

# JavaScript for PL/SQL DEVELOPERS



hroug 22

Spring | Proljeće

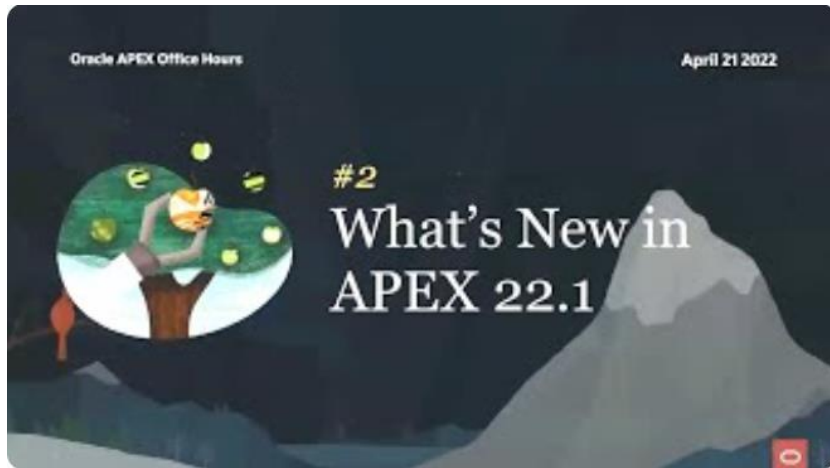
17. - 19.05.2022., Tuhelj

**Lino Schildenfeld**

@LinoSchilde

lschilde.blogspot.com

# APEX Office hours



## Part 2: What's new in Oracle APEX 22.1

**Marc Sewtz**, Chaitanya Koratamaddi, Mónica Godoy

247 views · April 21, 2022



## Part 1: Sneak Preview of What's New

**Marc Sewtz**, Chaitanya Koratamaddi, Mónica Godoy

2,188 views · April 14, 2022

[apex.oracle.com/officehours](https://apex.oracle.com/officehours)

# AUSOUG APEX News



[Become a Member](#) [What's On](#) [Branches](#) [Resources](#) [Past Events](#) [Blog](#) [My Account](#)



**AUSOUG**  
**APEX Special Series 2022**  
**ā'pěks**  
**(#orclapex)**  
**2022 SPECIAL SERIES**

[www.ausoug.org.au/apex-special-series-2022/](http://www.ausoug.org.au/apex-special-series-2022/)

# My story



- AUSOUG APEX webinars
- NZ APEX meetup organizer
- APEX blogger
- Conference speaker
- Developer since 2006

@LinoSchilder



**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">
  <head>
  <body class=" apex-theme-standard has-envBanner--left">
    <!--
    [if lte IE 9]><div id="outdated-browser">You are using an out
    web browser. For a list of supported browsers, please referen
    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"
  label="DEMO APEX.ORACLE.COM">
```

```
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

# 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

# DOM manipulation

- Add/remove classes
- Modify attributes
- Element manipulation
- Removal
- 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);

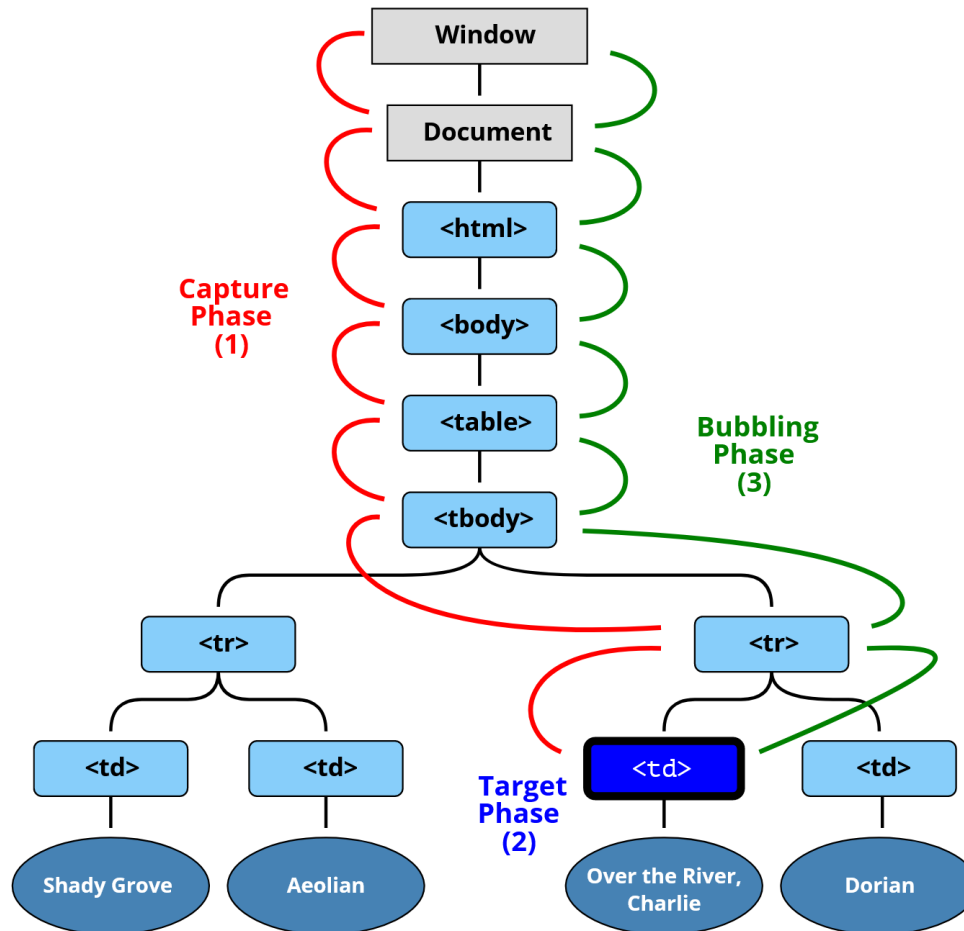
      }
    </script>
  </head>
  <body>
  </body>
</html>
```



# 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' );	.....

# APEX page too



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

# 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
- **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 <script>
16 $( "input[name*='man']" ).val( "has man in it!" );
17 </script>
18
19 </body>
20 </html>
```

# DOM vs JQuery

## Example: ID selector

DOM `document.getElementById('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 masters
- Only few lines to make things work
- Or do we?

# PLSQL vs JS

## PLSQL scope example

```
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;
```

```
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);
```

# 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>

<https://www.programiz.com/javascript/data-types>

# JS functions

- Functions are like any other data type
- Can be assign 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;
  // alert(gJSON);
}
```

# DEVELOPER TOOLS

- Similar to SQL Developer
- Open it with F12 or right click Inspect (Ctrl + Shift + I)
- Inspecting elements
- Debugging - Get familiar with Console and Network tab

The screenshot displays the Oracle APEX Developer Tools interface. The top section shows a table of work orders with columns for Work Order, Build Id, Asset Tag, Vote id, Destination Location, Description, Item Count, Complete, Sealed, Current Location, Parent Build, and Incomplete tasks. Two work orders are highlighted in green: 'Sealed' and 'Complete'. Below the table, the browser's developer tools are open, showing the HTML structure of the page. The selected element is a form with the following attributes: `<form id="wvFlowForm" actions="wv_flow.accept" method="post" name="wv_flow" data-oj-binding-provider="none" novalidate="" _lpcchecked="1">`. The developer tools also show the CSS styles for the selected element, including background-color, border-box, and padding-box.

# 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.

**FRONT-END  
TECHNOLOGIES**

80%

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

**¡FINAL STEP!**

100%

To achieve everything you can think of, you should learn jQuery and Dynamic Actions.

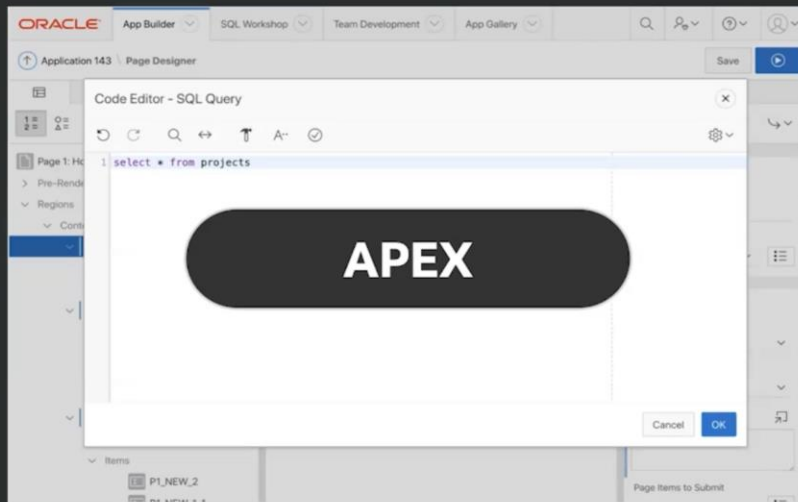
**There are no limits for you!**

# LOWCODE = APEX

- Focus on business solutions
- Improve productivity
- Deliver quality and consistency
- Directly leverage scalability, security attributes on Oracle DB
- Utilize all of your Oracle skills

Oracle APEX lets you build enterprise apps **20x faster** with **100x less code**

Developer Day —  
**Australia**

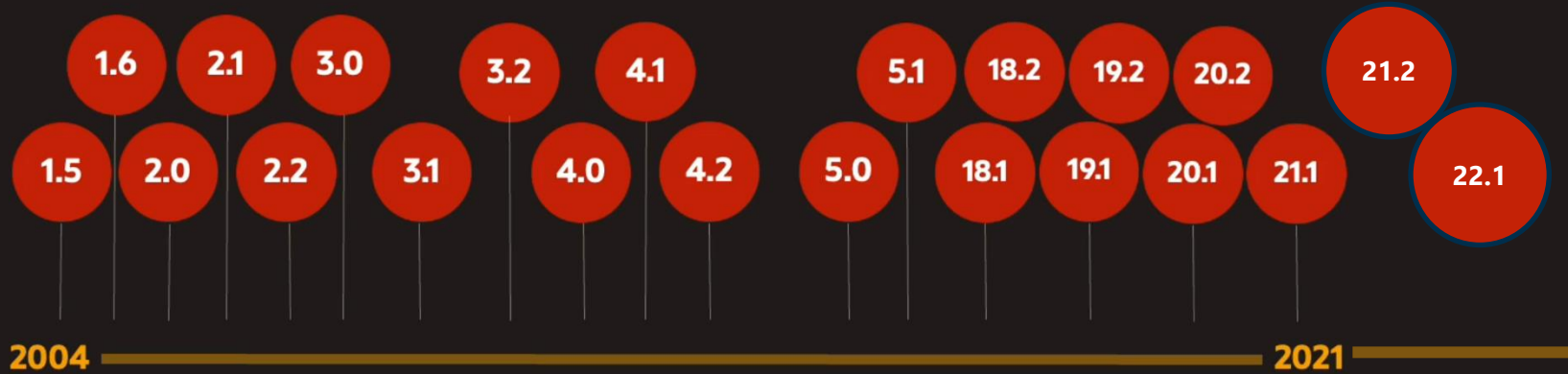
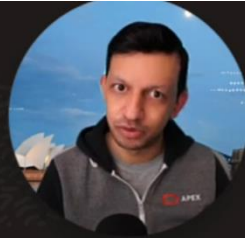


VS



# APEX versions

## APEX Product Timeline



**20**  
Releases

**250+**  
Major features

**0**  
required rewrites

# APEX and JavaScript

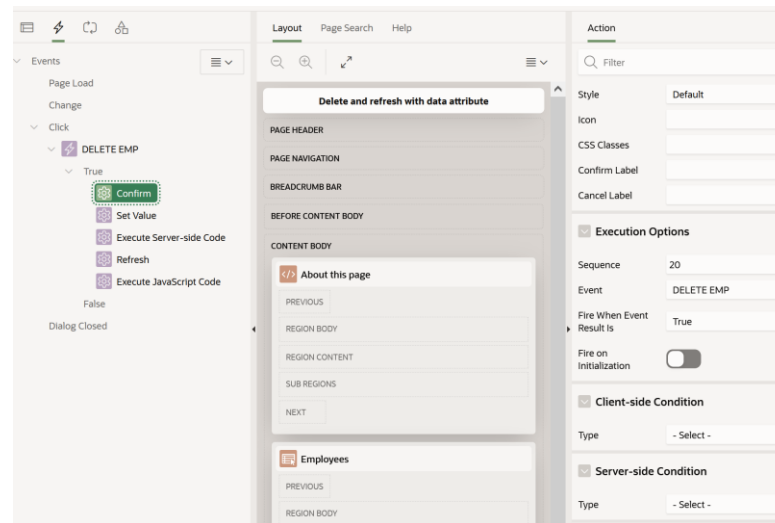
- Inline on a page
- Static application file or as application theme referenced files

Where can we find JS in APEX:

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

# DYNAMIC ACTIONS

- Easiest way to include JS in APEX
- Declarative => little coding
- **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

@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?



10 1 20



@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	902
26	496	904
27	302	488

# DYNAMIC ACTIONS

- Consolidate
- Do not be afraid to use manual code

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

    if (!apex.item("PO_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("PO_BACK_URL").isEmpty());

    if (fabe.navigation.back.stack.length === 0 &&
        apex.item("PO_BACK_URL").isEmpty()) {
        $(".fabe-back-all").hide();
        $(".brand-logo img").show();
    }

    if (fabe.navigation.back.stack.length === 0 &&
        apex.item("PO_BACK_URL MOBILE").isEmpty()) {
        $(".fabe-back-mobile").hide();
        $(".brand-logo img").removeClass("hide-on-med-and-down");
    }
},
```

The screenshot shows the configuration interface for a Dynamic Action in Salesforce Lightning. The top part shows a dropdown menu for the component 'P61\_DEPTNO' and a sub-menu for 'Dynamic Actions'. A blue bar highlights the 'New' button. Below this, a dropdown menu is set to 'True'. A list of actions is displayed, each with a gear icon and a label: Alert, Show, and Disable. The 'Alert' action is selected. At the bottom, there is a 'False' label.

# 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 JS APIs

## Shortcut to documentation

- [apex.oracle.com/jsapi](https://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.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

- 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]|[12][0-9]|0?[1-9])/(1[0-2]|0?[1-9])/(?:[0-9]{2})?[0-9]{4}$");
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  return isValidDate;
38 }
```

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

# VALIDATIONS - Client side

```
1 var textitem = apex.item(this.triggeringElement.id),
2   errors = [],
3   alphanum = /^[0-9a-zA-Z]+$/;
4   val = textitem.getValue();
5
6 // Perform Validation: Check if alphanumeric
7 if (val.length > 0 && val.match(alphanum) ) {
8   textitem.node.setCustomValidity(""); // valid
9 } else {
10  textitem.node.setCustomValidity("Invalid"); // rely on data-valid-message attribute to give a message
11 }
12
13 // Raise/clear errors based on validity
14 if (!textitem.getValidity().valid) {
15   errors.push({
16     message: textitem.getValidationMessage(),
17     location: "inline",
18     pageItem: textitem.id
19   });
20   apex.message.showErrors(errors);
21 } else {
22   apex.message.clearErrors(textitem.id);
23 }
```

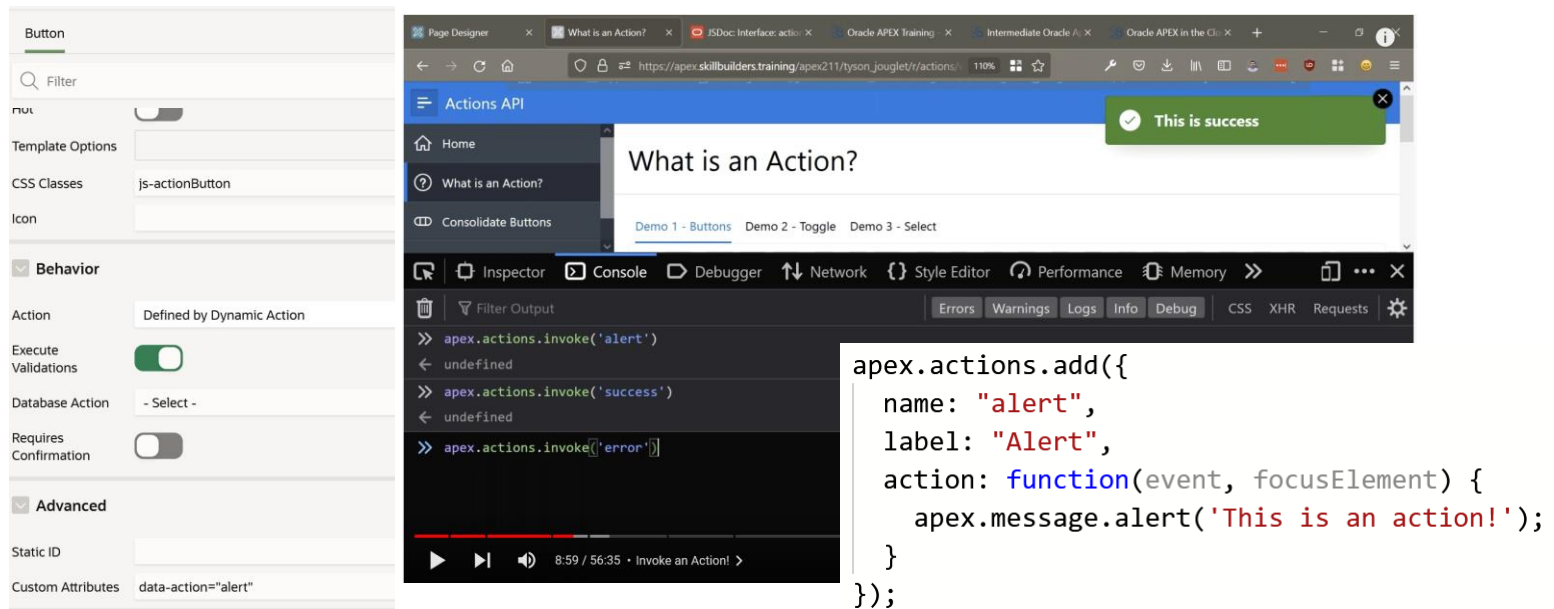
**apex.item Validity**

I

<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 the APEX Actions interface in two parts. On the left is the configuration panel for a button, and on the right is a browser window showing the application running with a console log.

**Configuration Panel (Left):**

- Filter:** Search field.
- Template Options:** Input field.
- CSS Classes:** js-actionButton
- Icon:** Input field.
- Behavior:**
  - Action: Defined by Dynamic Action
  - Execute Validations:
  - Database Action: - Select -
  - Requires Confirmation:
- Advanced:**
  - Static ID: Input field
  - Custom Attributes: data-action="alert"

**Browser Window (Right):**

- Page Designer: What is an Action?
- URL: https://apex.skillbuilders.training/apex211/tyson\_joulet/r/actions/
- Page Title: What is an Action?
- Page Content: Demo 1 - Buttons, Demo 2 - Toggle, Demo 3 - Select
- Console Log:

```
apex.actions.invoke('alert')
← undefined
apex.actions.invoke('success')
← undefined
apex.actions.invoke('error')
```
- Notification: This is success

**Code Snippet:**

```
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

```
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 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 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( "Successfully 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
- Console and Network tab

The screenshot displays a web browser's developer tools interface. At the top, a table lists items with columns for 'Work Order', 'Build Id', 'Asset Tag', 'Vote id', 'Destination Location', 'Description', 'Item Count', 'Complete', 'Sealed', 'Current Location', 'Parent Build', and 'Incomplete tasks'. Two items are highlighted in green: 'Sealed' and 'Complete'. Below the table, the browser's navigation bar shows the address bar and various tool icons like 'Inspector', 'Console', 'Debugger', 'Network', 'Style Editor', 'Performance', 'Memory', 'Storage', 'Accessibility', and 'Application'.

The 'Inspector' panel is open, showing the HTML structure of the page. The selected element is a `<div id="apexDevToolbar" class="a-DevToolbar a-DevToolbar--bottom" style="white-space: nowrap; left: 336px; width: 1039px;">`. The 'Style' pane on the right shows the computed styles for this element, including `background-color: #f5f5f5`, `color: #363636`, and `font-family: 'Helvetica Neue', 'Segoe UI', Helvetica, Arial, sans-serif`.

# 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=\\_IzurjI5Vm0](https://www.youtube.com/watch?v=_IzurjI5Vm0)
- [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=phydnTdH8IY>