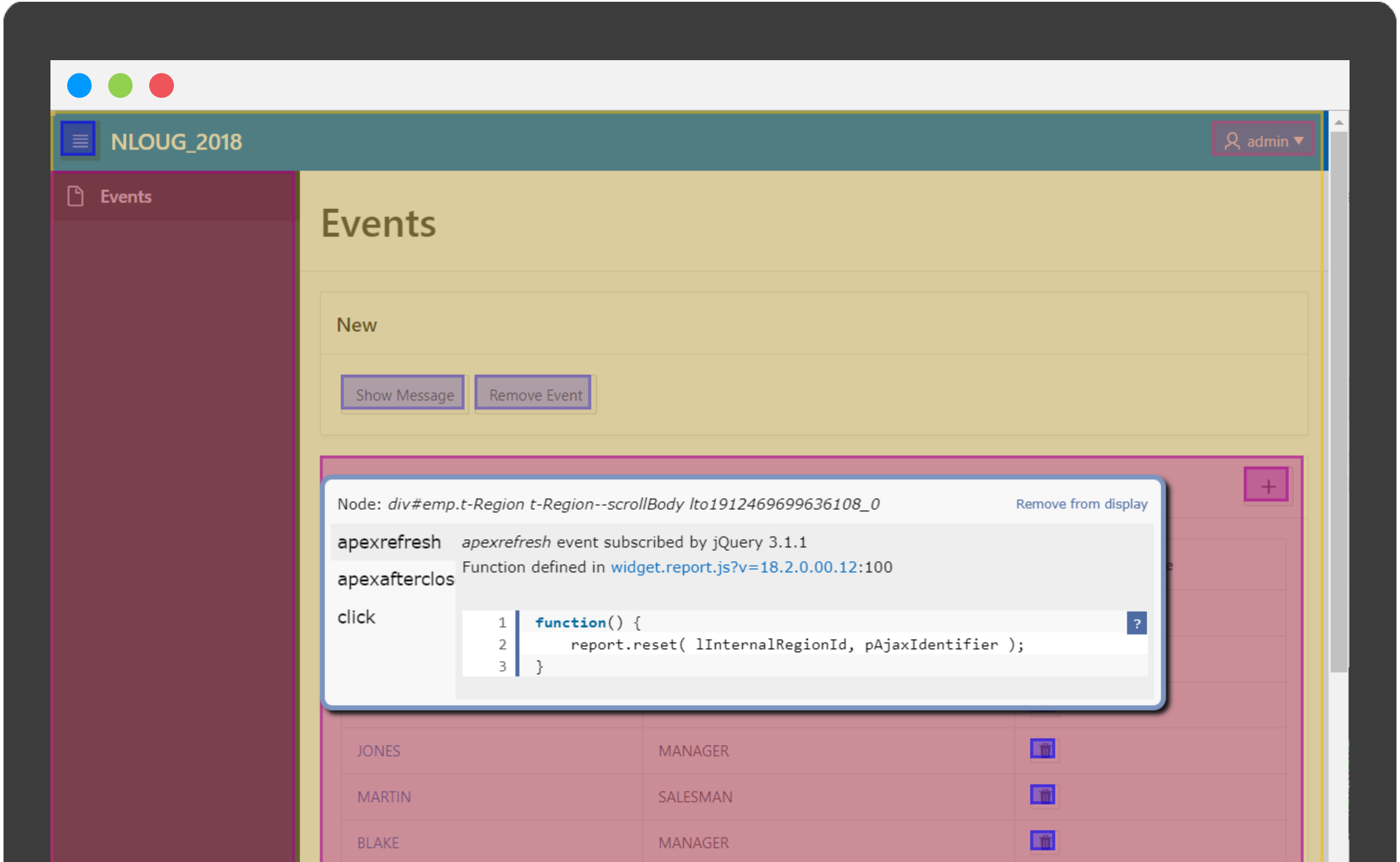
Event capturing

Event bubbling

# Visualizing event listeners

## A few options

- ✓ Browser extension (Visual Event in Chrome)
- ✓ Browser development tools

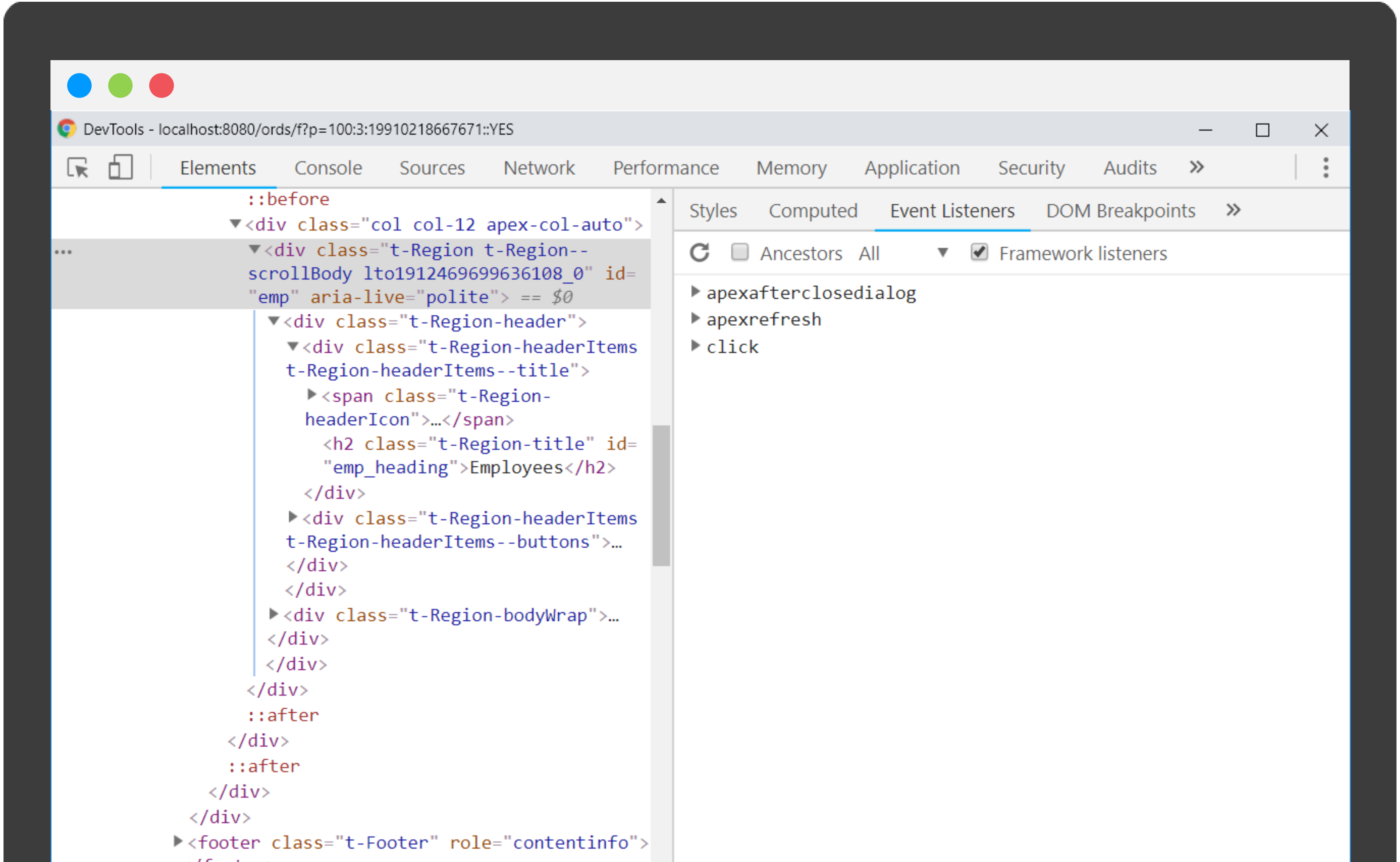


Node: *div#emp.t-Region t-Region--scrollBody lto1912469699636108_0*

*apexrefresh* event subscribed by jQuery 3.1.1
Function defined in widget.report.js?v=18.2.0.00.12:100

```
function() {
    report.reset( lInternalRegionId, pAjaxIdentifier );
}
```

| | | |
|---|---|---|
| JONES | MANAGER | |
| MARTIN | SALESMAN | |
| BLAKE | MANAGER | |



```
          ::before
        ▼<div class="col col-12 apex-col-auto">
...       ▼<div class="t-Region t-Region--
          scrollBody lto1912469699636108_0" id=
          "emp" aria-live="polite"> == $0
            ▼<div class="t-Region-header">
              ▼<div class="t-Region-headerItems
              t-Region-headerItems--title">
                ▶<span class="t-Region-
                headerIcon">…</span>
                 <h2 class="t-Region-title" id=
                 "emp_heading">Employees</h2>
                </div>
              ▶<div class="t-Region-headerItems
              t-Region-headerItems--buttons">…
              </div>
              </div>
            ▶<div class="t-Region-bodyWrap">…
            </div>
            </div>
          </div>
          ::after
        </div>
        ::after
      </div>
    </div>
  ▶<footer class="t-Footer" role="contentinfo">
```

- ▶ apexafterclosedialog
- ▶ apexrefresh
- ▶ click

# Tips for writing JavaScript

# Using loops

```
var fruits = ['apple', 'banana', 'grape']

for (var i = 0; i < fruits.length; i++) {
  console.log(fruits[i])
}

fruits.forEach(fruit => {
  console.log(fruit)
})

for (const fruit of fruits) {
  console.log(fruit)
}
```

# Making arrays unique

```javascript
var fruits = ['apple', 'banana', 'grape', 'grape']

var unique = fruits.filter(function(value, index, self) {
  return self.indexOf(value) === index;
});

let unique = [...new Set(fruits)];
```

# Using template literals

## For adding variables to strings

```
var person = {
  firstName: 'Larry',
  lastName: 'Ellison'
}

console.log('Hi ' + person.firstName + ' ' + person.lastName)

console.log(`Hi ${person.firstName} ${person.lastName}`)

apex.util.applyTemplate('Hi &P1_FIRSTNAME. &P1_LASTNAME.')
```

# If, else if..

## Same for switch statement

```
function greet (jobName) {
  var greeting = 'Hi'
  if (jobName === 'CLERK') {
    greeting = 'Yo'
  } else if (jobName === 'MANAGER') {
    greeting = 'Hello sir'
  } else if (jobName === 'PRESIDENT') {
    greeting = 'How do you do sir'
  }
  console.log(greeting)
}
```

# If, else if..

## Clean code

```
const greetings = {
  CLERK: 'Yo',
  MANAGER: 'Hello sir',
  PRESIDENT: 'How do you do sir'
}

function greet (jobName) {
  console.log(greetings[jobName] || 'Hi')
}
```

# Using apex.server.process

## Added in APEX 4.2 (Most common way)

```
apex.server.process("get_job", {
}, {
  success: function (data) {},
  error: function (err) {}
})
```

# Using a promise

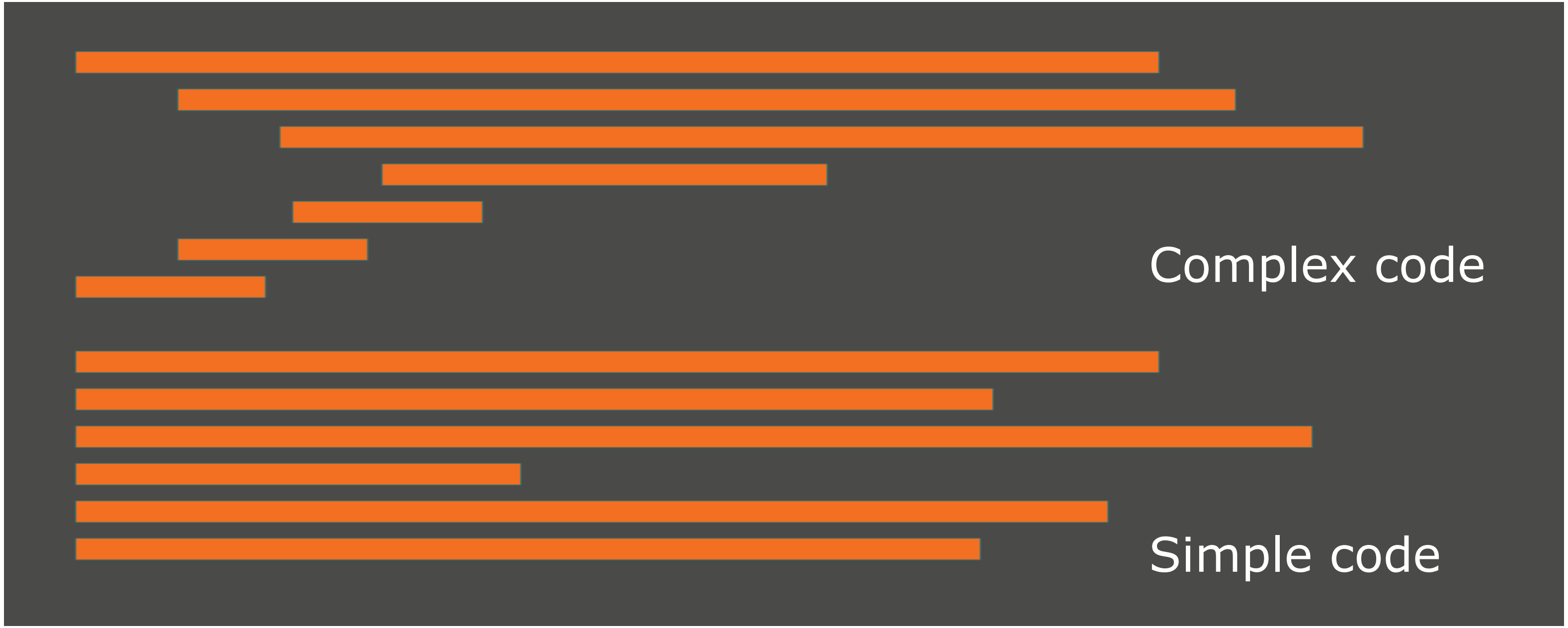
## Simple and works in all supported browsers

```
apex.server
   .process("get_job", {})
   .then(response => {})
   .catch(err => {})
```

Instead of callbacks

# Refactor complex code

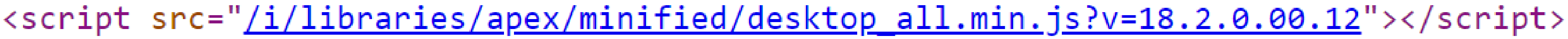
## Makes it easier to read



# Tips for structuring code in APEX

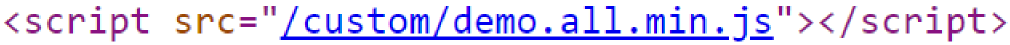
# Production mode

## APEX files



```
<script src="/i/libraries/apex/minified/desktop_all.min.js?v=18.2.0.00.12"></script>
```

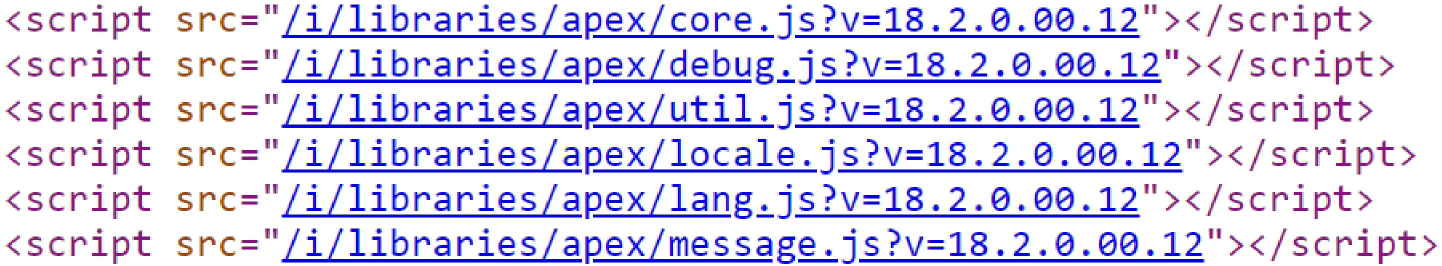
## Custom files



```
<script src="/custom/demo.all.min.js"></script>
```

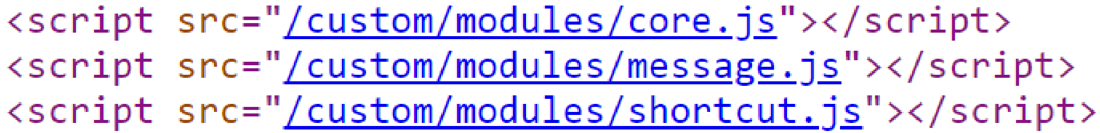
# Debug mode

## APEX files



```
<script src="/i/libraries/apex/core.js?v=18.2.0.00.12"></script>
<script src="/i/libraries/apex/debug.js?v=18.2.0.00.12"></script>
<script src="/i/libraries/apex/util.js?v=18.2.0.00.12"></script>
<script src="/i/libraries/apex/locale.js?v=18.2.0.00.12"></script>
<script src="/i/libraries/apex/lang.js?v=18.2.0.00.12"></script>
<script src="/i/libraries/apex/message.js?v=18.2.0.00.12"></script>
```

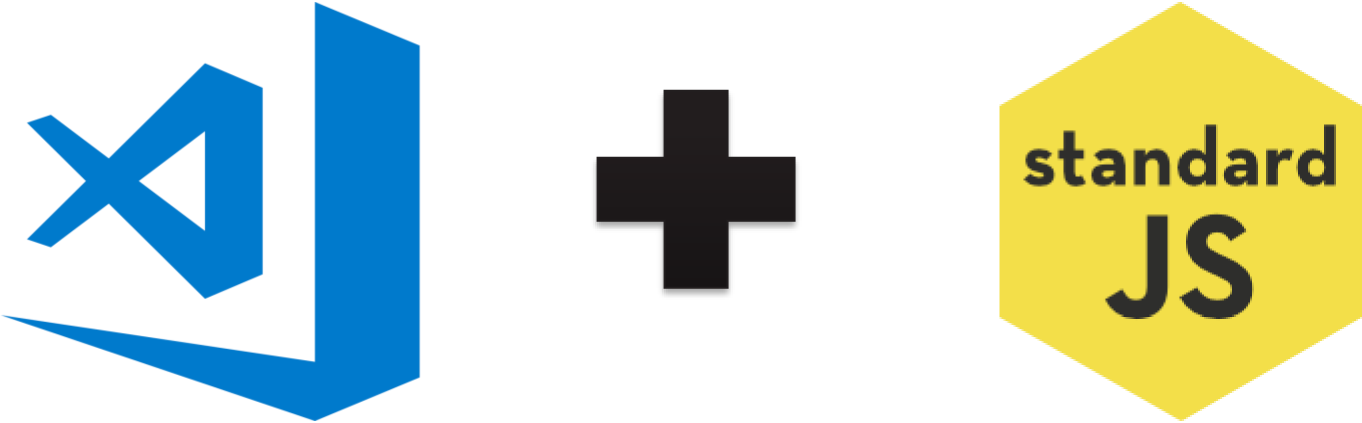
## Custom files



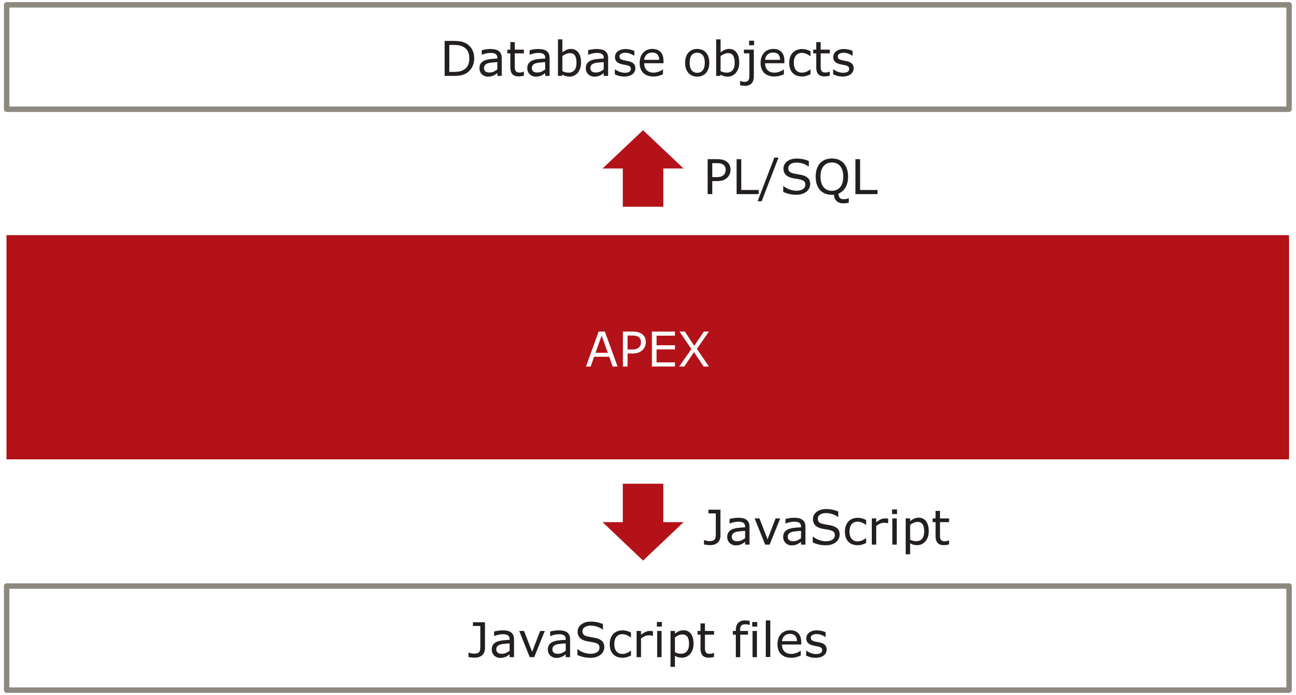
```
<script src="/custom/modules/core.js"></script>
<script src="/custom/modules/message.js"></script>
<script src="/custom/modules/shortcut.js"></script>
```

# Style consistency

## Format your code on file save



# Place your code where it belongs



Database objects

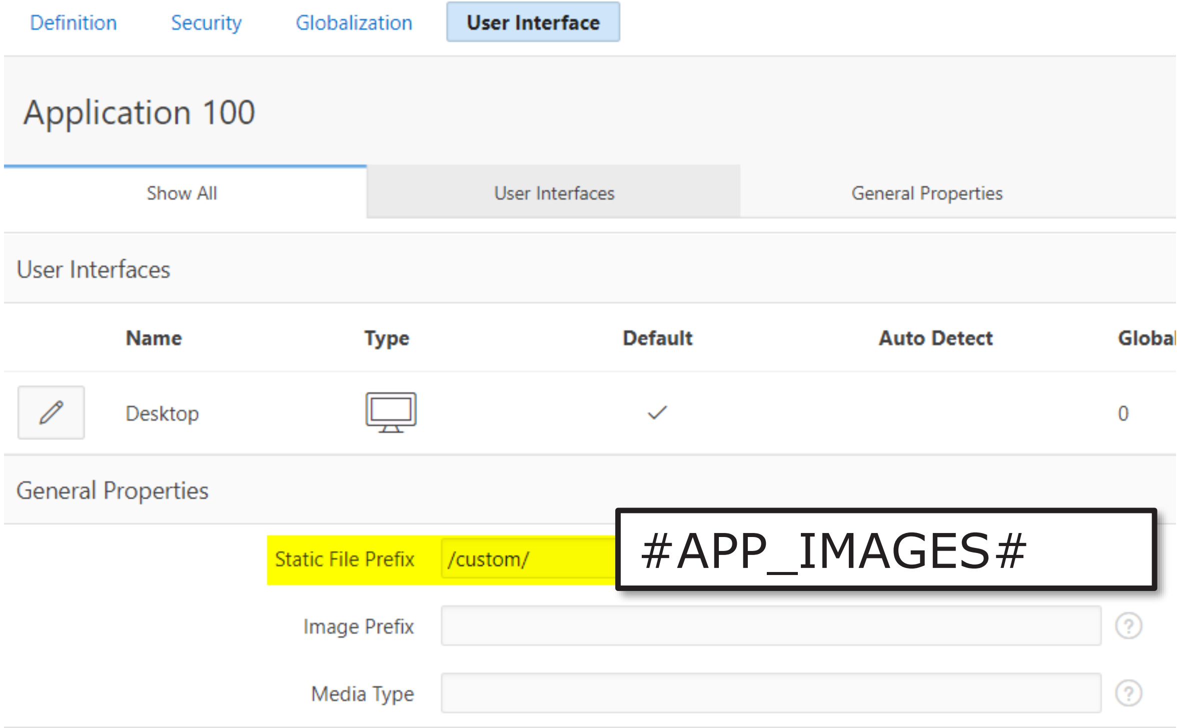
PL/SQL

APEX

JavaScript

JavaScript files

# Avoid #IMAGE_PREFIX# as custom file location



Application 100

User Interfaces

|  | Name | Type | Default | Auto Detect | Global |
|---|---|---|---|---|---|
|  | Desktop |  | ✓ |  | 0 |

General Properties

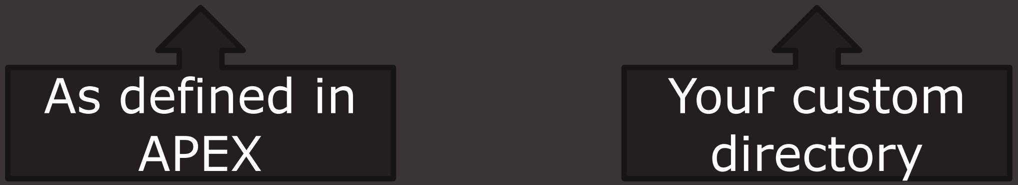
| | |
|---|---|
| Static File Prefix | /custom/ |
| Image Prefix | |
| Media Type | |

#APP_IMAGES#

# Configure extra document directory in HTTP server

## Example for $CATALINA_BASE/conf/server.xml

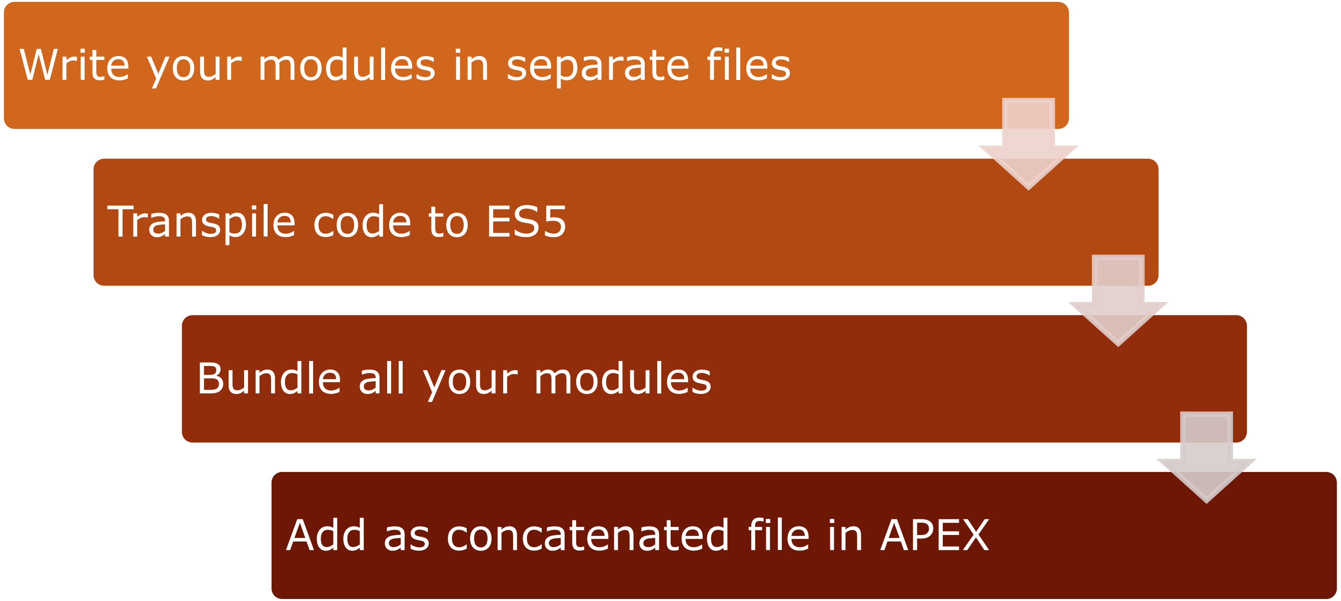
```
<Host name="localhost" appBase="webapps"
unpackWARs="true" autoDeploy="true">
  ...
  <!-- A custom path to my files -->
  <Context path="/custom" docBase="C:\git\nlOUG-Demo\dist" />
</Host>
```



As defined in APEX

Your custom directory

# Modular JavaScript in APEX



Write your modules in separate files

Transpile code to ES5

Bundle all your modules

Add as concatenated file in APEX

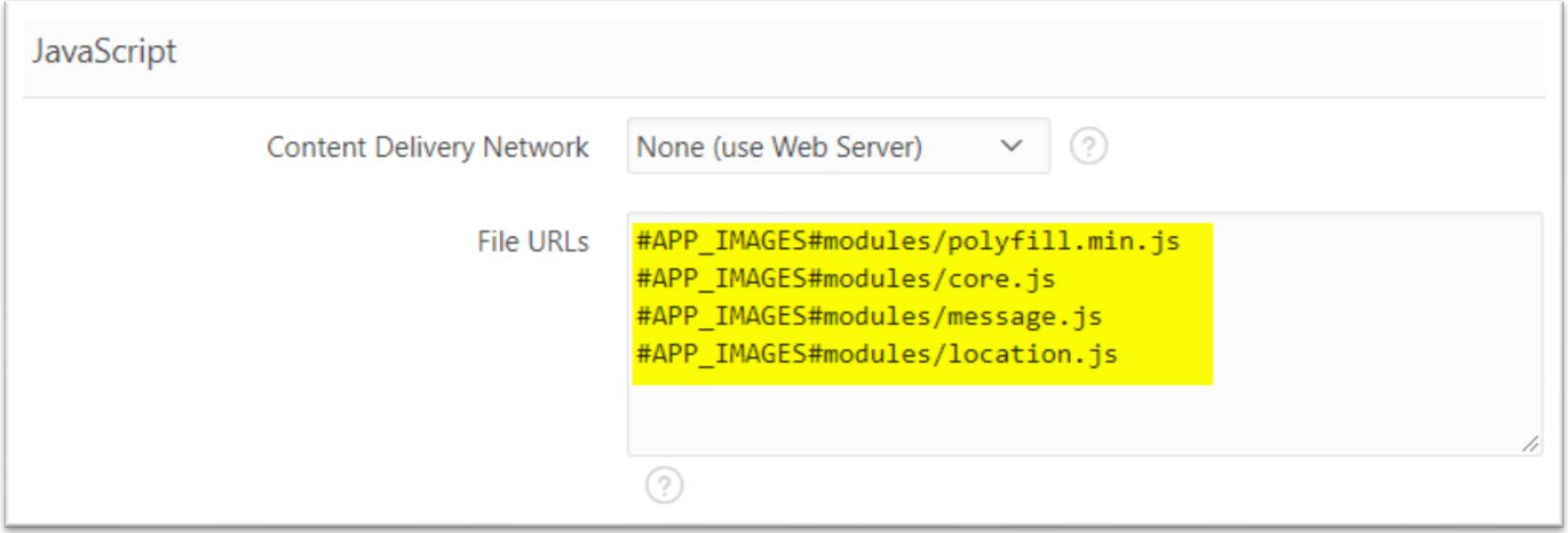
# Transpile code to ES5 and bundle the modules

## Automate the process

node JS
Gulp
BABEL

# Add individual modules in APEX

## User interface attributes -> Desktop -> JavaScript



JavaScript

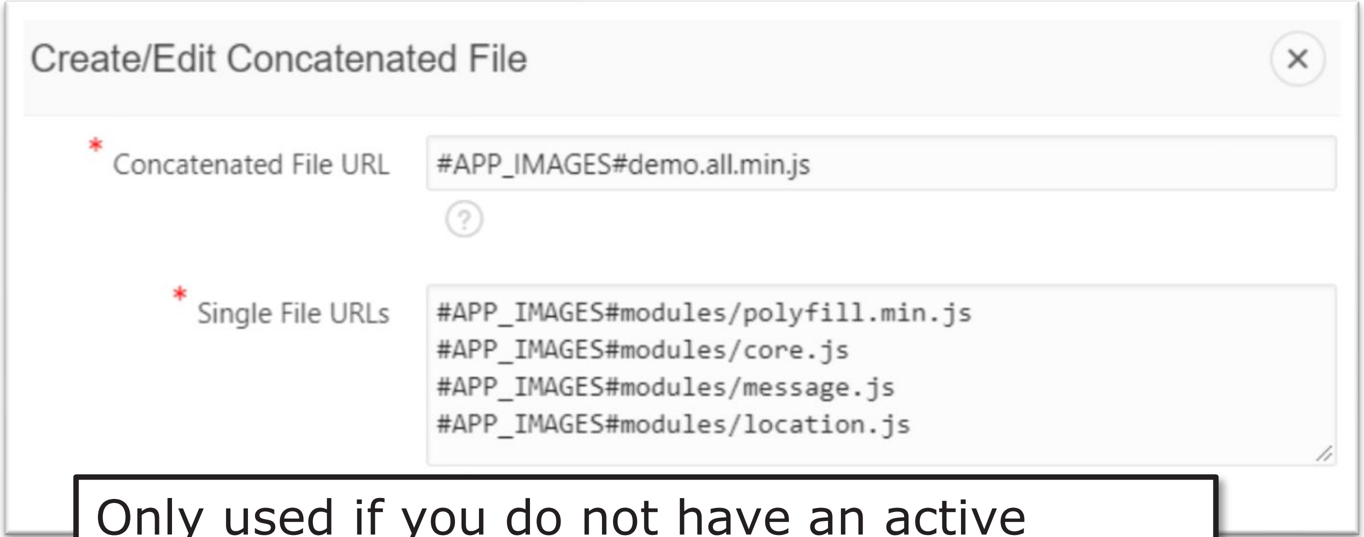
Content Delivery Network: None (use Web Server)

File URLs:

```
#APP_IMAGES#modules/polyfill.min.js
#APP_IMAGES#modules/core.js
#APP_IMAGES#modules/message.js
#APP_IMAGES#modules/location.js
```

# Add concatenated file in APEX

## User interface attributes -> Desktop -> Concatenated file



Create/Edit Concatenated File

* Concatenated File URL: #APP_IMAGES#demo.all.min.js

* Single File URLs:

```
#APP_IMAGES#modules/polyfill.min.js
#APP_IMAGES#modules/core.js
#APP_IMAGES#modules/message.js
#APP_IMAGES#modules/location.js
```

Only used if you do not have an active application builder session

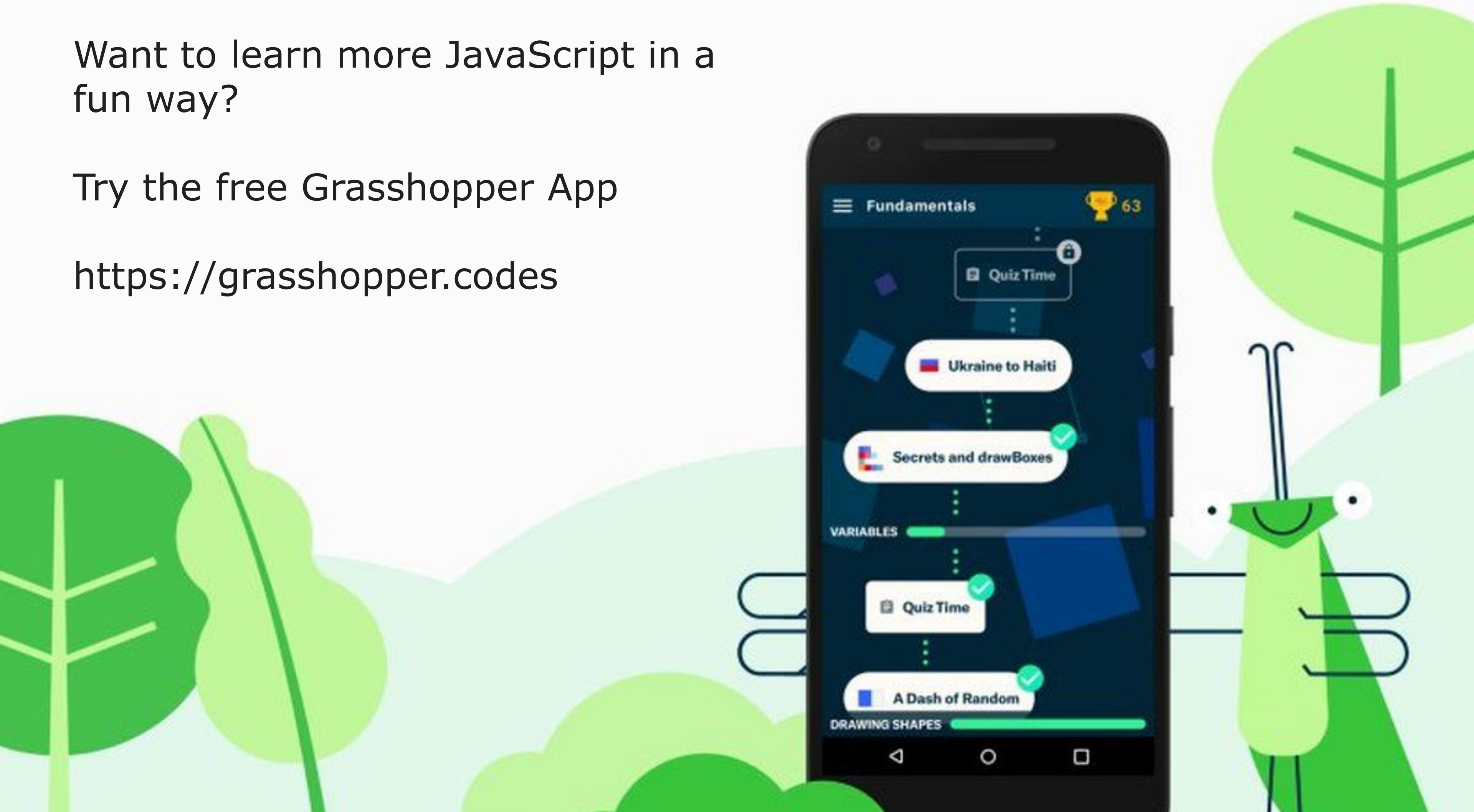
# In a nutshell

- JavaScript in files
- DRY
- Pick a coding standard
- Write simple code
- Keep an eye on performance
- Keep learning

Want to learn more JavaScript in a fun way?

Try the free Grasshopper App

https://grasshopper.codes



Thank you