



# LAOUC Community Tour 2023

## JavaScript for PL/SQL Developers

**Lino Schildenfeld**

@LinoSchild  
lschilde.blogspot.com



# AUSOUG APEX News



<https://www.ausoug.org.au/news/apex-special-series-2023/>



## Mentor and Speaker Hub

- Our goal is to *connect* speakers with mentors to assist in *preparing* technical sessions and *improving* presentation skills

Interested? Read more and get in touch

<https://mashprogram.wordpress.com>

# My story



AUSOUG

- AUSOUG APEX webinars
- NZ APEX meetup organizer

@LinoSchilde



FOEX

- APEX World Member of the Month
- Conference speaker



APEX RD  
research & development

- Blogger

Oracle ACE  
Pro

# Motivation - Modern apps

- User-friendly
- Engaging apps
- User have expectations
- Provide immediate response and feedback
- Put it another way, users expect **web pages to be very interactive**
- They need to work on any device
- This applies to LOW CODE frameworks too

# **HTML**

# HTML DOM Basics

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>

  <body>
    <h1>A Heading</h1>
  </body>
</html>
```

```
<!DOCTYPE html>
<html class="no-js page-1000 app-4500" lang="en"> event scroll
  > <head> ...
  > <body class=" apex-theme-standard has-envBanner--left"> event overflow
    <!--
      [if lte IE 9]><div id="outdated-browser">You are using an outdated
      web browser. For a list of supported browsers, please reference the
      Oracle Application Express Installation Guide.</div><![endif]
    -->
    <noscript>You must run this product with JavaScript enabled.
    </noscript>
  > <div class="a-EnvBanner a-EnvBanner--accent-14" role="region" aria-
    label="DEMO APEX.ORACLE.COM"> ... </div>
```

```
window = {
  document: {
    head: {
      ...
    },
    body: {
      ...
    },
  },
};
```

- To work with an HTML page in JavaScript, we create a representation of this structure - Document Object Model (DOM).
- **DOM** is a JavaScript object
- It lets us access and manipulate the elements in an HTML page
- APEX is no exception

# DOM manipulation

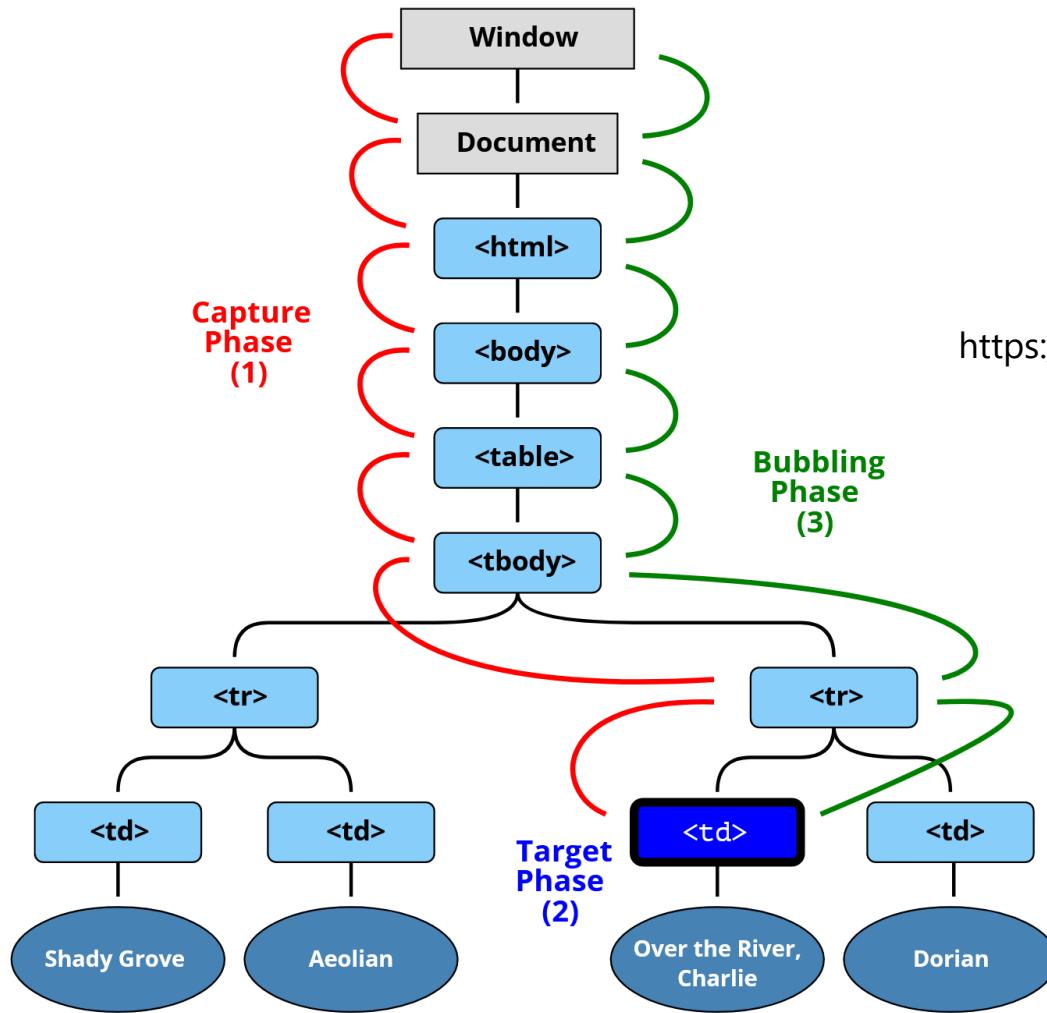
- Add/remove classes
- Modify attributes
- Element manipulation
- Remove or add elements
- Change css styles

```
<html>
  <head>
    <script>
      // run this function when the document is loaded
      window.onload = function() {

        // create a couple of elements in an otherwise empty HTML page
        const heading = document.createElement("h1");
        const heading_text = document.createTextNode("Big Head!");
        heading.appendChild(heading_text);
        document.body.appendChild(heading);

      }
    </script>
  </head>
  <body>
    </body>
</html>
```

# APEX page too



<https://javascript.info/bubbling-and-capturing>

# Common terms

Object	Description
window	Global object (browser window)
document	part of the DOM API
console	Our dbms_output
JSON	Methods provided for JSON manipulation
setTimeout	or timing the actions
.....	.....
apex.items	Some APEX stuff
apex.submit( 'DELETE' );	.....

# jQuery web API

- Makes it more easy to manipulate a page of HTML
- Library we include on the page
- Fast, small, and feature-rich JavaScript library
- It is included in APEX too
- Wrappers

- Simplified APIs

addClass,removeClass,attr,removeAttr, val, html, text,  
append,prepend,remove,empty

```
<input id="input-test" type="input" name="input">
<script>
  $('#input-test').on('change', function() {
    console.log('it changed!');
  });
</script>
```

# SELECTORS

- Like:

`$ ('#id')` - ID selector

`$ ('.class')` - class selector

`$ ('td')` - element selector

`$ ("input [name~='man']")` - attribute selector

...

...

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>attributeContains demo</title>
6   <script src="https://code.jquery.com/jquery-3.5.0.js"></script>
7 </head>
8 <body>
9
10 <input name="man-news">
11 <input name="milkman">
12 <input name="letterman2">
13 <input name="newmilk">
14
15 </body>
16 </html>
```

# DOM vs JQuery

Example:

## ID selector

DOM      `document.getElementById('myReport')`

JQUERY    `$( '#myReport' )`

# DOM vs JQuery

Example:

## Class selector

DOM      `document.getElementsByClassName('myReport')`

JQUERY    `$('.myReport')`

# JavaScript

# PLSQL vs JS

- JavaScript is the only language to change data on client side in Browser.
- SQL & PLSQL manipulate the data
- It adds lots to overall “feeling” in modern apps
- JavaScript is everywhere

## LOWCODE

- We do not have to be gurus
- It take only few lines to make things work
- Or do we?

# PLSQL vs JS

```
1  declare
2      foo NUMBER;
3  begin
4      foo := 2;
5      dbms_output.put_line(foo);
6
7  declare
8      foo_local number := foo;
9  begin
10     dbms_output.put_line('Entering nested block');
11     dbms_output.put_line(foo_local);
12     foo_local := 3;
13     dbms_output.put_line(foo_local);
14     dbms_output.put_line('Exiting nested block');
15 end;
16 dbms_output.put_line(foo);
17
18 --wrong type will not work
19 foo := 'It does not work';
20
21 end;
```

## PLSQL scope example

```
2
Entering nested block
2
3
Exiting nested block
2
```

# PLSQL vs JS

## JS scope example

```
1  var globalTest = 'Greetings';
2
3  ▶ function outer(){
4      var maxVal = 100;
5
6  ▶   function inner(){
7      var val;|
8
9      val = maxVal;
10     console.log(val);
11
12     val = 'No issues with this';
13     console.log(val);
14   }
15 }
```

# PLSQL vs JS

## PLSQL

- Scope are PLSQL blocks
- Not case sensitive
- Data type does not change
- Variables are always in declaration section
- Procedures and functions supported
- Each variable needs a name and a type
- Many data types

clob, blob, records,  
collections, number, varchar2,  
boolean, char, date,  
timestamp...

## JS

- Scopes are functions
- It is case sensitive
- Variables can be defined anywhere
- There are no procedures
- Data type can be swapped at any time
- Variable only needs a name

# JS primitive vs object types

```
1  var myString = 'Test';
2  var myNumber = 2;
3
4  var myBoolean = true;
5
6  var myArray = ['a', 'b'];
7  var myObject = {name:'Lino', role:'Admin'};
8
9  //using NEW constructor
10 var myDate = new Date(2021, 2, 2);
11
```

# JS data types

Data Types	Description	Example
<code>String</code>	represents textual data	<code>'hello'</code> , <code>"hello world!"</code> etc
<code>Number</code>	an integer or a floating-point number	<code>3</code> , <code>3.234</code> , <code>3e-2</code> etc.
<code>BigInt</code>	an integer with arbitrary precision	<code>900719925124740999n</code> , <code>1n</code> etc.
<code>Boolean</code>	Any of two values: true or false	<code>true</code> and <code>false</code>
<code>undefined</code>	a data type whose variable is not initialized	<code>let a;</code>
<code>null</code>	denotes a <code>null</code> value	<code>let a = null;</code>
<code>Symbol</code>	data type whose instances are unique and immutable	<code>let value = Symbol('hello');</code>
<code>Object</code>	key-value pairs of collection of data	<code>let student = {};</code>

# JS functions

- Functions are like any other data type
- Can be assigned to a variable
- Passed around as parameters
- Returned from other functions

```
1  ▶ function myTest1(fun){  
2      console.log('Yeah!!!!');  
3      fun();  
4  };  
5  
6  ▶ function myTest2(){  
7      console.log('Neah!!!');  
8  };  
9  
10 myTest1(myTest2);
```

Yeah!!!!

Neah!!!

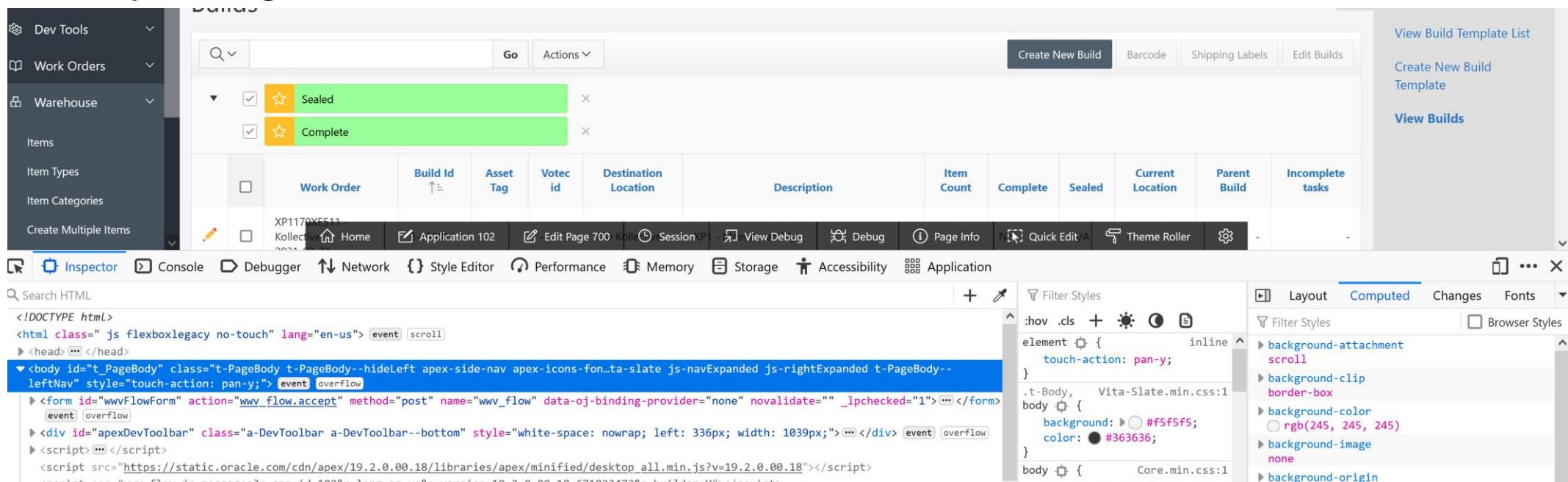
# AJAX

- It stands for Asynchronous JavaScript and XML
- Web pages can send HTTP requests to the server and receive a response
- Without needing to reload the entire page
- We use it in APEX on many places
- Report refresh being a typical example

```
// Async examples: https://javascript.info/fetch
async function getJson() {
    const url = "f?p=&APP_ID.:0:&SESSION.:APPLICATION_PROCESS=app_get_clob:NO:RP:::"
    const response = await fetch(url);
    const text = await response.text();
    gJSON = text;
    // Do something with gJSON.
}
```

# DEVELOPER TOOLS

- Good text editor and browser
- Similar to SQL Developer
- Open it with F12 or right click Inspect (Ctrl + Shift + I)
- Inspecting elements



# Oracle Developers

# Perfect candidates

30%

Search for books, videos and tutorials to learn APEX and get a personal Workspace on [apex.oracle.com](http://apex.oracle.com). **It's free!**

## ORACLE DATABASE

60%

Learn the basic principles of the Oracle Database, SQL, and PL/SQL.

80%

You can develop in APEX without knowing HTML, CSS, and JavaScript but you'll discover a whole new world after learning it.

100%

To achieve everything you can think of, you should learn jQuery and Dynamic Actions.  
**There are no limits for you!**

## FRONT-END TECHNOLOGIES

## ¡FINAL STEP!

# APEX and JavaScript

- Inline on a page
- Static application file or as application theme referenced files
- Anywhere where we can do HTML in APEX

## **Where can we find JS in APEX:**

- Dynamic actions/AJAX
- Interactive grids configurations (JET Chart, Cards and other regions)
- Actions interface
- Inclusion of additional JS library like Numeral.js
- Plugins
- Client side validations

# APEX developer perspective

The screenshot shows the Oracle APEX Page Designer interface for a page named "Error debug".

**Page Header:** Application 25616 \ Page Designer

**Page Number:** 55

**Page Buttons:** Back, Forward, Go, Open, New, Save, Print, Refresh.

**Page Identification:**

- Name: Error debug
- Alias:
- Title: Error debug
- Page Group: - Select -

**Page Appearance:**

- User Interface: DESKTOP
- Page Mode: Normal
- Page Template: Standard
- Template Options: Use Template Defaults
- CSS Classes:
- Media Type:

**Page Navigation:**

- Layout: PAGE HEADER, PAGE NAVIGATION, BREADCRUMB BAR, BEFORE CONTENT BODY, CONTENT BODY.
- Content Body Components:
  - Info (selected)
  - Maintain Employee
  - Region Body
    - P55\_ROWID
    - P55\_EMPNO
    - P55\_ENAME
    - P55\_JOB
    - P55\_MGR
    - P55\_HIREDATE
    - P55\_SAL
    - P55\_BONUS
    - P55\_COMM
    - P55\_DEPTNO
    - P55\_LOCATION
    - P55\_NO\_OF\_EMPLOYEES

**Page Footer:**

- Buttons: EDIT, CLOSE, CREATE, COPY
- Regions: REGION BODY, REGION CONTENT, SUB REGIONS
- Links: CHANGE, PREVIOUS, NEXT

**Page Footer:**

- Maintain Employee



Edit Page 3



js/demo.js

Cancel

Delete

Save Changes

✓ File saved and minified X

Directory

js ☰

\* File Name

demo.js ?

Reference

#APP\_FILES#js/demo#MIN

#.js



Mime Type

text/javascript ?

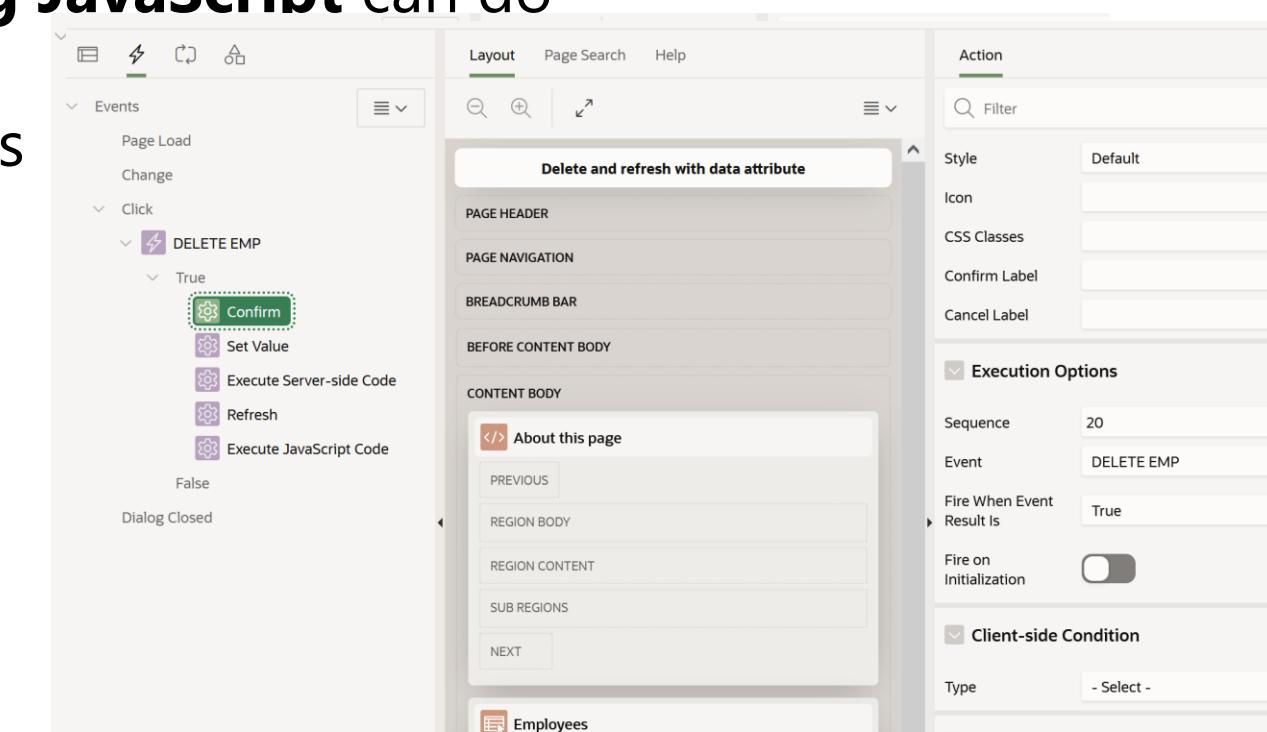
```
1 var value = apex.item( "P1_ITEM" ).element;  
2  
3 apex.message.showPageSuccess("Employee #"+ $v("P3_EMPNO")+" removed.");
```

# Including JS on a page

&gt;21.2 APEX version

# DYNAMIC ACTIONS

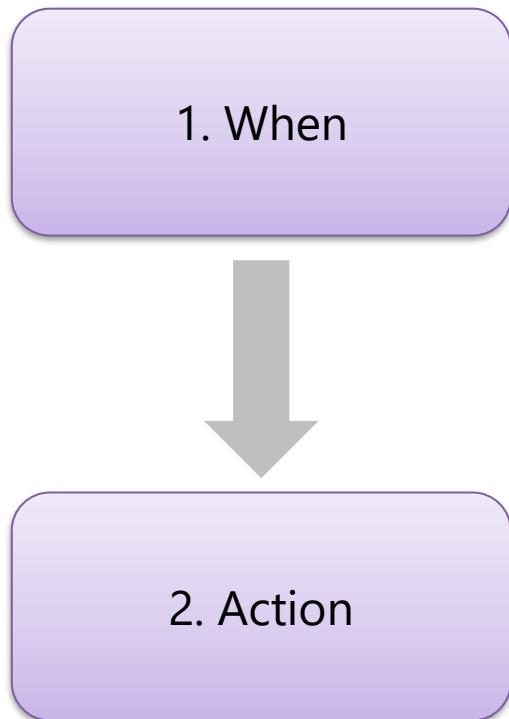
- Easiest way to include JS in APEX
- Declarative => little coding with little to no knowledge of JS
- **Most likely will not cover everything JavaScript can do**
- When complexity grows put it into files
- Code Sharing between pages
- We can now create and edit static JS files from APEX Builder



# DYNAMIC ACTIONS

```
1  <html>
2  <body>
3
4  <h2>JavaScript addEventListener()</h2>
5
6  <button id="myBtn">Try it</button>
7
8  <script>
9  document.getElementById("myBtn").addEventListener("click", myFunction);
10
11 function myFunction() {
12   alert ("Hello World!");
13 }
14 </script>
15
16 </body>
17 </html>
```

# DYNAMIC ACTIONS 101



## **1. When the Dynamic Action should happen or "fire"**

- Browser Events
- Framework Events
- Component Events

**OPTIONAL** client side condition

## **2. What you want to do when an event takes place**

- True Actions
- False Actions

## **3\*. Affected elements or Scope**

# DYNAMIC ACTIONS

@JuergenSchuster



Juergen @JuergenSchuster · Nov 19

I'm afraid I have become a #orclapex Dynamic Action junkie 🙈 100 DAs on one page, anybody else who has this addiction?



@Matt FOEX

```
1 select da_count_tab.*  
2   from (select aap.page_id  
3          ,(select count(*)  
4            from apex_application_page_da aapd  
5              where aapd.application_id = 200  
6                and aapd.page_id = aap.page_id) as count_da  
7      ,(select count(*)  
8        from apex_application_page_da Acts aapda  
9          where aapda.application_id = 200  
10             and aapda.page_id = aap.page_id) as count_da_actions  
11    from apex_application_pages aap  
12      where aap.application_id = 200) da_count_tab  
13  where da_count_tab.count_da > 100  
14  order by da_count_tab.count_da desc
```

PAGE_ID	COUNT_DA	COUNT_DA_ACTIONS
1	350	819
2	379	995
3	457	996
4	369	928
5	478	973
6	468	937
7	477	952
8	490	954
9	469	961
10	493	946
11	461	959
12	494	950
13	474	979
14	501	483
15	473	943
16	486	934
17	497	922
18	487	883
19	489	931
20	472	935
21	491	926
22	492	929
23	1500	517
24	488	868
25	495	992

# DYNAMIC ACTIONS

- Consolidate
- Do not be afraid to use manual code

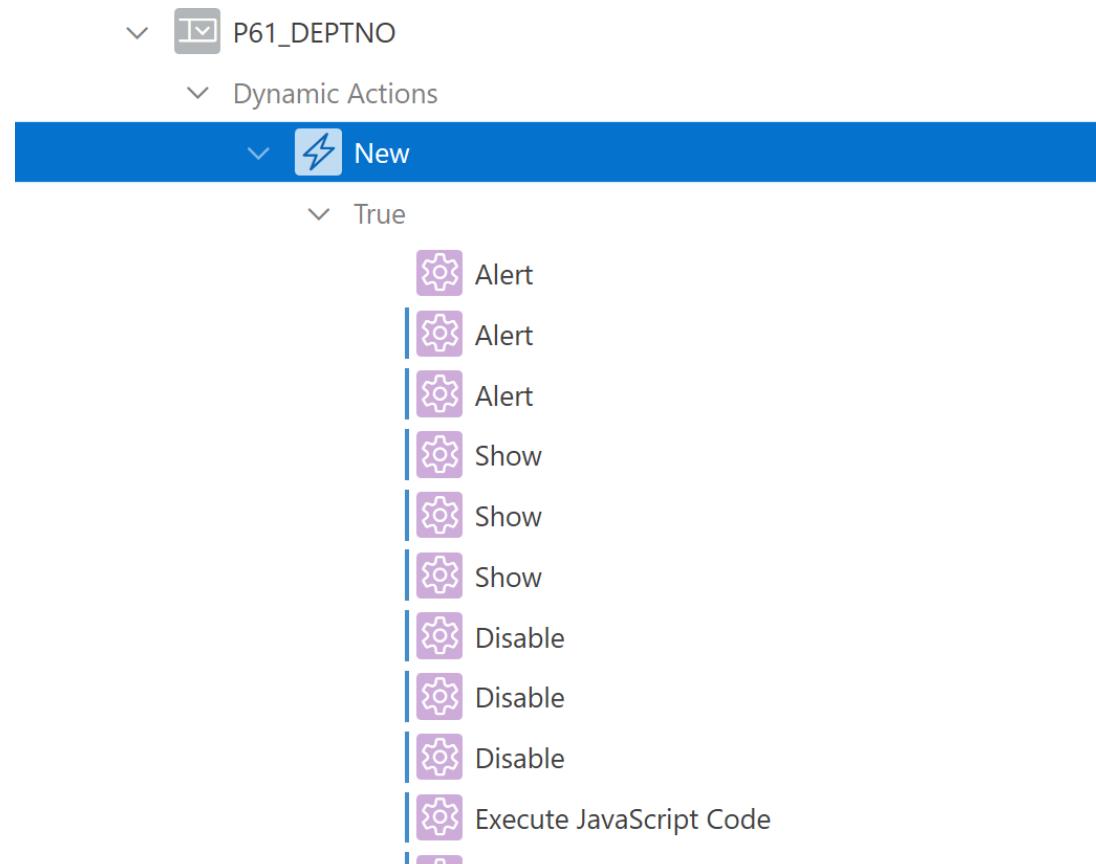
```
/***
 * @function show
 ***/
show: function () {
    if (fabe.navigation.back.stack.length > 0 || !apex.item("P0_BACK_URL").isEmpty()) {
        $(".fabe-back-all").show();
        $(".brand-logo img").hide();
    }

    if (!apex.item("P0_BACK_URL_MOBILE").isEmpty()) {
        $(".fabe-back-mobile").show();
        $(".brand-logo img").addClass("hide-on-med-and-down");
    }
}

/***
 * @function hide
 * @example
 * fabe.navigation.back.hide();
 ***/
hide: function () {
    console.log(fabe.navigation.back.stack.length === 0,
        apex.item("P0_BACK_URL").isEmpty());

    if (fabe.navigation.back.stack.length === 0 && apex.item("P0_BACK_URL").isEmpty()) {

```



# Modularize JS code

- Gives context to JavaScript functions
- Example one JavaScript module per APEX page

```
1  /**
2  * @namespace page1
3  */
4  const page1 = {};
5
6  /**
7  * @module demo
8  */
9  page1.demo = {
10    /**
11     * @function doSomething
12     * @example page1.demo.doSomething();
13     */
14    doSomething:function(){
15      //write your JS source here
16    }
17  };
18
```

page1.demo.doSomething();

# APEX versions – new APIs



# APEX JS APIs

## Shortcut to documentation

- apex.oracle.com/jsapi
- Namespaces
- Interfaces
- Widgets

### Example

```
// Displays a page-level success message 'Changes saved!'.
apex.message.showPageSuccess( "Changes saved!" );
```

```
apex.debug.error( "Update Failed" );
```

```
apex.page.submit( "DELETE" );
```

```
> apex.items.P26_AREA_DESCRIPTION.value
< 'Ore Preperation'
```

```
var dateString = apex.date.format( myDate, "YYYY-MM-DD HH24:MI" );
// output: "2021-06-29 15:30"

apex.pwa.openInstallDialog();
```

```
apex.server.process( "MY_PROCESS", {
    x01: "test",
    pageItems: "#P1_DEPTNO,#P1_EMPNO"
}, {
    success: function( data ) {
        // do something here
    },
    error: function( jqXHR, textStatus, errorThrown ) {
        // handle error
    }
} );
apex.env

▼ Object { APP_USER: "LSCHILDE@GMAIL.COM", APP_ID: "83349", API_APEX_FILES: "https://static.oracle.com/cdn/apex/21.2.0/" }
  APEX_FILES: "https://static.oracle.com/cdn/apex/21.2.0/"
  APEX_VERSION: "21.2.0"
  APP_FILES: "lschilde/r/83349/files/static/v1/"
  APP_ID: "83349"
  APP_PAGE_ID: "13"
  APP_SESSION: "1474258414065"
  APP_USER: "LSCHILDE@GMAIL.COM"
  WORKSPACE_FILES: "lschilde/r/files/static/v103/"
▶ <prototype>: Object { ... }
```

# VALIDATIONS - Client side

The screenshot shows a user interface for adding a new customer. On the left is a vertical sidebar with icons for Home, Search (31 results), Cart (10 items), and other functions. The main area has a "Cancel" button on the left and a "Add Customer" button on the right. The form fields include:

- First Name \*: An input field with a red asterisk and a small info icon.
- Last Name \*: An input field with a red asterisk.
- Street Address: An input field.
- Line 2: An input field.
- City: An input field.
- State \*: A dropdown menu with a red asterisk and a placeholder "- Choose State -".
- Zip Code \*: An input field with a red asterisk.
- Credit Limit \*: An input field with a red asterisk.
- Phone Number: An input field containing "999-999-9999".
- Alternate Number: An input field containing "999-999-9999".
- Email: An input field.
- URL: An input field.
- Tags: An input field.

Red asterisks (\*) are placed next to required fields, and a blue border highlights the First Name input field, indicating it is currently selected or has focus.

# VALIDATIONS - Client side

- Better user experience
- Data quality
- Only notified after submitting the page
- **Important to give the users immediate feedback**
- Validate items, forms, IGS
- We should do it also on server side

Prevent invalid user inputs

Email addresses

Number

Password complexity

Phone numbers

```
1 function isEmpty(pValue) {
2     var isEmpty = false;
3     if ($.trim(pValue) === "") {
4         isEmpty = true;
5     }
6     return isEmpty;
7 }
8
9 function isPositiveInteger(pValue) {
10    // an integer is a number that can be written without a fractional or decimal component
11    var isPositiveInteger = false;
12    var positiveIntegerRegex = /\^\d+$/;
13
14    if (pValue.match(positiveIntegerRegex)) {
15        isPositiveInteger = true;
16    }
17    return isPositiveInteger;
18 }
19
20 function isValidDate(pValue) {
21     var isValidDate = false;
22     // date format is DD/MM/YYYY
23     var dateFormatRegex = new RegExp("^(3[01]|1[2][0-9]|0[1-9])/((1[0-2]|0[1-9])/(?:[0-9]{2})?[0-9]{2}$)");
24
25     if (pValue.match(dateFormatRegex)) {
26         // seems that the date format is correct, but can we parse the date to a date object?
27         var dateArray = pValue.split("/");
28         var year = parseInt(dateArray[2]);
29         var month = parseInt(dateArray[1], 10);
30         var day = parseInt(dateArray[0], 10);
31         var date = new Date(year, month - 1, day);
32
33         if (((date.getMonth() + 1) === month) && (date.getDate() === day) && (date.getFullYear() === year)) {
34             isValidDate = true;
35         }
36     }
37
38     return isValidDate;
39 }
```

<https://www.youtube.com/watch?v=-fn4mHZC-AA>

# ACTIONS interface

- Declarative way to improve your apps with built in APEX JS capability
- Reduce number of DAs
- Reduce number of buttons
- **Less replication of your code**
- Define action shortcuts

The image shows two screenshots illustrating the Oracle APEX Actions interface. On the left, the 'Actions API' configuration for a 'Button' is displayed. The 'Behavior' section is expanded, showing 'Action: Defined by Dynamic Action', 'Execute Validations' turned on, and 'Requires Confirmation' turned off. The 'Advanced' section includes 'Static ID' and 'Custom Attributes' with 'data-action="alert"'. On the right, a browser window titled 'What is an Action?' shows a green success message 'This is success'. Below the page title, there are three demo links: 'Demo 1 - Buttons', 'Demo 2 - Toggle', and 'Demo 3 - Select'. The browser's developer tools console tab is open, displaying the following JavaScript code:

```
apex.actions.add({
  name: "alert",
  label: "Alert",
  action: function(event, focusElement) {
    apex.message.alert('This is an action!');
  }
});
```

<https://skillbuilders.com/course/how-the-oracle-apex-actions-interface-make-apps-better-free-tutorial/>

# PL/SQL and passing values

## Establish communication between client and server side

- React to an event
- Pass required data in
- Execute PL/SQL code
- Return the result
- Notify the user

```
1 declare
2   p_project_id number := APEX_APPLICATION.g_x01;
3   --p_action varchar2(50) := APEX_APPLICATION.g_x02;
4 begin
5   delete from eba_demo_da_emp where empno = p_project_id;
6   apex_json.open_object;
7   apex_json.write('success', true);
8   apex_json.close_object;
9 exception
10   when others then
11     apex_json.open_object;
12     apex_json.write('error', false);
13     apex_json.write('message', sqlerrm);
14     apex_json.close_object;
15 end;
```

3

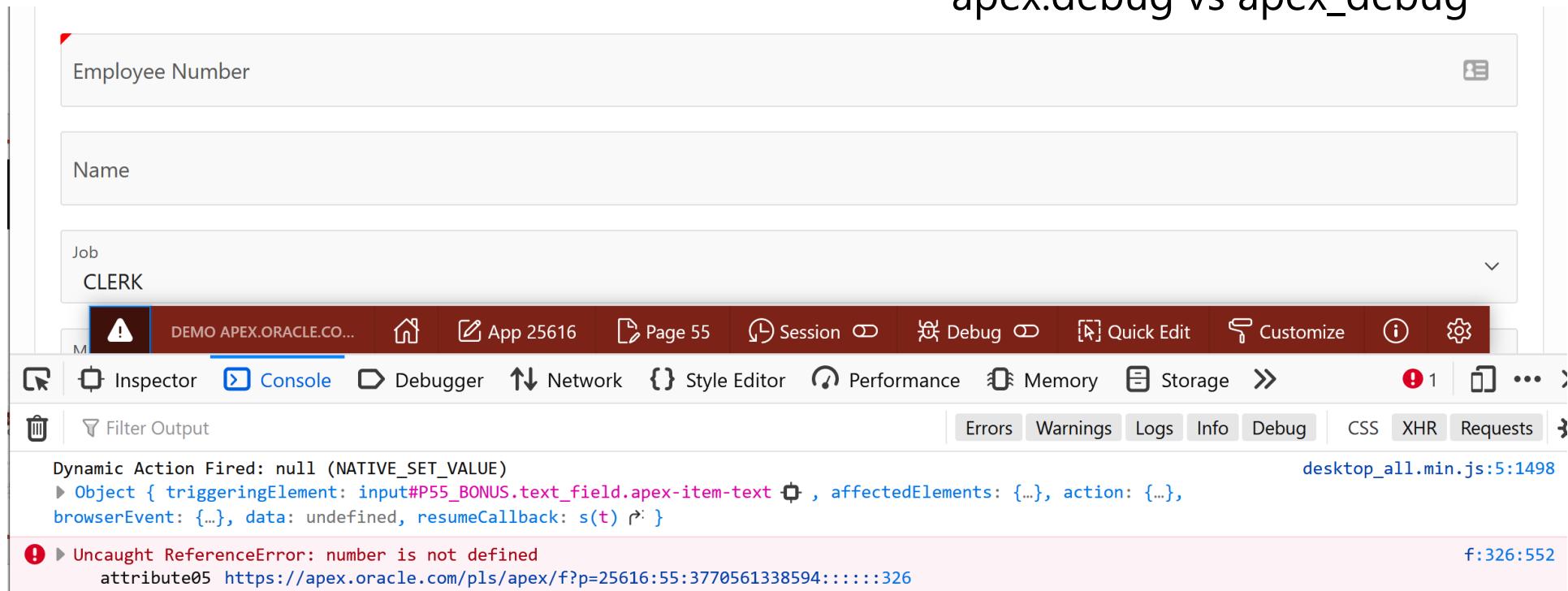
```
1 <button type="button" title="My Button" aria-label="My Button" class="t-Button t-Button--noLabel"
2   onclick="apex.event.trigger(document,'deleteEmp',[{action:'rejecting', id:'#ID#'}]);">
3   |   <span aria-hidden="true" class="t-Icon fa fa-trash" ></span>
4 </button>

1 var sentID = this.data.id;
2 var action = this.data.action;
3 //var lSpinner$ = apex.util.showSpinner();
4 apex.server.process("DELETE_EMP",
5   |   { x01: pID,
6     |     x02: pAction,
7     |   },
8   |   { loadingIndicatorPosition : "page",
9     |     success: function(pData) {
10       // If the AJAX is successful set the value or the returned items
11       if (pData.success === true){
12         apex.region( "employees" ).refresh();
13         apex.message.showPageSuccess( "Sucessfully deleted ID = " + pID + "!" );
14       }
15     },
16     error: function(request, status, error) {
17       apex.message.clearErrors();
18       // Now show new errors
19       apex.message.showErrors([
20         |       ])
```

2

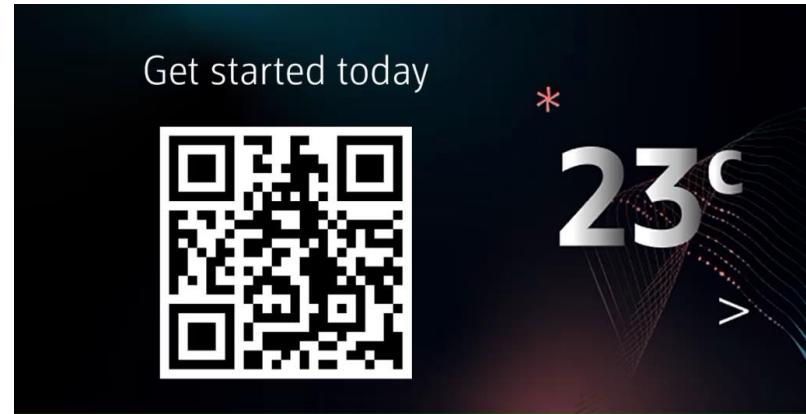
# DEBUGGING?

- Browser Tools
  - F12
  - Right click inspect
  - Ctrl + Shift + I
- Get familiar with features
- Console log & Network tab
  - Make sure you enable APEX debug
  - Check the name of the DA
  - Check parameters sent in
  - Check the payload returned
  - apex.debug vs apex\_debug



# That is not all!!!!

- **JavaScript stored procedures**
- Integration of GraalVM
- Execute SQL from within JavaScript
- Leverage a huge set of JavaScript libraries
- Processes, Computations, Validations, and the Execute Server-side code DA



The screenshot shows the Oracle APEX developer interface. At the top, there is a navigation bar with tabs for "Identification", "Language" (set to "JavaScript (MLE)"), "Environment", "Default", "Rows" (set to 10), and other settings. Below the navigation bar, there are sections for "Action" (set to "Execute Server-side Code") and "Settings". In the "JavaScript Code" section, there is a code editor containing the following JavaScript code:

```
apex.conn.execute("begin dbms_output.put_line('Hello from plsql'); end;");
```

Below the code editor, there is another section of JavaScript code:

```
var rows;
function needsSalaryRaise( empno ) {
    // placeholder for some complicated logic to calculate if employee needs a raise
    return true;
}
rows = apex.conn.execute( "select empno from emp where deptno = :deptno", { deptno: apex.env.P2_DEPTNO } ).rows;
for ( var row of rows ) {
    if ( needsSalaryRaise( row.EMPNO ) ) {
        apex.conn.execute( "update emp set sal = sal * 1.03 where empno = :empno", { empno: row.EMPNO } );
    }
}
```

# JavaScript - MLE

Language JavaScript (MLE) Environment Default Rows 10 Clear Command Find Tables

↻ ⌂ Q A..

```
1 function extendProjectTasks( name ) {
2     if (name !== '') {
3         return true;
4     }
5     else {
6         return false;
7     }
8 }
9
10 for ( var row of apex.conn.execute( "select id, name from eba_project_tasks where project_id = :project", { project: "5" } ).rows ) {
11     if ( extendProjectTasks( row.name ) ) {
12         apex.conn.execute( "update eba_project_tasks set end_date = end_date + 1 where id = :id", { id: row.ID } );
13         console.log("The task with the ID: " + row.ID + " and the name " + row.NAME + " has been extended successfully!")
14     }
15 }
16
17
```

Results Explain Describe Saved SQL History

The task with the ID: 34 and the name Design production screens for main Partner Portal app has been extended successfully!  
The task with the ID: 35 and the name Train Partners has been extended successfully!  
The task with the ID: 36 and the name Monitor Partners has been extended successfully!  
The task with the ID: 37 and the name Review Beta Feedback from Partners has been extended successfully!  
The task with the ID: 38 and the name Improve existing feature functions based on feedback has been extended successfully!  
Statement processed.

0.14 seconds

# PWA – push notifications

## Notifications

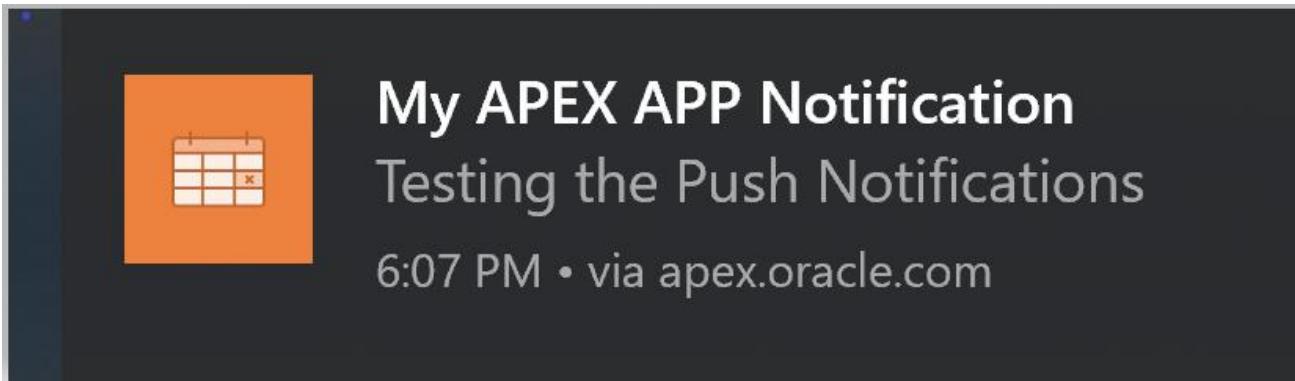
- Receive push notifications directly on your device, featuring a message title, body, icon, and link.

## Easy Subscription

- APEX users can subscribe to push notifications from within their applications, while instance administrators can manage a queue of notifications.

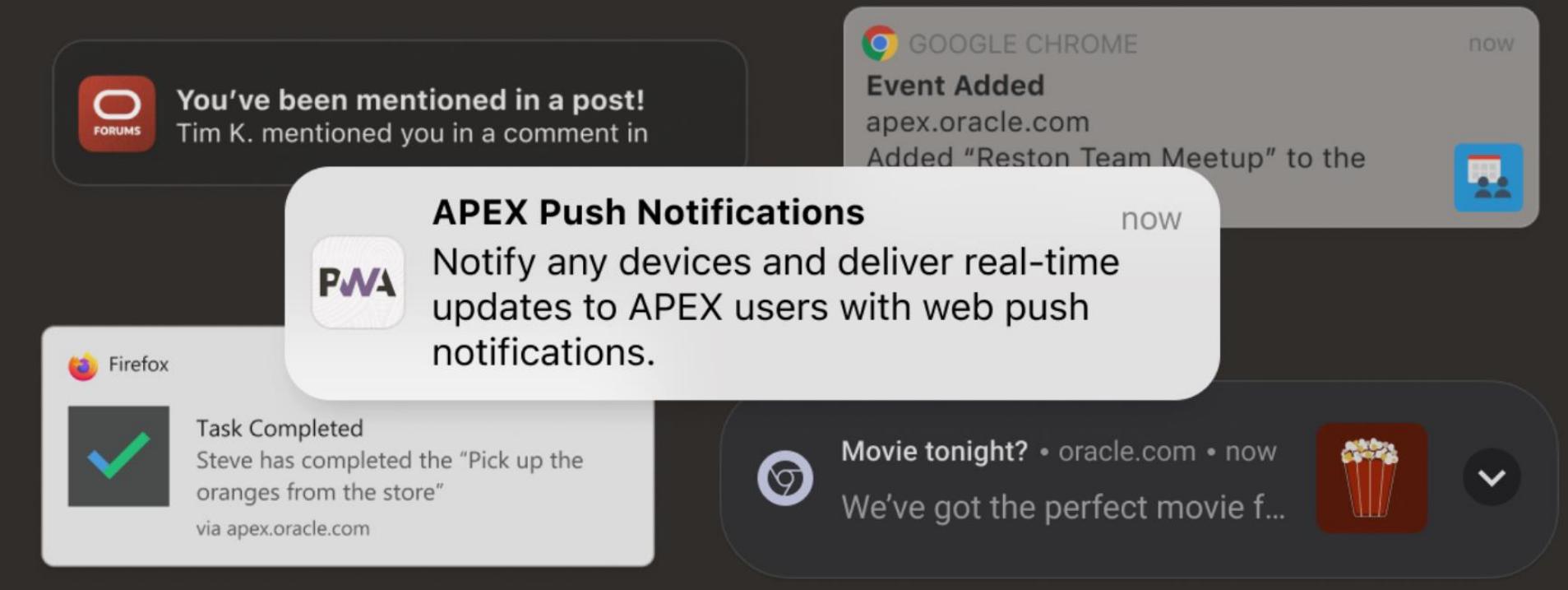
## Simple NO config using new APEX APIs

- apex\_pwa.send\_push\_notification



# PWA – push notifications

Low code push notifications, made available with a single switch.  
Easily notify desktop and mobile users who've opted-in to receive push notifications.



# **Workshop**

[codeshare.io/WdK9vd](https://codeshare.io/WdK9vd)

# Summary

JavaScript is here to stay

- Little we know is enough to get us through
- Familiarize your self with JS and available built in features

Consolidate

- if it can trigger on same event or global page
- using actions interface

Minimize custom code

- But don't be afraid to use it
- Structure it well

Think security

- Apply server side security too
- Don't trust user inputs

# Q&A

---

Thank you for attending



# Resources

- [https://www.youtube.com/watch?v=\\_Izurjl5Vm0](https://www.youtube.com/watch?v=_Izurjl5Vm0)
- [https://www.youtube.com/watch?v=02thxqv-m\\_c](https://www.youtube.com/watch?v=02thxqv-m_c)
- <https://www.youtube.com/watch?v=-l6E5LuNU3U>
- <https://www.youtube.com/watch?v=uK7vCqfXxNs>
- <https://skillbuilders.com/course/learning-javascript/>
- <https://www.youtube.com/watch?v=phydnTdH81Y>
- <https://medium.com/graalvm/javascript-as-a-server-side-language-in-oracle-apex-20-2-457e073ca4ca>
- <https://technology.amis.nl/database/executing-javascript-from-pl-sql-in-oracle-database/>