19.11.2025

The Interactive Grid: Blessing or Curse? Let's find out!

Timo Herwix, Senior Consultant

code of change

Hyand by GOD | MT

Who am I?

Timo Herwix
Senior Consultant

Oracle ACE Pro

- Senior Consultant at Hyand Solutions since 2019
- Previously worked as a Data Warehouse Developer
- Oracle APEX since 2016
- Oracle Databases since 2008
- Blog author, conference speaker
- Born in 1983, two children and living in Germany

Hyand

Our key facts

Hyand

Germany

- Brunswick
- Ratingen
- Hamburg
- Dortmund
- Cologne
- Frankfurt
- Munich
- Berlin

Poland

- Warsaw

Lithuania

- Vilnius
- Kaunas

Romania

- Cluj-Napoca

India

- Pune

850+
Employees

150+
Customers

110+
million €
turnover

What is the Interactive Grid?

The Interactive Grid is a powerful component in Oracle APEX that combines the features of Interactive Reports and Tabular Forms. It allows users to view, edit, and manipulate data in a single, dynamic interface. It was introduced with APEX 5.1 to provide a modern, spreadsheet-style user interface.

What is the Interactive Grid?

Similar to Interactive Report, users can customize the display:

- Filtering and sorting: Apply multiple filters and sort across different columns.
- Aggregations: Easily calculate sums, averages, counts, and more.
- Control breaks: Group your data by column values for better organization.
- Computations: Create new columns using formulas.
- Customizations: Hide or show columns, rearrange them with drag & drop, and freeze columns.
- Saving reports: Users can save their views as either private or public reports.
- Export: Export your data to CSV, Excel, and other formats.

What is the Interactive Grid?

Editable interactive grids let users easily change or update data.

- Inline editing: Users can edit cells right on the page, just like in a spreadsheet, without needing to reload.
- Add/delete rows: You can create new records or delete the ones you don't need anymore.
- CRUD operations: Fully supports creating, reading, updating, and deleting data.
- Master-detail relationships: Easily set up master-detail forms with multiple interactive grids on one page.

Architecture of the Interactive Grid.

Architecture of the Interactive Grid.

The Interactive Grid is a highly complex JavaScript construct that combines the advantages of a powerful client-side model (speed, interactivity) with the declarative configuration of APEX.

The architecture can be divided into three main components:

- Client side (browser)
- Data model
- Server side (APEX engine & database)

The Ups and Downs of the Interactive Grid!

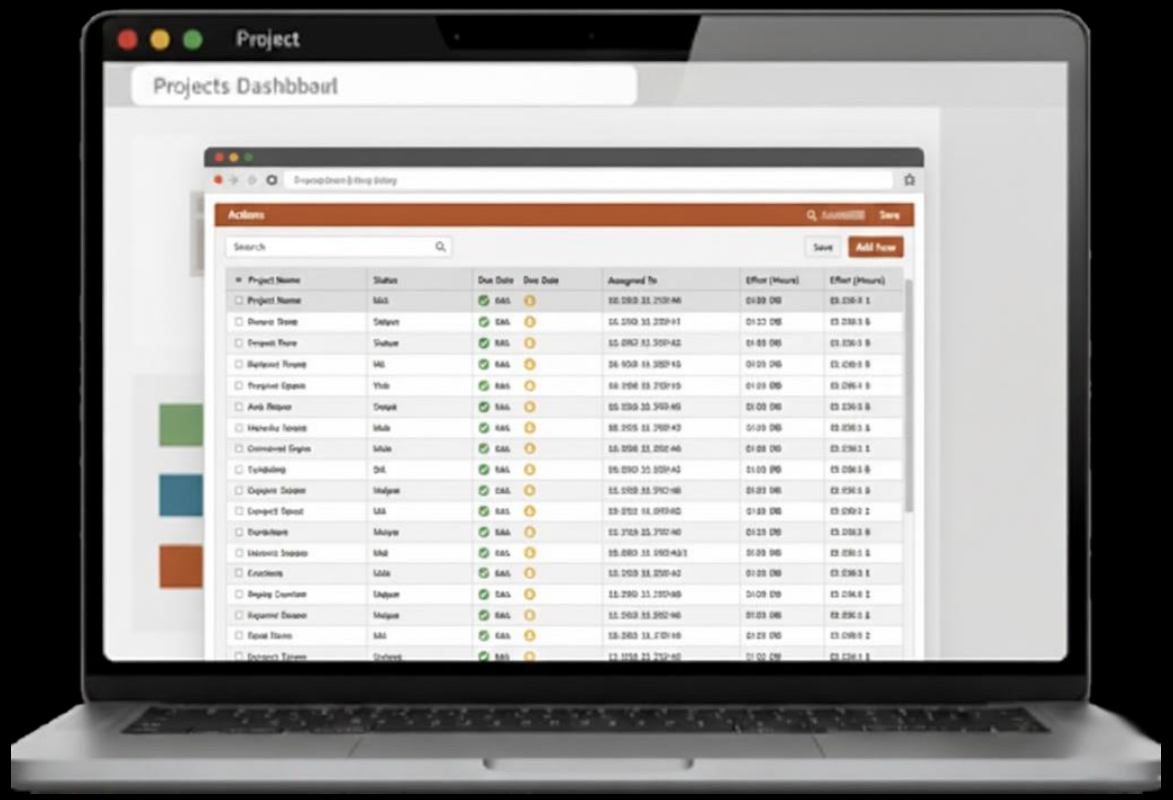
Hyand

The Ups and Downs of the Interactive Grid!

- Maximum interactivity: Allows direct inline editing of data like Excel
- Combined functionality: Merges reporting functions with editing features in one component
- Client-side architecture: Uses a client-side data model for fast execution
- Extensibility: Offers a JavaScript API for customizing
- Master-detail capability: Simplifies creating master-detail forms
- Modern look: Integrates with the Universal Theme for a modern appearance

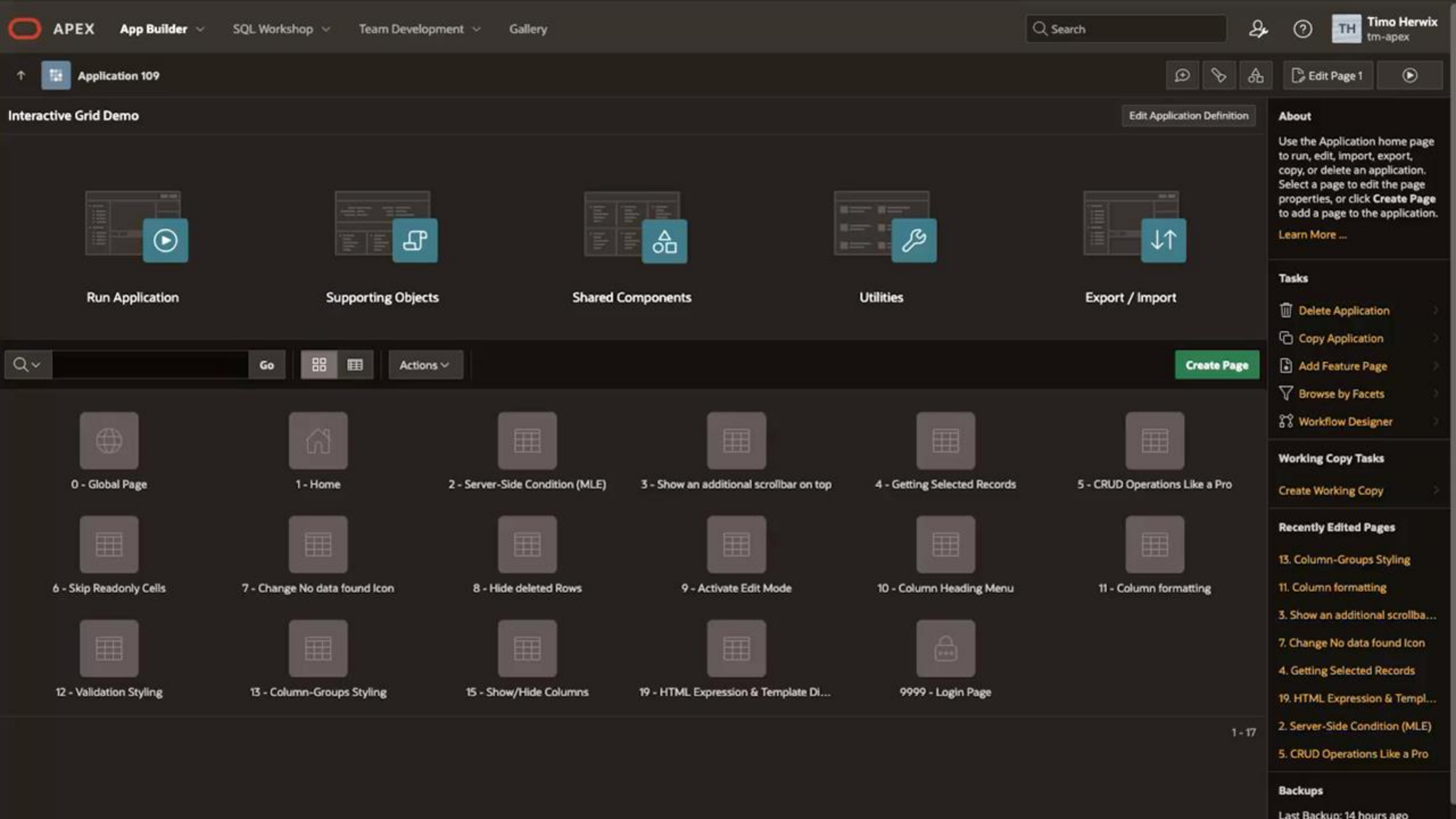
The Ups and Downs of the Interactive Grid!

- Complexity for developers: The client-side data model (apex.model) is more complex than other reports, requiring advanced JavaScript knowledge for customizations.
- Performance with large data sets: IG is optimized for scrolling, but handling very large data sets can still slow performance compared to server-side reports.
- Overhead without editability: If editing isn't needed, IG can add unnecessary loading time and JavaScript logic compared to simpler reports.
- Learning curve: Users who want to use the full range of features may need a short training period to understand all the options



Interactive Grid: Now in action!

Hyand


APEX
App Builder
SQL Workshop
Team Development
Gallery
Search
Timo Herwix
tm-apex
Application 109
Edit Page 1
Interactive Grid Demo
Edit Application Definition
Run Application
Supporting Objects
Shared Components
Utilities
Export / Import
Go
Actions
Create Page
0 - Global Page
1 - Home
2 - Server-Side Condition (MLE)
3 - Show an additional scrollbar on top
4 - Getting Selected Records
5 - CRUD Operations Like a Pro
6 - Skip Readonly Cells
7 - Change No data found Icon
8 - Hide deleted Rows
9 - Activate Edit Mode
10 - Column Heading Menu
11 - Column formatting
12 - Validation Styling
13 - Column-Groups Styling
15 - Show/Hide Columns
19 - HTML Expression & Template Di...
9999 - Login Page
1 - 17
About
Use the Application home page to run, edit, import, export, copy, or delete an application. Select a page to edit the page properties, or click Create Page to add a page to the application.
Learn More ...
Tasks
Delete Application
Copy Application
Add Feature Page
Browse by Facets
Workflow Designer
Working Copy Tasks
Create Working Copy
Recently Edited Pages
13. Column-Groups Styling
11. Column formatting
3. Show an additional scrollba...
7. Change No data found Icon
4. Getting Selected Records
19. HTML Expression & Templ...
2. Server-Side Condition (MLE)
5. CRUD Operations Like a Pro
Backups
Last Backup: 14 hours ago

1

Interactive Grid
like a Pro

2

JavaSript
Enhancements

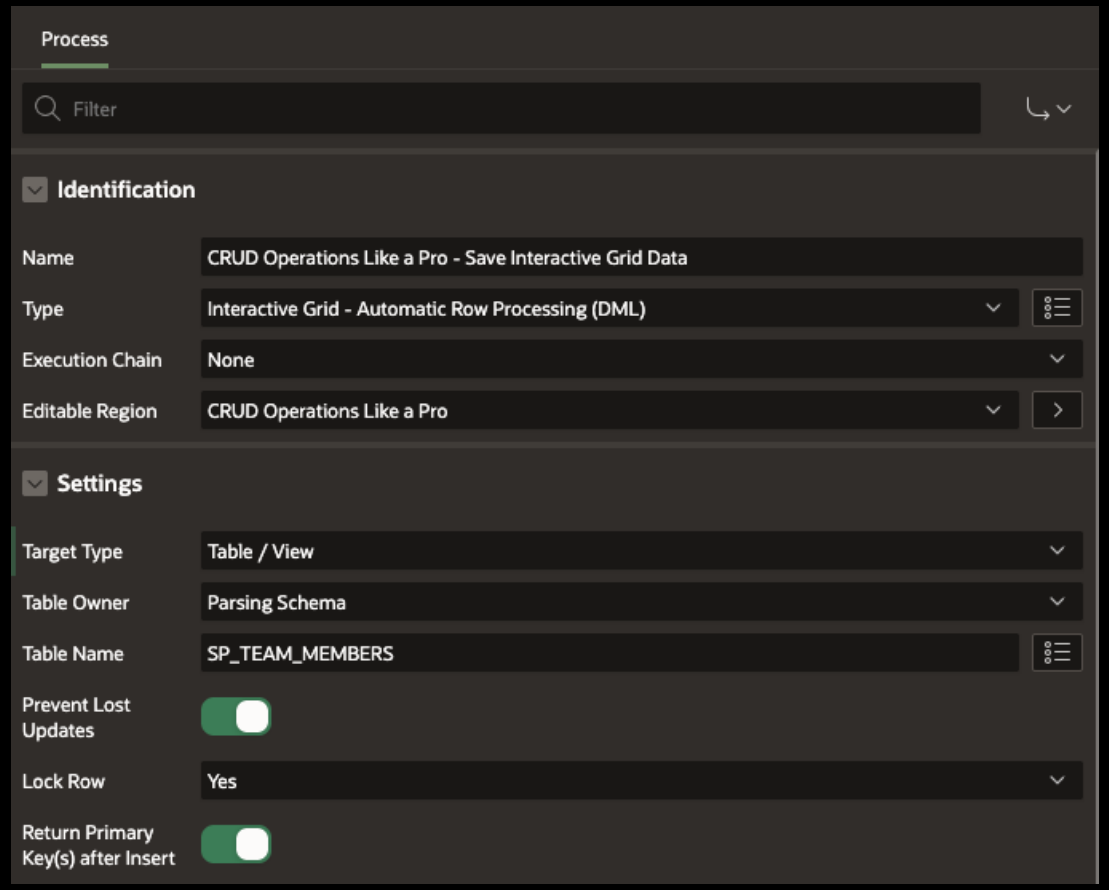
3

CSS
Enhancements

Hyand

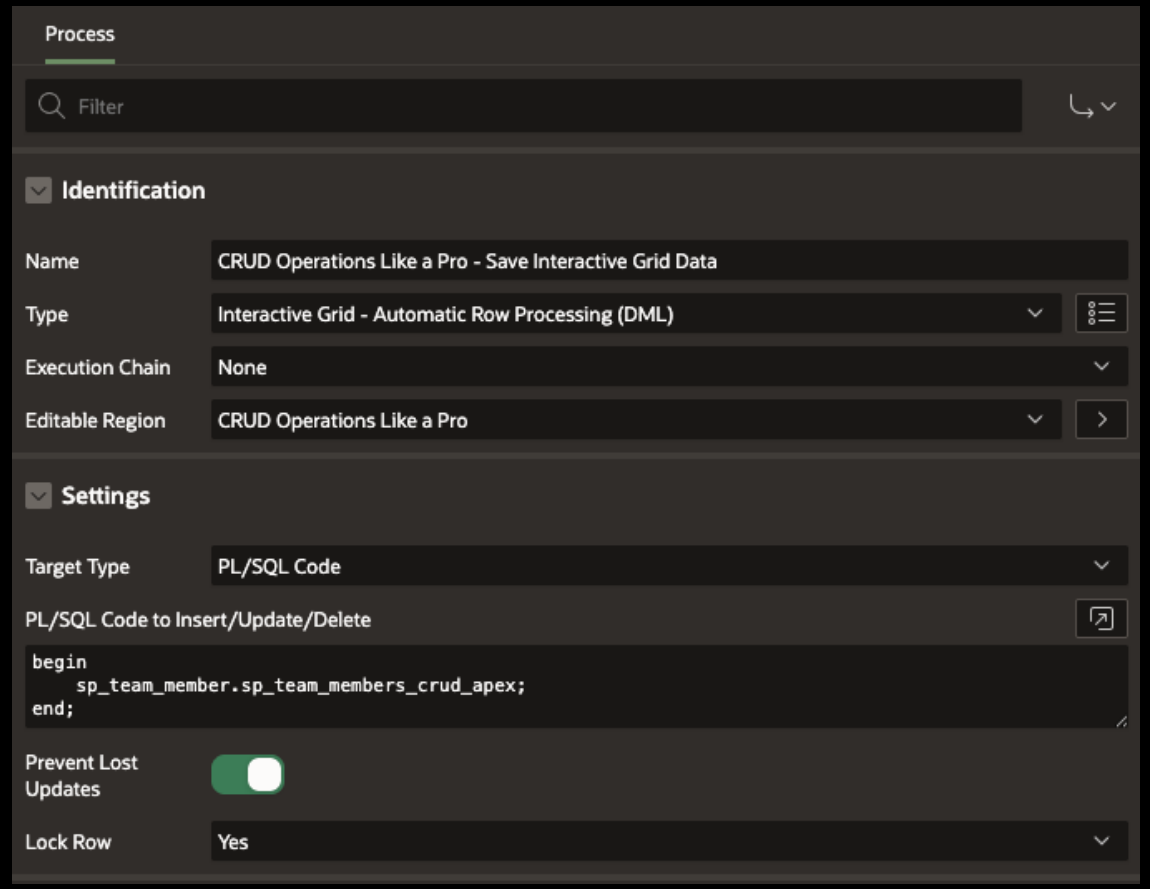
Try to keep the code logic in the APEX app to a minimum and shift it over to a PL/SQL package instead. This way, it's easier to reuse and maintain!

CRUD Operations Like a Pro



Hyand

CRUD Operations Like a Pro



Hyand

CRUD Operations Like a Pro

The procedure, sp_team_members_crud_apex, is called from the APEX Process. For an IG, it's called once for each record you create, update, or delete, depending on how you set the Execution Scope property.

The whole point of this procedure is to grab the values sent to the APEX Process, fill up a PL/SQL record, and then call the right procedure to (C)reate, (U)pdate, or (D)elete the record.

You might think of it as extra work, but it's really helpful for keeping PL/SQL complexity out of our APEX page.

```plsql
create or replace package body SP_TEAM_MEMBER as

    PROCEDURE sp_team_members_crud_apex
    IS

      l_row_status       VARCHAR2(1);
      lr_member_rec      sp_team_members%ROWTYPE;
      l_new_id           sp_team_members.id%TYPE;

    BEGIN

      -- Determine if user (C)reated, (U)pdated or (D)eleted this record.
      l_row_status := V('APEX$ROW_STATUS');

      -- Only need the ID for Update and Delete.
      IF l_row_status IN ('U','D') THEN
        -- Save ID to the PL/SQL Record.
        lr_member_rec.id := V('ID');
      END IF;

      IF l_row_status IN ('C','U') THEN
        -- Populate PL/SQL record with values from Interactive Grid.
        lr_member_rec.first_name := V('FIRST_NAME');
        lr_member_rec.last_name  := V('LAST_NAME');
        lr_member_rec.email      := V('EMAIL');
      END IF;

      -- Call appropriate API to Create, Update or Delete the member.
      CASE l_row_status
        WHEN 'C' THEN
          create_member (pr_member_rec => lr_member_rec, x_id => l_new_id);
          -- Return the new member ID (Primary key) to the field
          --   identified as the Primary Key in the APEX IG / Form Region
          apex_util.set_session_state ('ID', l_new_id);
        WHEN 'U' THEN
          update_member (pr_member_rec => lr_member_rec);
        WHEN 'D' THEN
          delete_member (p_id => lr_member_rec.id);
      END CASE;

    END sp_team_members_crud_apex;

    ------------------------------------------------------------------------------

    ...

end SP_TEAM_MEMBER;
```

Try to keep your business logic in SQL simple and free from any UI logic. This makes your SQL queries easier to read, improves performance, and reduces the risk of cross-site scripting attacks.

Keep your Code clean with Template Directives!

```
select ID,
       FIRST_NAME,
       LAST_NAME,
       EMAIL,
       CASE IS_CURRENT_YN
        WHEN 'Y' THEN 'fa fa-check-circle u-color-5-text'
        WHEN 'N' THEN 'fa fa-minus-circle u-color-9-text'
        ELSE 'fa fa-minus-circle u-color-9-text'
       END IS_CURRENT_YN,
       CREATED,
       CREATED_BY,
       UPDATED,
       UPDATED_BY
  from SP_TEAM_MEMBERS
```

Keep your Code clean with Template Directives!

Identification

Column Name: IS_CURRENT_YN

Type: HTML Expression

Heading

Heading: Is Current

Alignment

Alternative Label

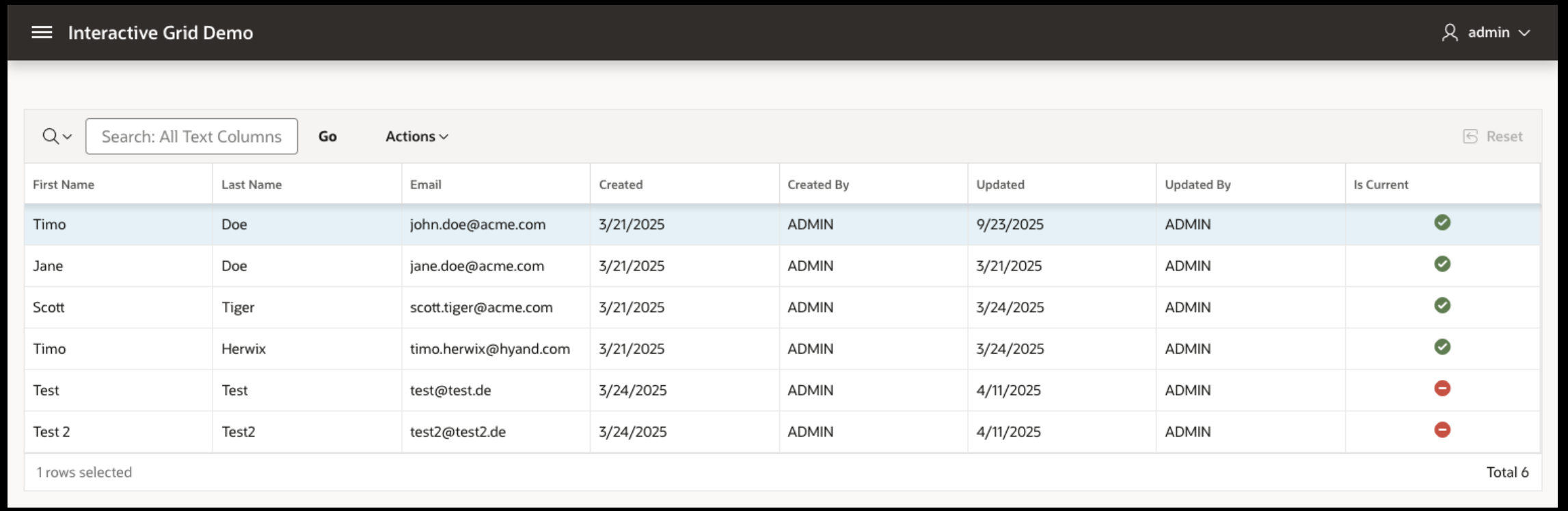
Settings

HTML Expression

```
{case IS_CURRENT_YN/}
  {when Y/}
    <span class="fa fa-check-circle u-color-5-text"></span>
  {when N/}
    <span class="fa fa-minus-circle u-color-9-text"></span>
  {otherwise/}
    <span class="fa fa-minus-circle u-color-9-text"></span>
{endcase/}
```

Hyand

Keep your Code clean with Template Directives!



Don't try to reinvent the wheel!

Use what's already available to keep your code to a minimum.

Using MLE JavaScript modules for APEX validations.

```javascript
// Regular expression pattern for email validation
const emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;

/**
 * Function to validate an email address
 * @param {string} email - The email address to validate
 * @returns {boolean} - True if the email is valid, false otherwise
 */
function validateEmail(email) {
  return emailPattern.test(email);
}

// Example usage
const emailToValidate = 'user@example.com';
if (validateEmail(emailToValidate)) {
  console.log('Valid email address!');
} else {
  console.log('Invalid email address.');
}
```

Using MLE JavaScript modules for APEX validations.

Starting with Oracle Database 21c, developers can now execute JavaScript within the database.

validator.js

npm v13.15.22 | CI no status | codecov 100% | downloads rate limited by upstream service | backers 9 | sponsors 1
License MIT | chat on gitter

A library of string validators and sanitizers.

Strings only

This library validates and sanitizes strings only.

If you're not sure if your input is a string, coerce it using `input + ''`. Passing anything other than a string will result in an error.

Validation

Editable Region	Team Members IG
Type	Function Body (returning Boolean)
Language	JavaScript (MLE)

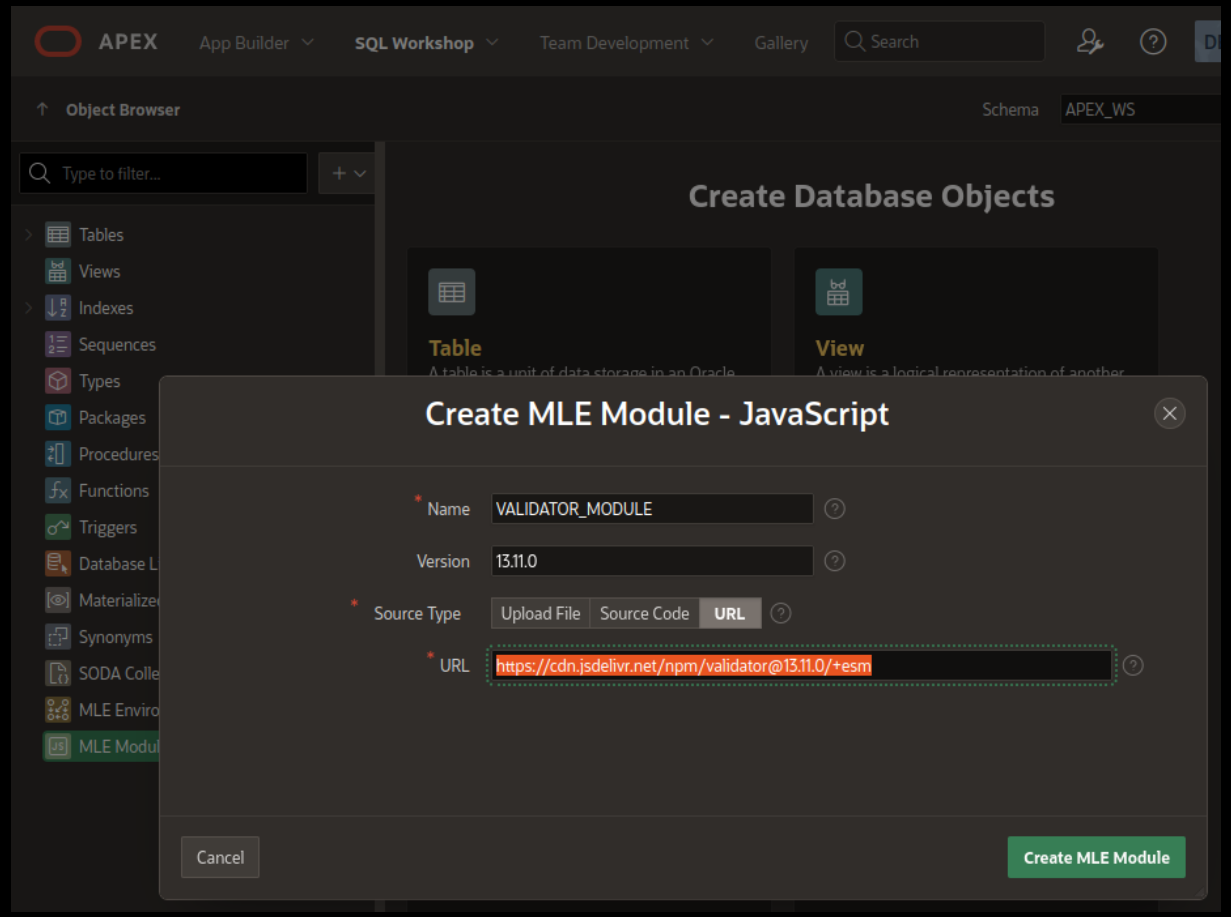
JavaScript Function Body

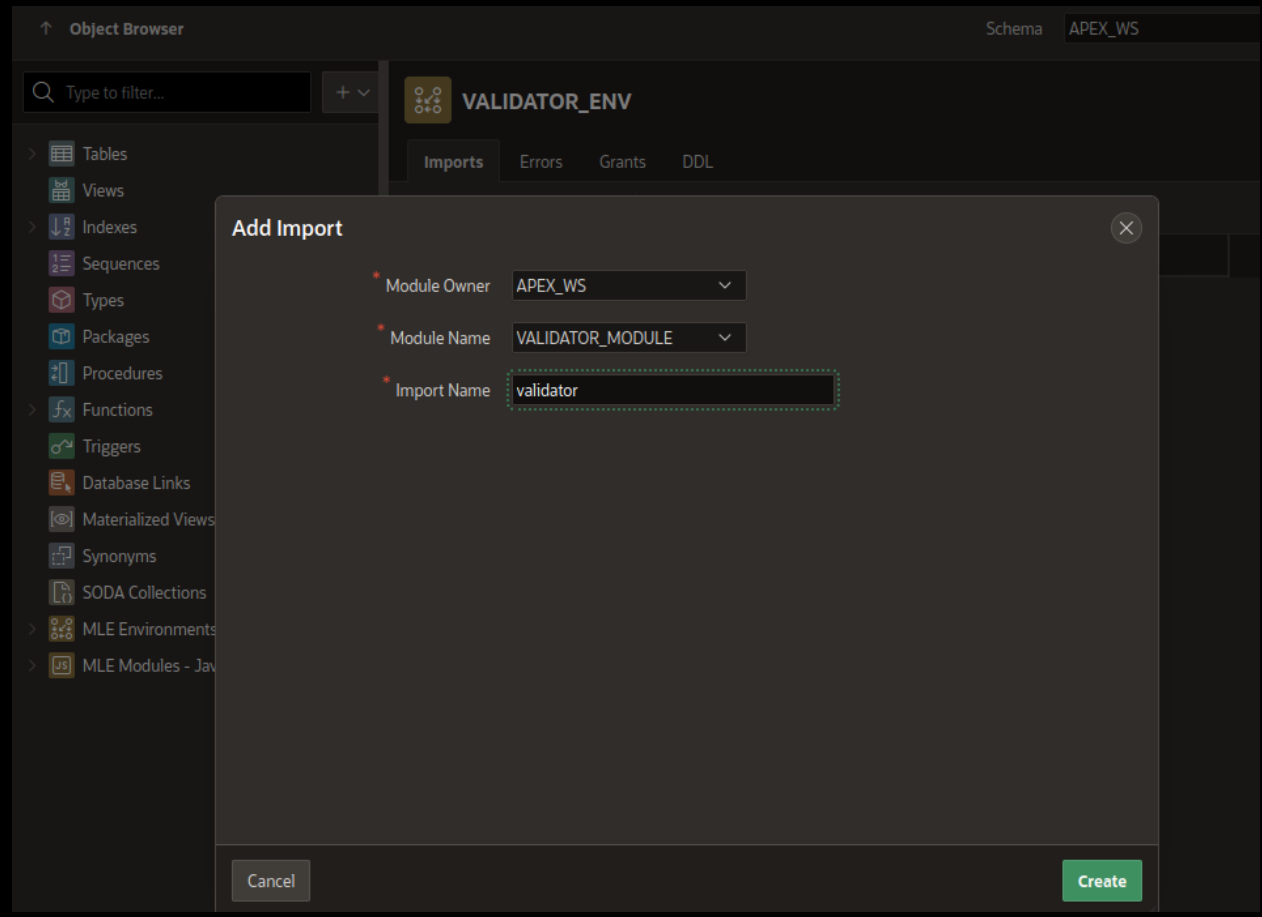
```
const { default: v } = await import ('validator-module');
return v.isEmail( apex.env.EMAIL );
```

Hyand

Using MLE JavaScript modules for APEX validations.

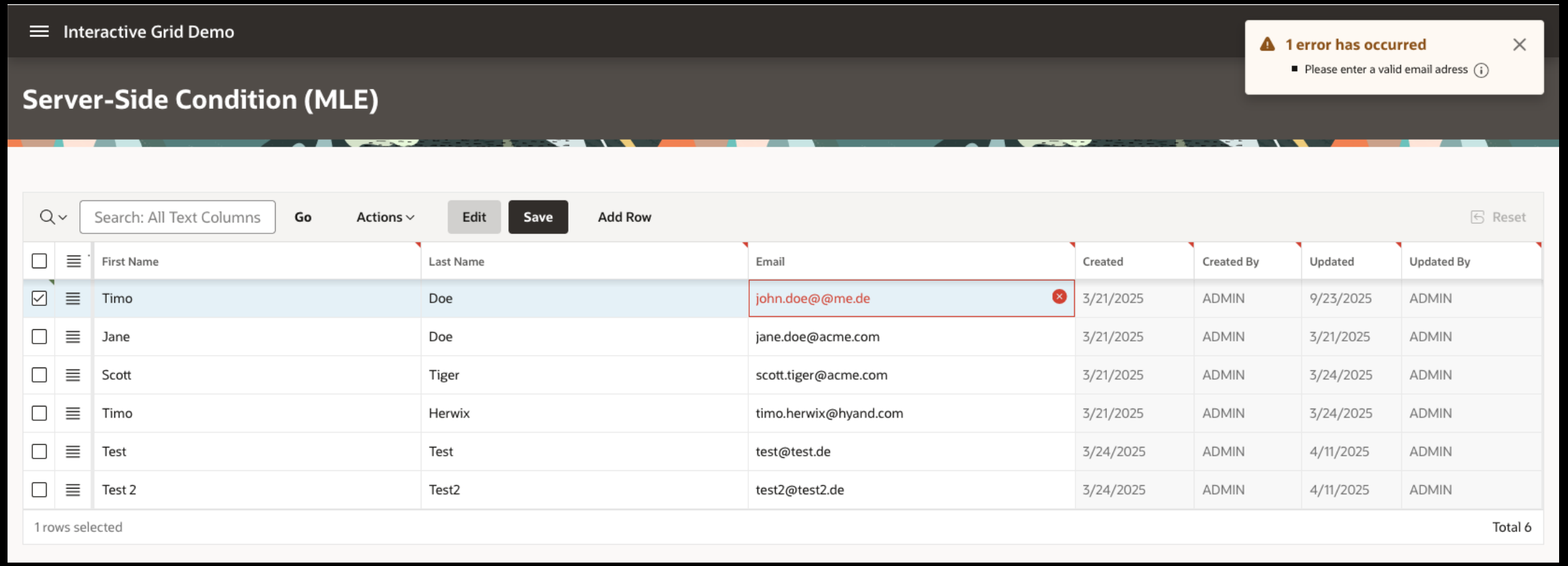
JavaScript modules on NPM can really save you a lot of time and effort with coding.
All you need to do is set up an MLE Module & MLE Environment.





Hyand

Using MLE JavaScript modules for APEX validations.



Hyand

1

Interactive Grid
like a Pro

2

JavaSript
Enhancements

3

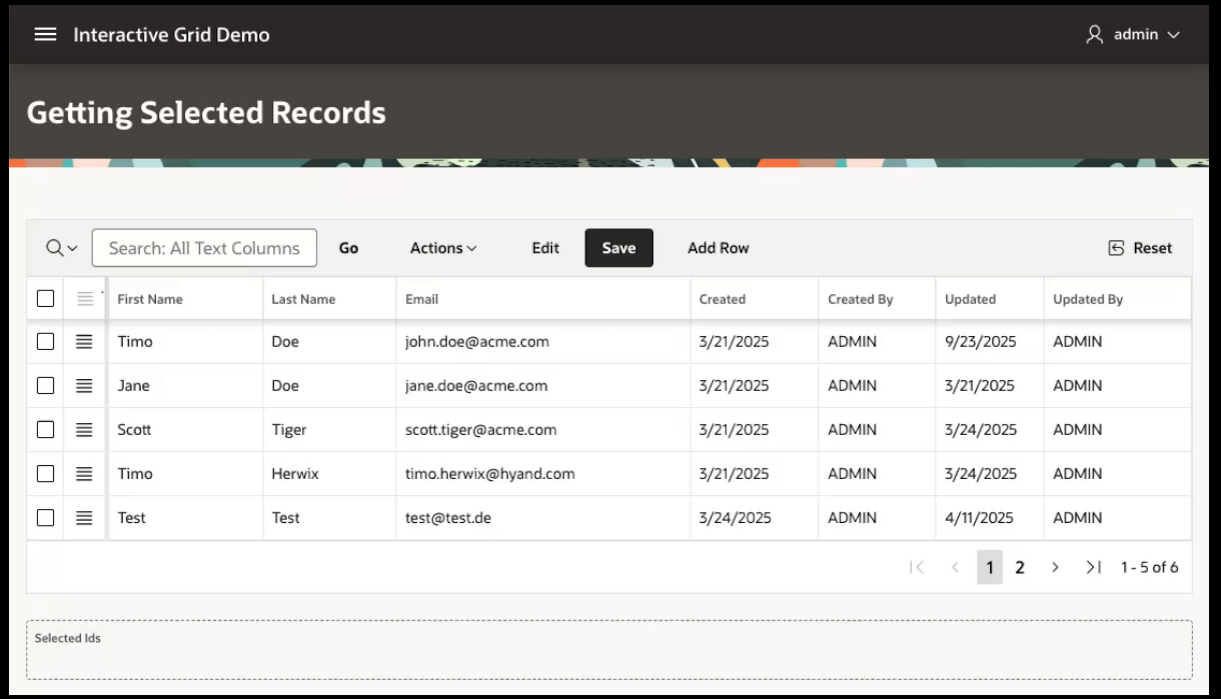
CSS
Enhancements

Hyand

JavaSript Enhancements

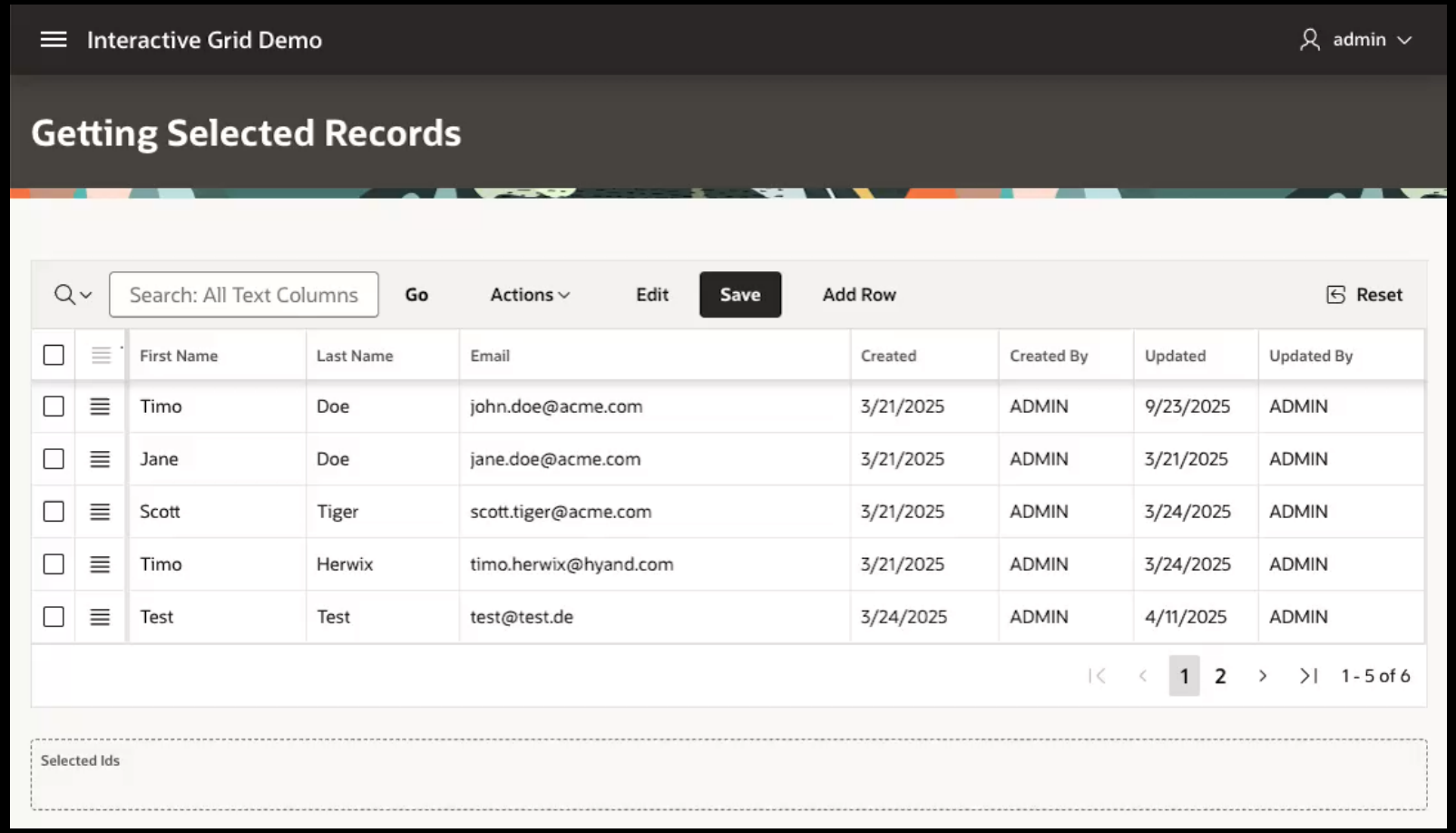
Get the Selected Row Column Values from an Interactive Grid and then process them, like in PL/SQL.

```
function( config ) {
  config.defaultGridViewOptions = {
    selectionStateItem: "P4_SELECTED_IDS",
    multiple: true
  };
  return config;
}
```



JavaSript Enhancements

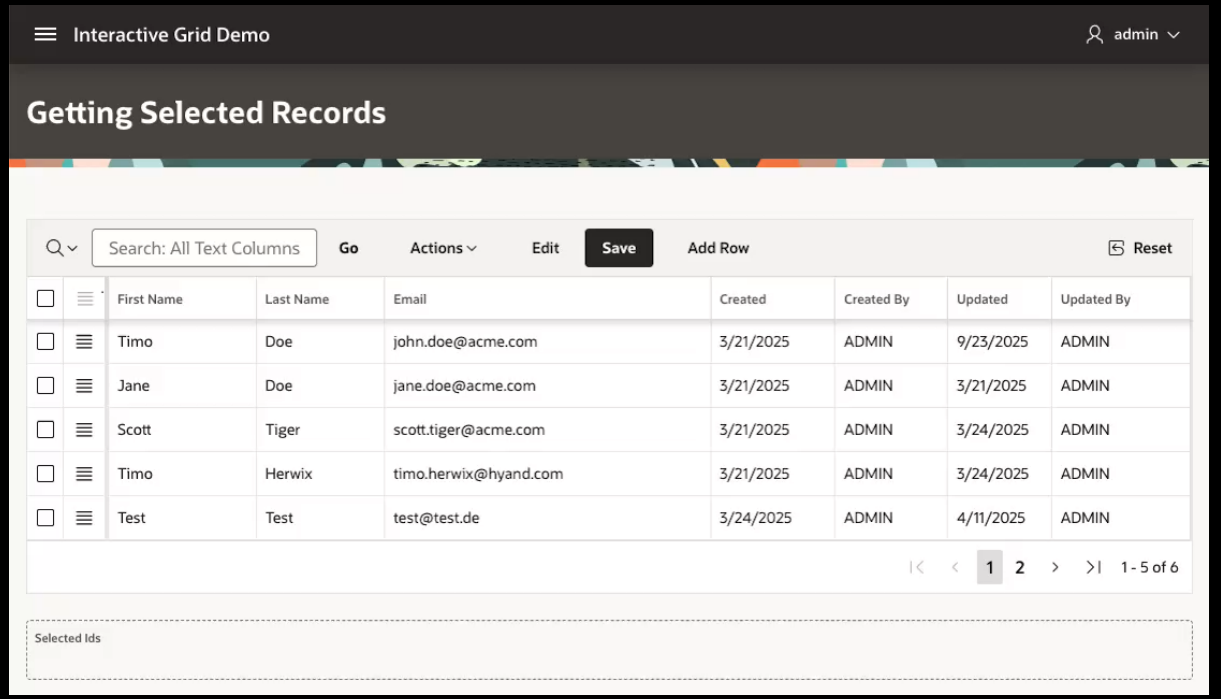
By default, when you switch between pages with pagination set to "Page," any selected records get lost!



JavaScript Enhancements

Easy fix! Just add Persistent Selection 😉

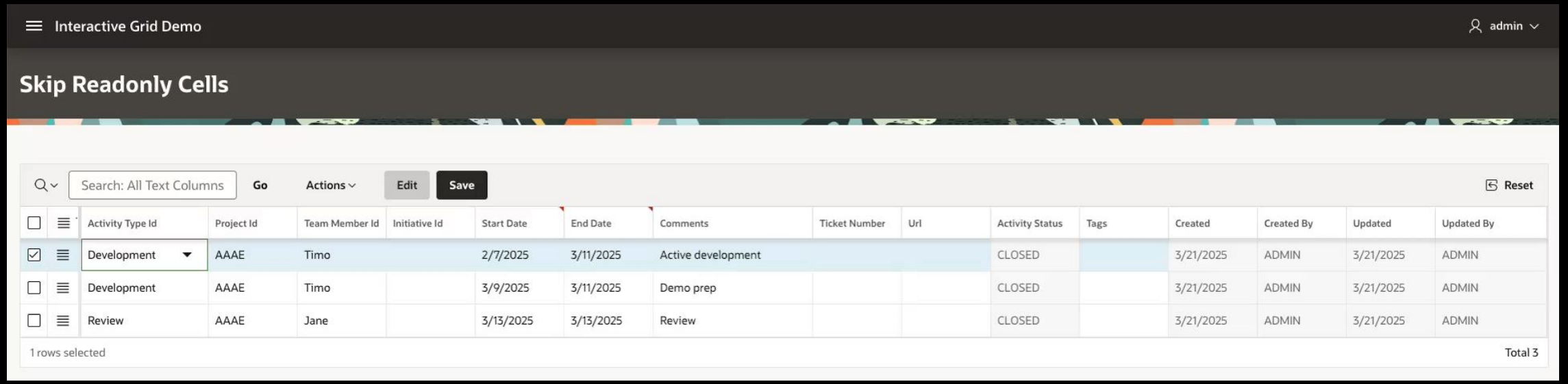
```
function( config ) {
  config.defaultGridViewOptions = {
    selectionStateItem: "P4_SELECTED_IDS",
    multiple: true,
    persistSelection: true
  };
  return config;
}
```



JavaSript Enhancements

Skip the Read-Only Cells

```
function (options) {
    options.defaultGridViewOptions = {
    skipReadonlyCells: true
  };
  return options;
}
```

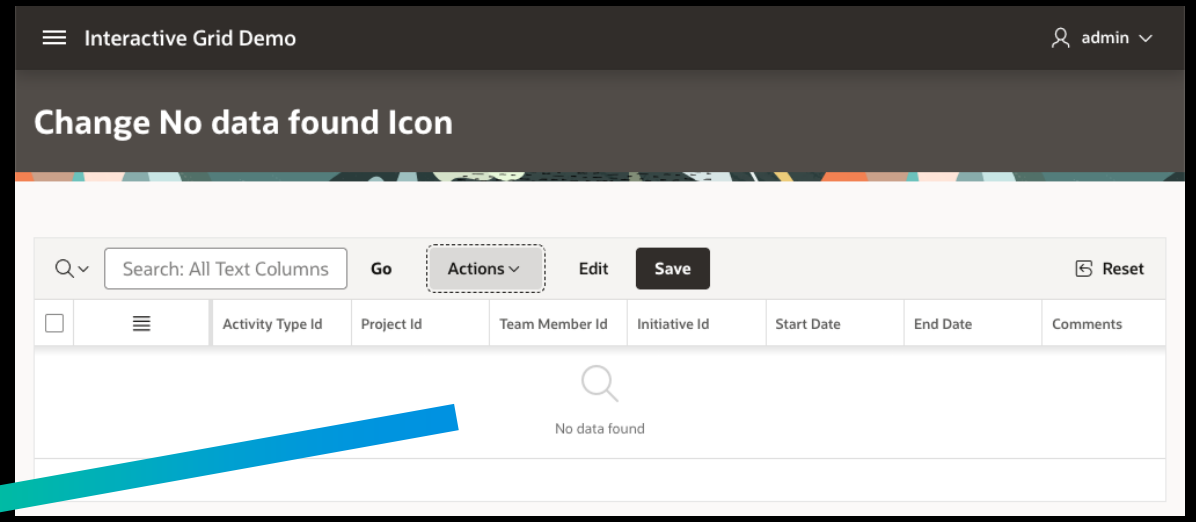


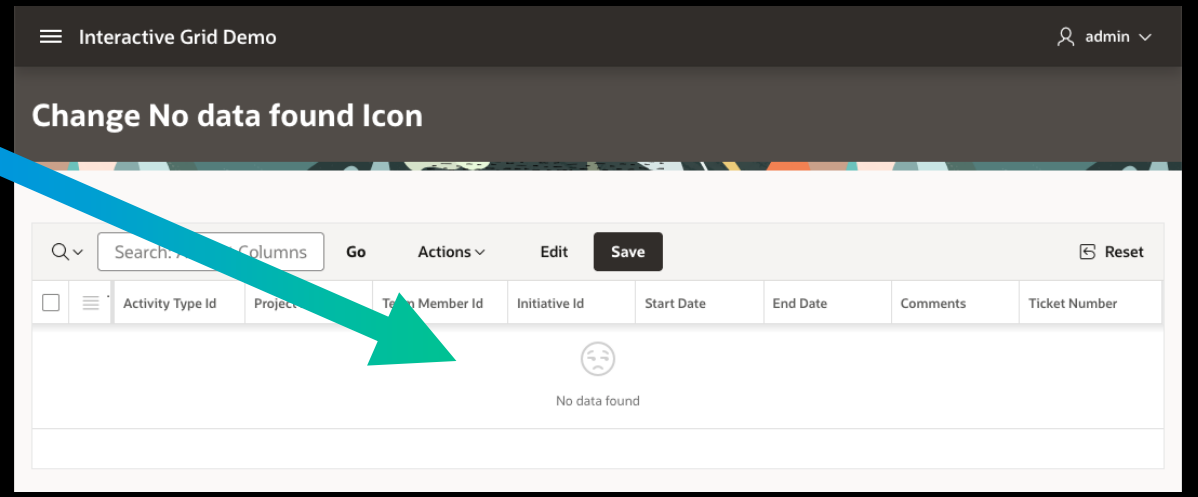
Hyand

JavaSript Enhancements

Change the "No Data Found" Icon

```
function(options) {
    options.defaultGridViewOptions = {
    noDataIcon: "fa fa-emoji-unamused"
  };
  return options;
}
```

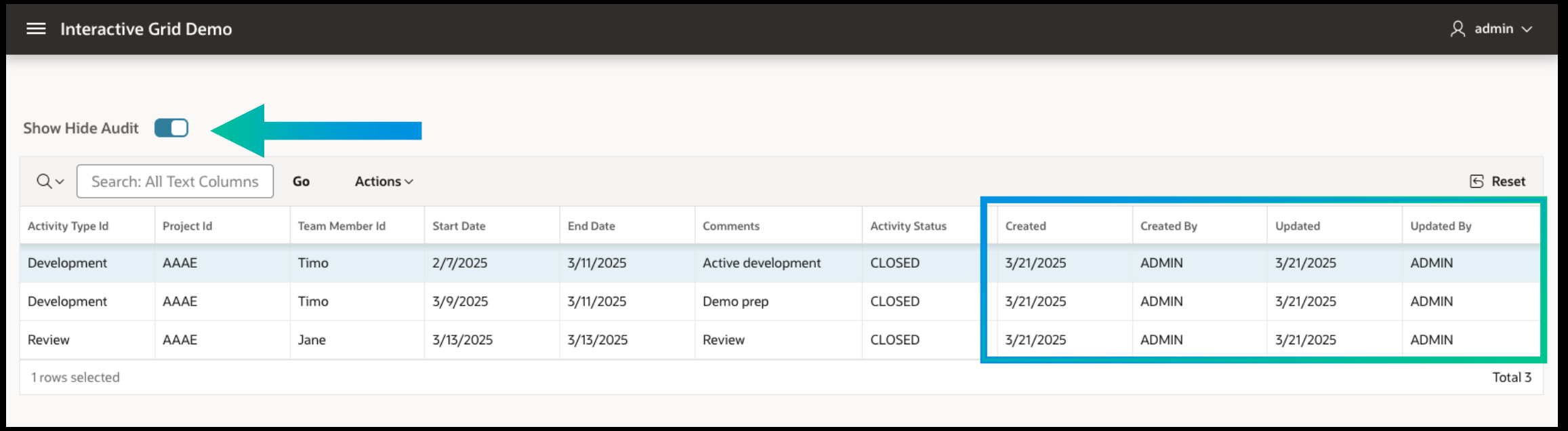




Hyand

JavaSript Enhancements

Show/Hide Colums



Hyand

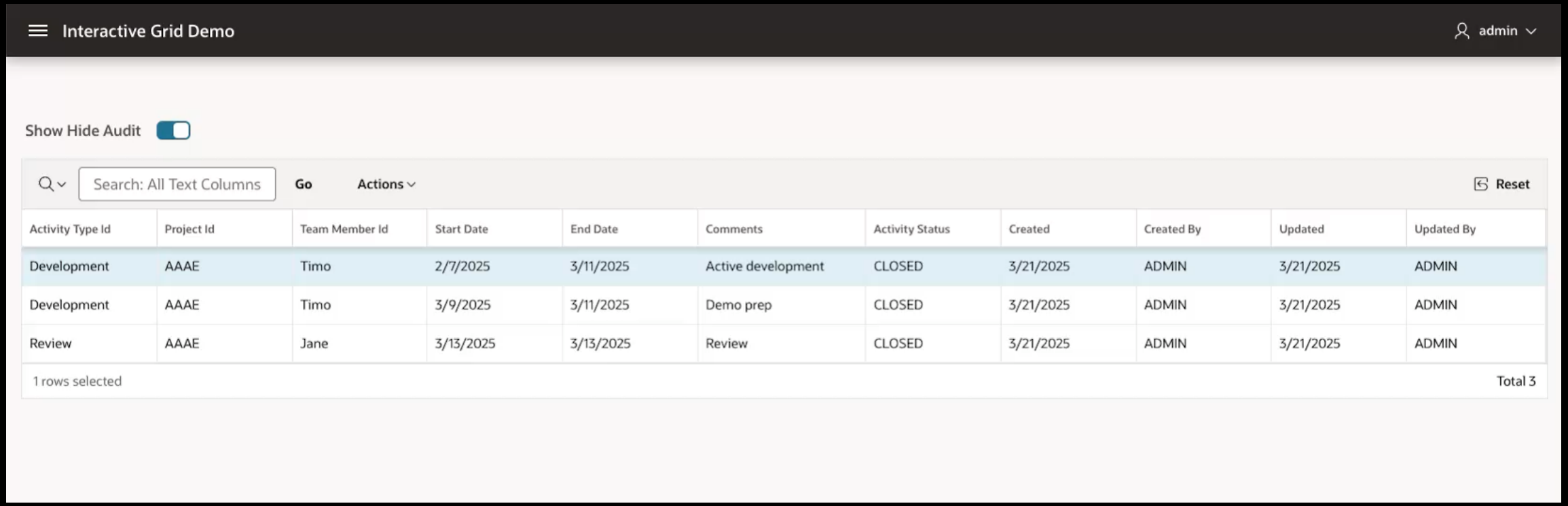
JavaSript Enhancements

Show/Hide Colums

```
let auditColumns = [
    "CREATED",
    "CREATED_BY",
    "UPDATED",
    "UPDATED_BY"
],
view = apex.region("p15_ig").widget().interactiveGrid("getCurrentView").view$;

auditColumns.forEach( function (columnName) {
  view.grid(
    (apex. items.P15_SHOW_HIDE_AUDIT.value === "y" ? "showColumn" : "hideColumn"),
    columnName
  );
});
```

JavaSript Enhancements

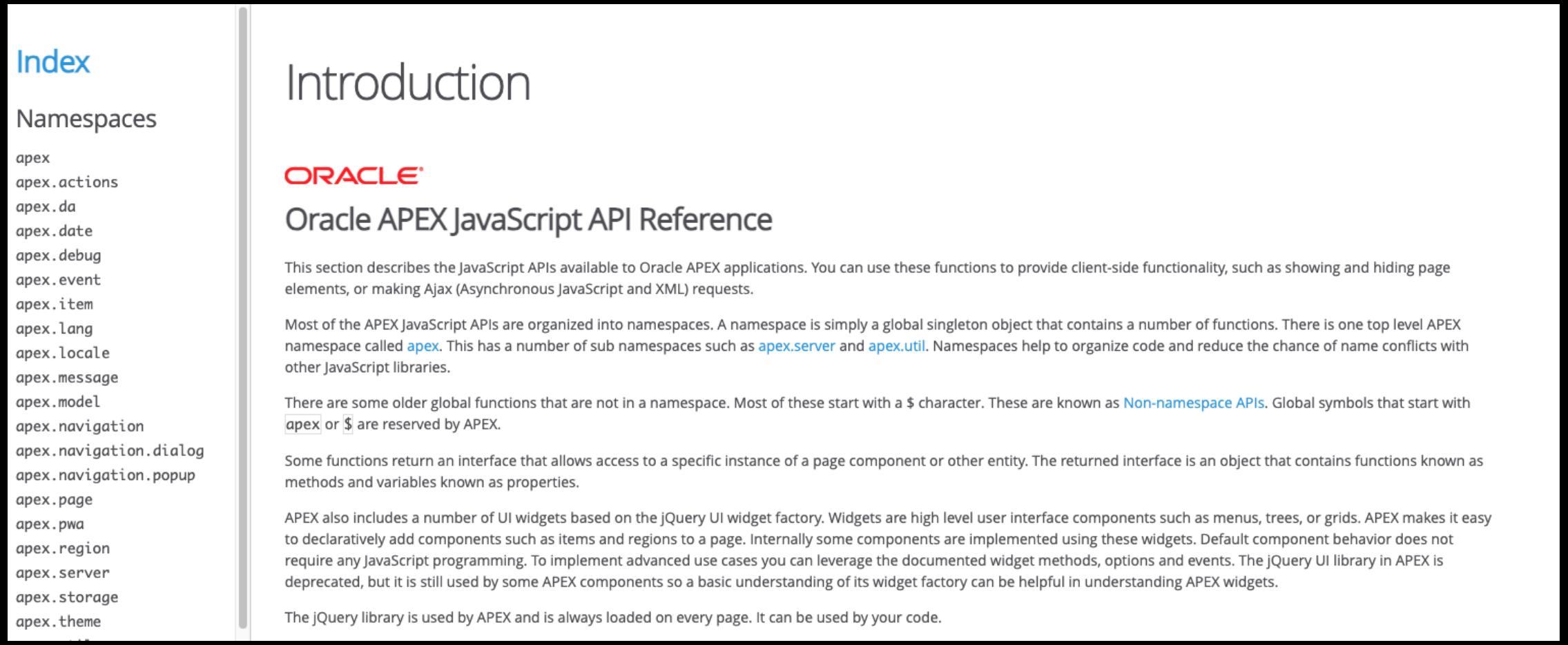
Show/Hide Colums



Hyand

JavaScript Enhancements

JavaScript API Documentation (https://apex.oracle.com/jsapi)



Index

Namespaces

apex
apex.actions
apex.da
apex.date
apex.debug
apex.event
apex.item
apex.lang
apex.locale
apex.message
apex.model
apex.navigation
apex.navigation.dialog
apex.navigation.popup
apex.page
apex.pwa
apex.region
apex.server
apex.storage
apex.theme

Introduction

ORACLE

Oracle APEX JavaScript API Reference

This section describes the JavaScript APIs available to Oracle APEX 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 APEX 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 APEX 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 that allows access to a specific instance of a page component or other entity. The returned interface is an object that contains functions known as methods and variables known as properties.

APEX also includes a number of UI widgets based on the jQuery UI widget factory. Widgets are high level user interface components such as menus, trees, or grids. APEX makes it easy to declaratively add components such as items and regions to a page. Internally some components are implemented using these widgets. Default component behavior does not require any JavaScript programming. To implement advanced use cases you can leverage the documented widget methods, options and events. The jQuery UI library in APEX is deprecated, but it is still used by some APEX components so a basic understanding of its widget factory can be helpful in understanding APEX widgets.

The jQuery library is used by APEX and is always loaded on every page. It can be used by your code.

1

Interactive Grid
like a Pro

2

JavaSript
Enhancements

3

CSS
Enhancements

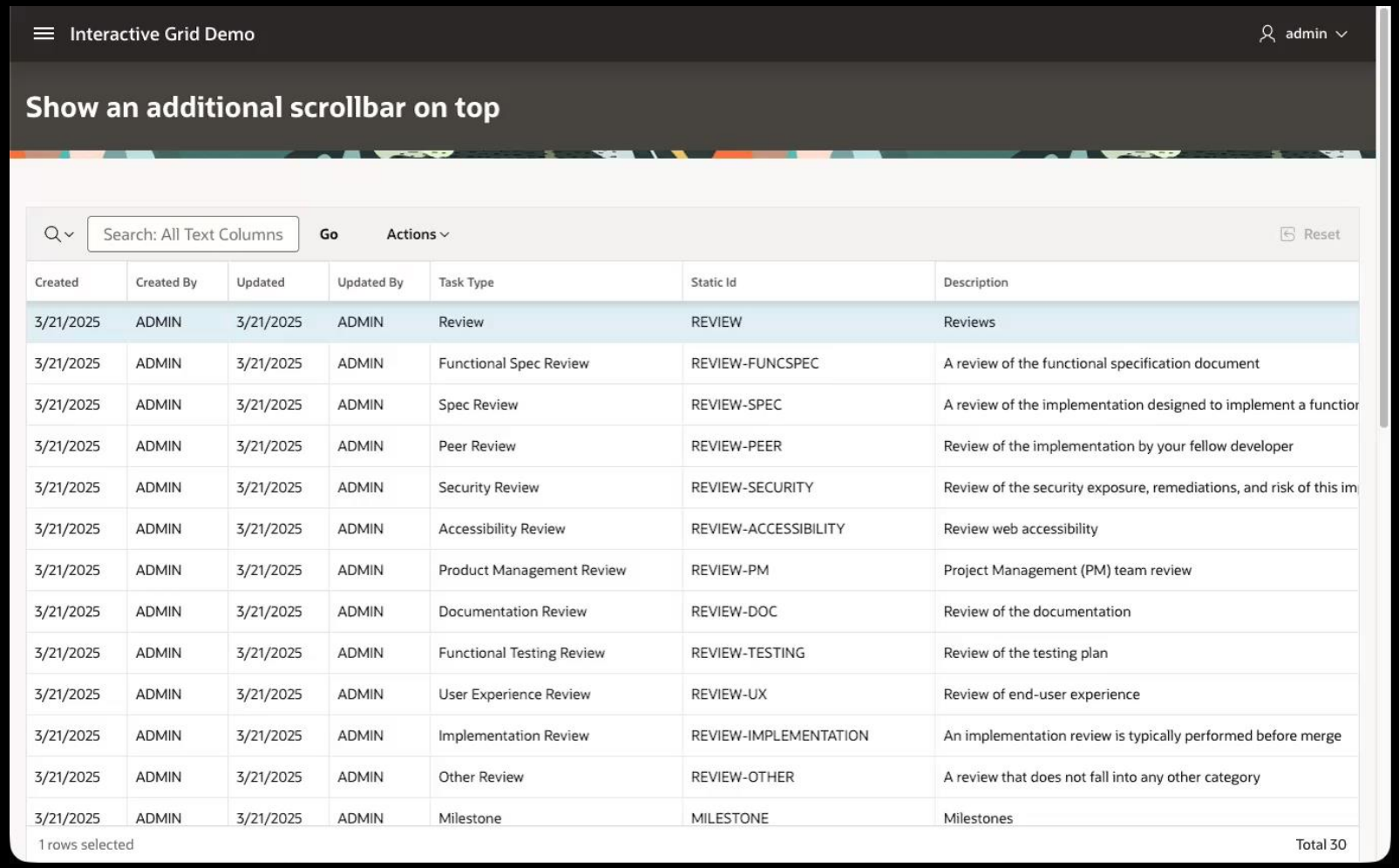
Hyand

Missing Scrollbar?

Sometimes a grid has tons of columns and ends up wider than the screen. This makes it hard to see the columns on the right. You have to scroll down to move the horizontal bar, then scroll back up. Kind of annoying, right?

CSS Enhancements

Missing Scrollbar?

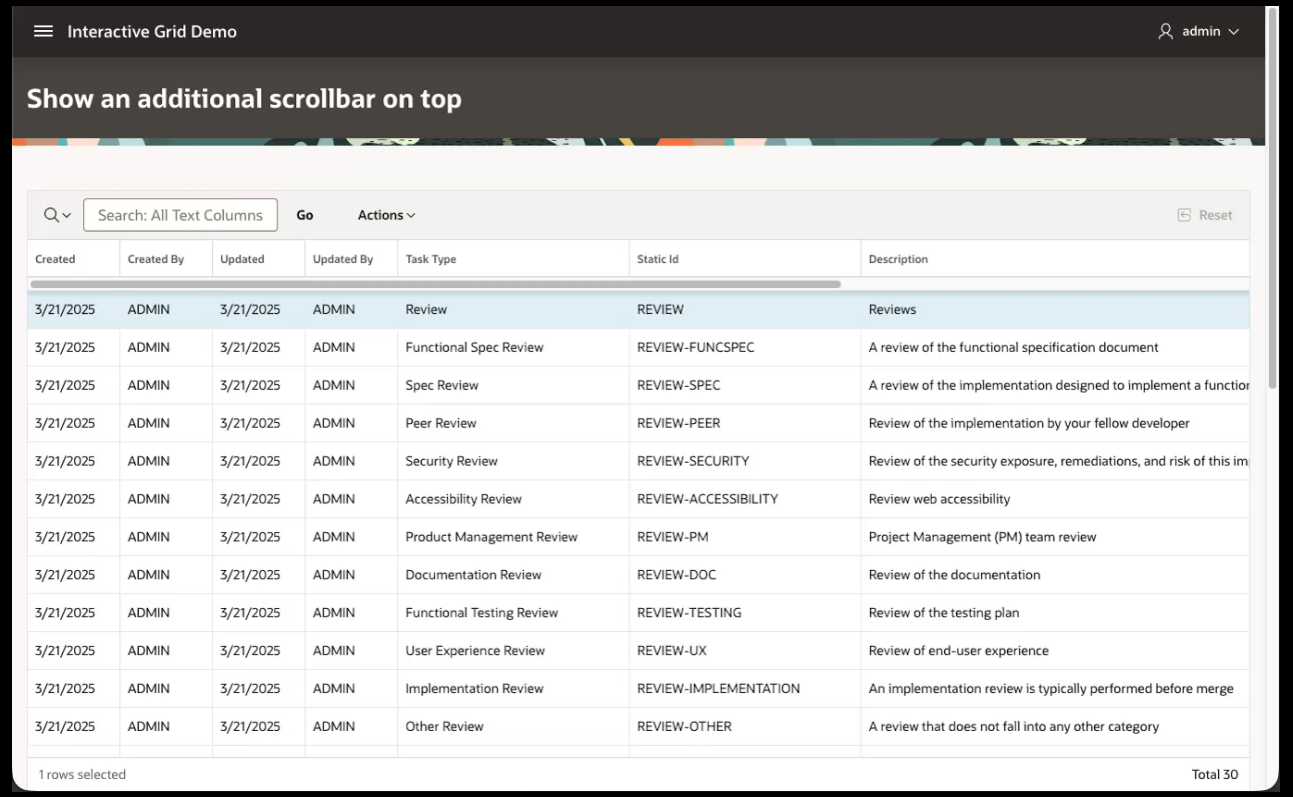


Hyand

CSS Enhancements

Show an additional scrollbar on top!

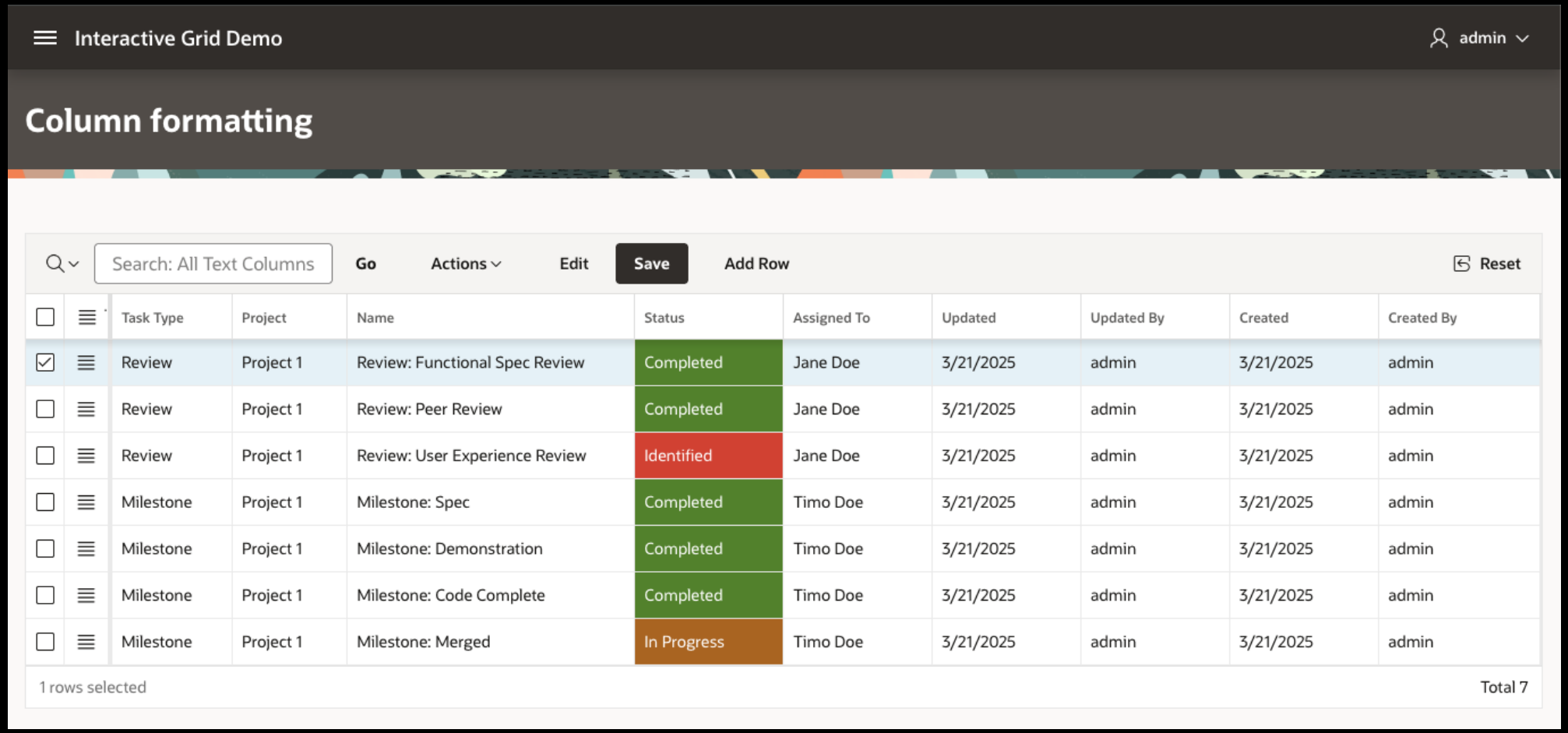
```css
#my_ig .a-GV-w-hdr{
    overflow-x: auto !important;
}
```



Hyand

CSS Enhancements

Column formatting using :has()



CSS Enhancements

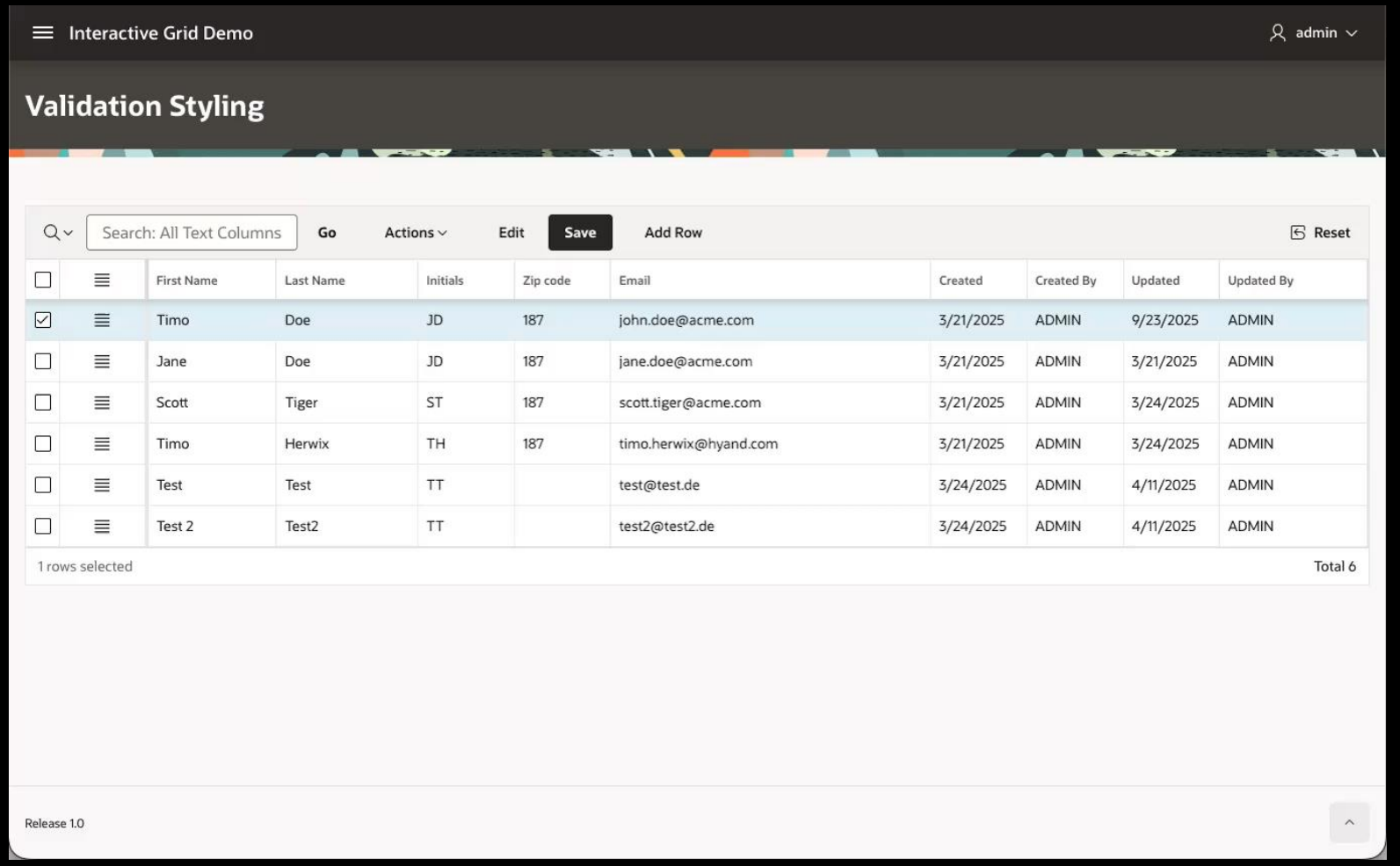
Define an HTML expression using a template directive and add a bit of CSS for style.


```css
td:has(.status-completed) {
    --a-gv-row-hover-background-color: var(--a-palette-success);
    background-color: var(--a-palette-success);
    color: var(--a-palette-success-contrast);
}

td:has(.status-identified) {
    --a-gv-row-hover-background-color: var(--a-palette-danger) ;
    background-color: var(--a-palette-danger);
    color: var(--a-palette-danger-contrast);
}

td:has(.status-in_progress) {
    --a-gv-row-hover-background-color: var(--a-palette-warning);
    background-color: var(--a-palette-warning);
    color: var(--a-palette-warning-contrast);
}

td:has(.status-other) {
    --a-gv-row-hover-background-color: var(--a-palette-color-14);
    background-color: var(--a-palette-color-14);
    color: var(--a-palette-color-14-contrast);
}
```

Hyand

CSS Enhancements

Client-Side Validation Styling



Hyand

CSS Enhancements

This CSS styles table cells in a grid view, highlighting error and changed states.

Error state cells:

- Red border and text color
- Shake animation to draw attention

Changed state cells:

- Green border and text color
- No animation

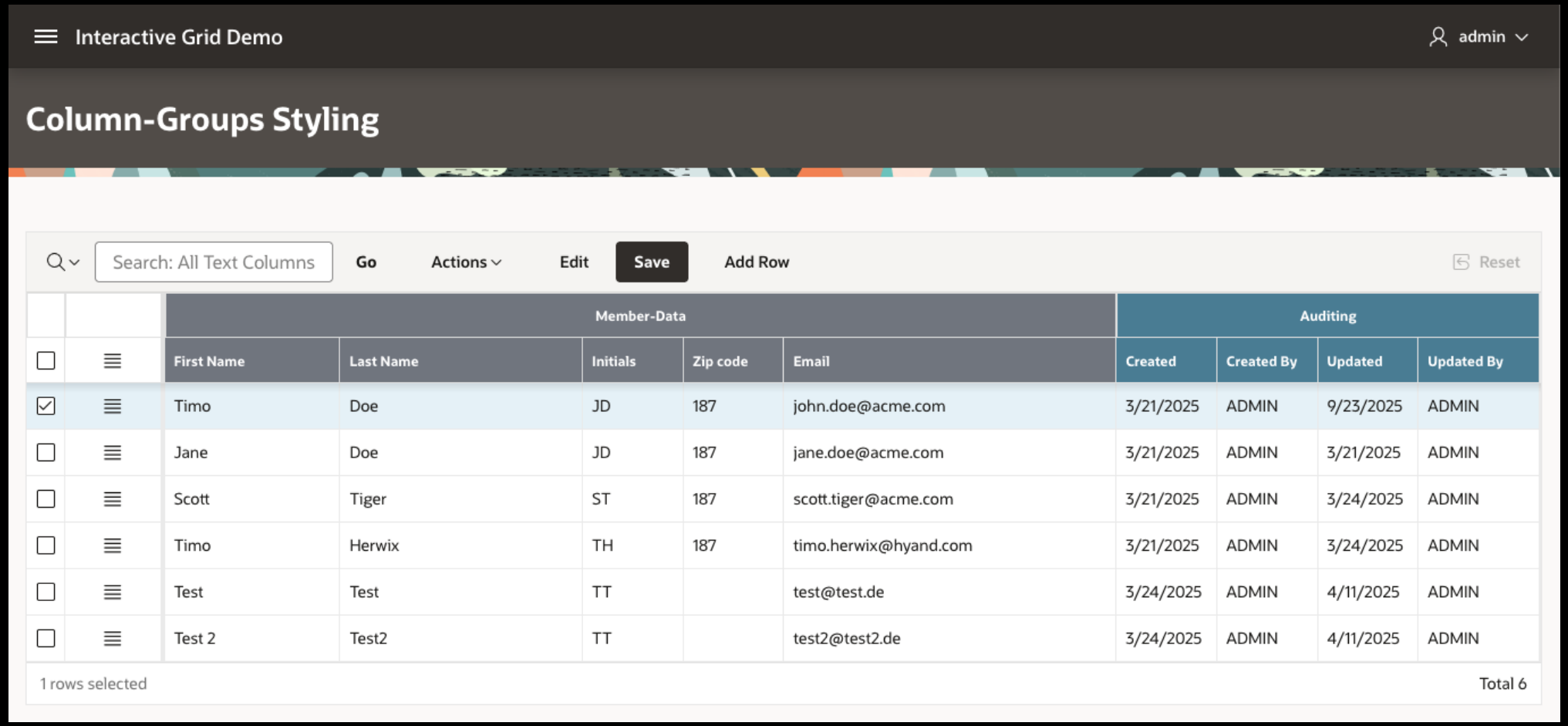
Shake animation:

- Creates a horizontal oscillation effect
- Enhances visibility for invalid entries

```css
/* Target Item Container with invalid page item */
td.a-GV-cell.is-error:not(.is-active), td.a-GV-cell.is-error:not(.is-active):before{
    /* item border color to red */
    border-color: var(--a-palette-danger);
    /* item value text color to red */
    color: var(--a-palette-danger);
    /* shake animation */
    animation: shake-horizontal 0.5s cubic-bezier(0.455, 0.030, 0.515, 0.955) both;
}

/* Target Item Container with invalid page item */
td.a-GV-cell.is-changed:not(.is-active), td.a-GV-cell.is-changed:not(.is-active):before{
    /* item border color to green */
    border-color: var(--a-palette-success) var(--a-palette-success) transparent transparent;
    /* item value text color to green */
    color: var(--a-palette-success);
}

@keyframes shake-horizontal {
  0% { transform: translateX(0) }
  25% { transform: translateX(5px) }
  50% { transform: translateX(-5px) }
  75% { transform: translateX(5px) }
  100% { transform: translateX(0) }
}
```

CSS Enhancements

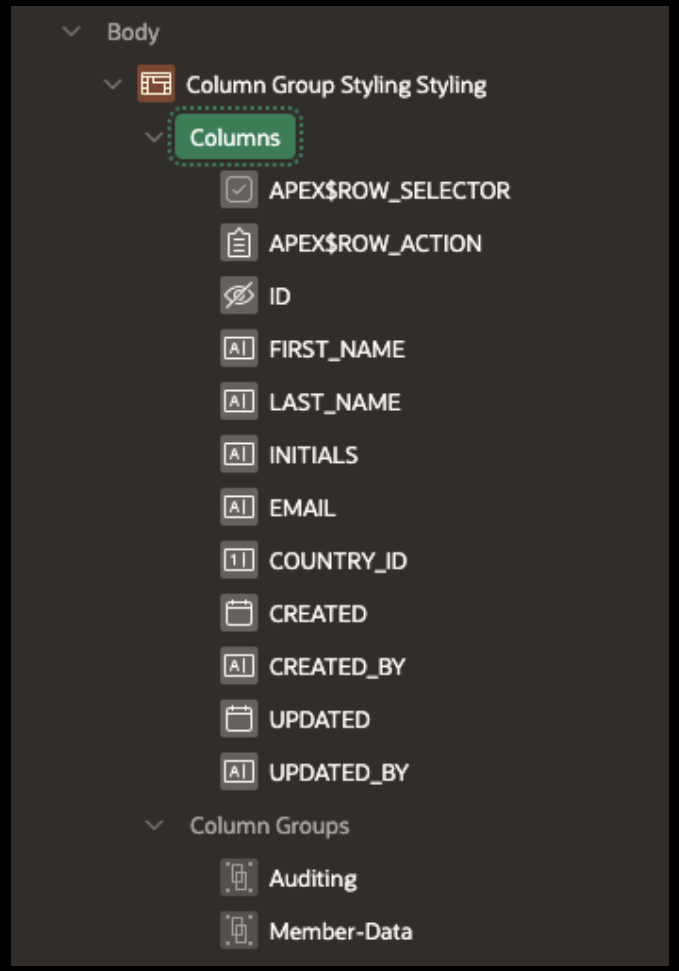
Column-Groups Styling



Hyand

CSS Enhancements

Column-Groups Styling



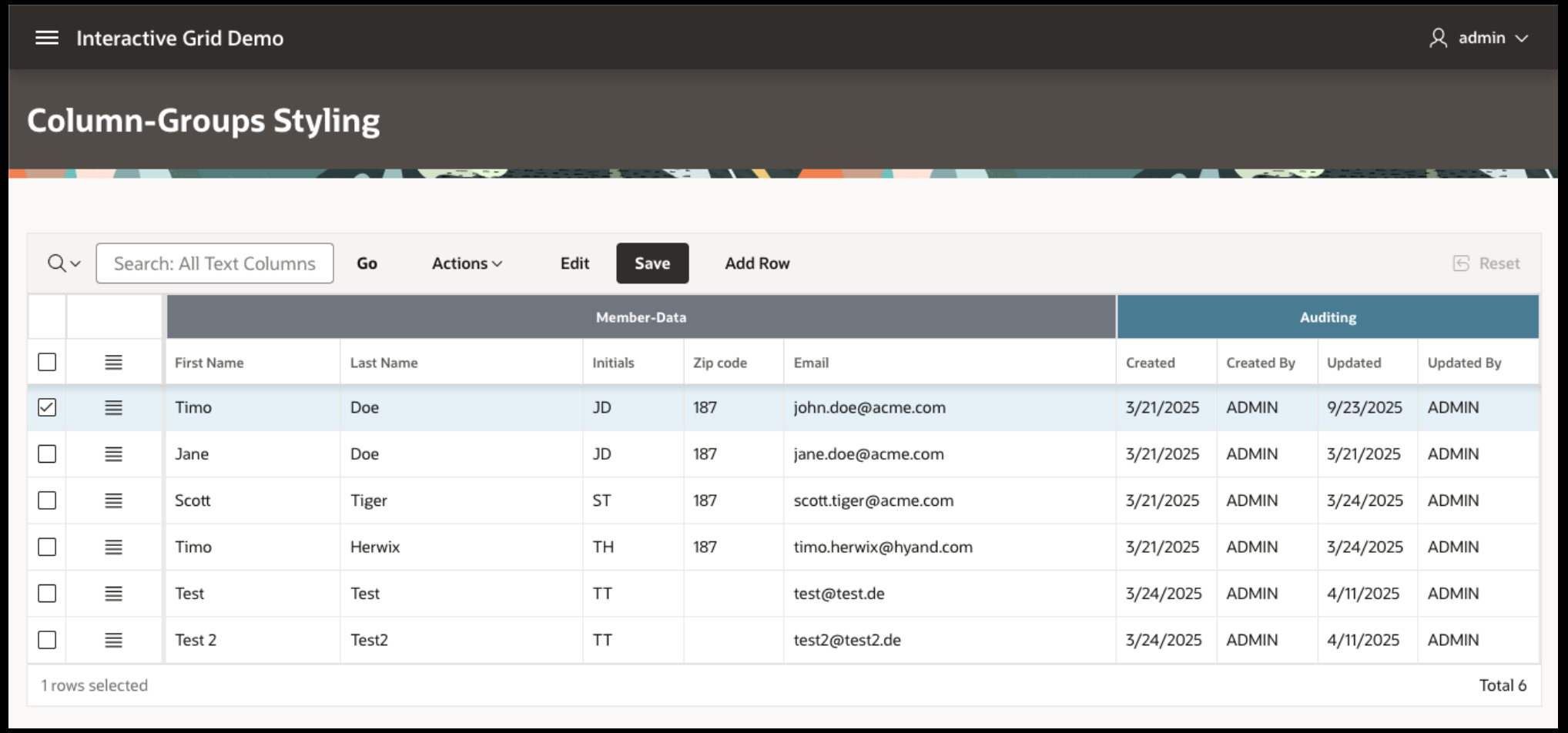

```css
#my_ig .a-GV-headerGroup[data-idx="1"] {
    background-color: var(--a-palette-color-14);
    color: white;
    font-weight: bold;
}

#my_ig .a-GV-headerGroup[data-idx="2"] {
    background-color: var(--a-palette-color-1);
    color: white;
    font-weight: bold;
}
```

```
<th role="columnheader" class="a-GV-headerGroup" aria-roledescription="Group header" colspan="5" data-idx="1" tabindex="0">…
</th>
<th role="columnheader" class="a-GV-headerGroup" aria-roledescription="Group header" colspan="4" data-idx="2" tabindex="-1">…
</th>
</tr>
```

Hyand

CSS Enhancements

Column-Groups Styling



Hyand

CSS Enhancements

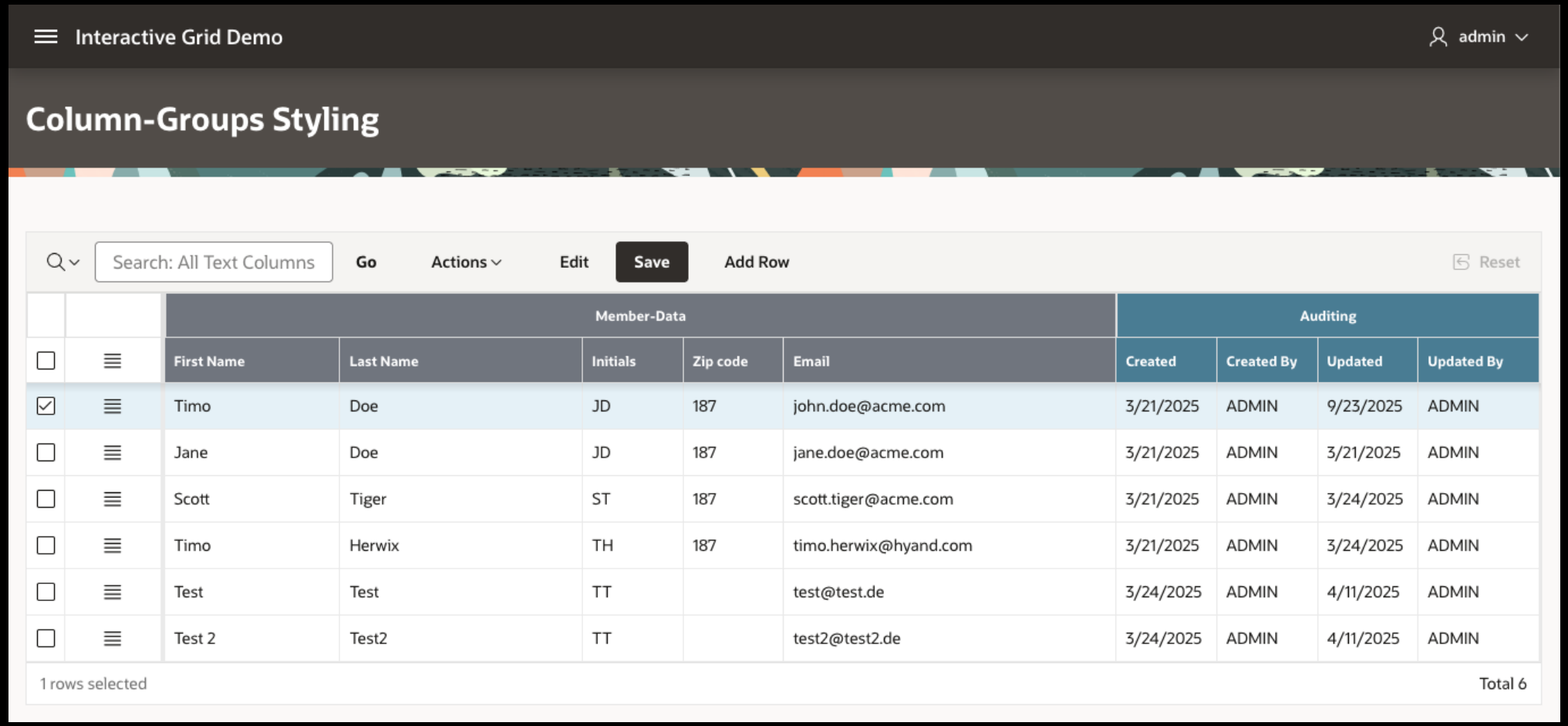
Indeed, we want to add a background color to the column headings too. But there's no declarative CSS attribute for the column header cell! So, we'll need to add a bit of JavaScript, and then we can add a CSS rule for the column headings.

```
function(config) {
    config.defaultGridColumnOptions = {
        headingCssClasses: "master-header"
    }
    return config;
}

function(config) {
    config.defaultGridColumnOptions = {
        headingCssClasses: "audit-header"
    }
    return config;
}
```

```
#my_ig .master-header {
    background-color: var(--a-palette-color-14);
    color: white;
    font-weight: bold;
}

#my_ig .audit-header {
    background-color: var(--a-palette-color-1);
    color: white;
    font-weight: bold;
}
```

CSS Enhancements

Column-Groups Styling



Hyand

Wrap-Up

Hyand

Wrap-Up

Interactive Grid Like a Pro: Best Practices and Enhancements

Code Structure:	**Move your business logic into PL/SQL packages to make it easier to reuse and maintain, and keep your SQL clean from any UI logic.**
JavaScript Enhancements:	**Try to use the APEX JavaScript API if you want to dive deeper into customization.**
CSS Styling:	**To make the UI/UX better for your users, use CSS and pay attention to how things look (like color, font, and layout) and how the user feels when they use it.**
Clean Code:	**Don't try to reinvent anything that's already working well, and be sure to put any extra code in the right spots.**

5 things to keep in mind

Only use it when you really need to

Makes things smoother for users

Think about security

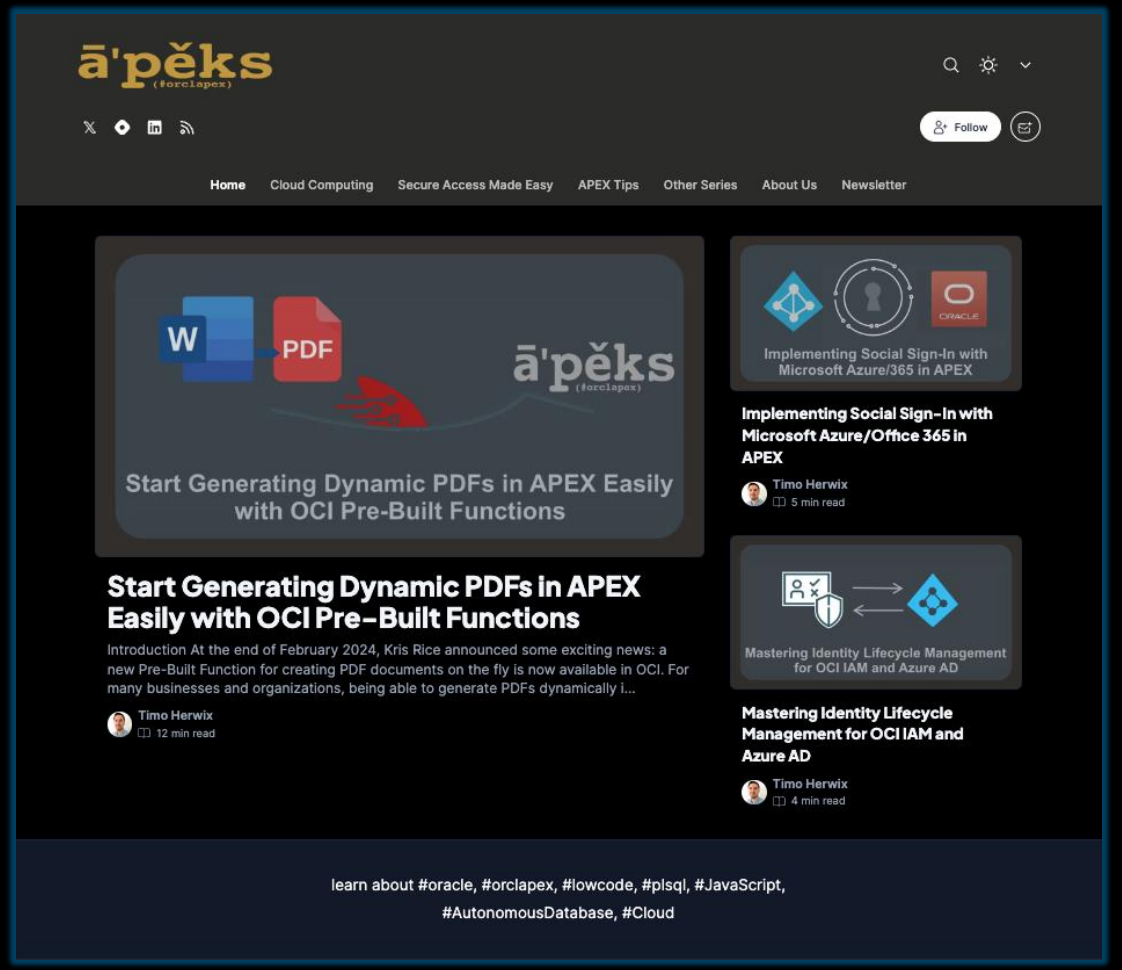
Try not to overwhelm your users

Make sure to understand everything you do

Hyand

Blog

Scan me!



Hyand

Are you interested?

Timo Herwix
Senior Consultant

Telefon: +49 2102 30 961-0
Mobil: +49 176 20185455
Mail: timo.herwix@hyand.com

Timo Herwix

@Therwix

tm-apex.hashnode.dev