Validate Your Validations: ALL Sides Now

Karen Cannell
kcannell@thtechnology.com
TH TECHNOLOGY
@thtechnology

KODEKS
30 GODINA RJEŠENJA.

DELL Technologies
PLATINUM PARTNER

maistra
ROVINJ-VRSAR

ALGEBRA

MREŽA

About Me ...

- TH TECHNOLOGY – Oracle Consulting Services, APEX Focus
- Mechanical/SW Engineer - Analyzed, designed, developed, converted, upgraded, enhanced legacy & Oracle database applications for 30+ Years
- Web/APEX applications for Govt, Medical, Engineering, Fisheries since HTMLDB
- ODTUG Vice President, Editor Emeritus, Technical Journal
- APress Author
- Oracle ACE Director

Oracle ACE Program

500+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers

Oracle ACE Director | Oracle ACE Pro | Oracle ACE Associate

For more details on Oracle ACE Program:
ace.oracle.com

Oracle ACE

Nominate

yourself or someone you know:

ace.oracle.com/nominate

Connect: aceprogram_ww@oracle.com | Facebook.com/OracleACEs | @oracleace


APEX Grids: Editable Essentials
@thtechnology

#StaySafe
#StayVigilant

#Vaccinate
#Boost

ODTUG Kscope23

aurora, co june 25-29

Kscope23.odtug.com

GAYLORD ROCKIES
RESORT & CONVENTION CENTER
Colorado

About You ...

- APEX Experience?
- APEX Versions?
- Validation Strategy?
- JavaScript Experts?

Validations

Part I

- Why Validate?
- Have a Strategy
- Intro to Client-Side HTML & JS Validations
- What Matters

Part II

- Part I Recap
- Validations in Grids
- Strategy / Standards
- From the Floor (Beer)

Know Your Tool
Use Your Tool

What is Oracle APEX?

Oracle Application Express (APEX) is a low-code development platform that enables you to build scalable, secure enterprise apps, with world-class features, that can be deployed anywhere.

"Low Code" vs No Code

- Generate 100%
  → No Code

- Generate then Customize
  → Low Code

Let APEX Work For You

- Wizards
  - App, Page, Form/Report, Master/Detail ...
- Layout, Templates
- Security
- Validations
- Utilities

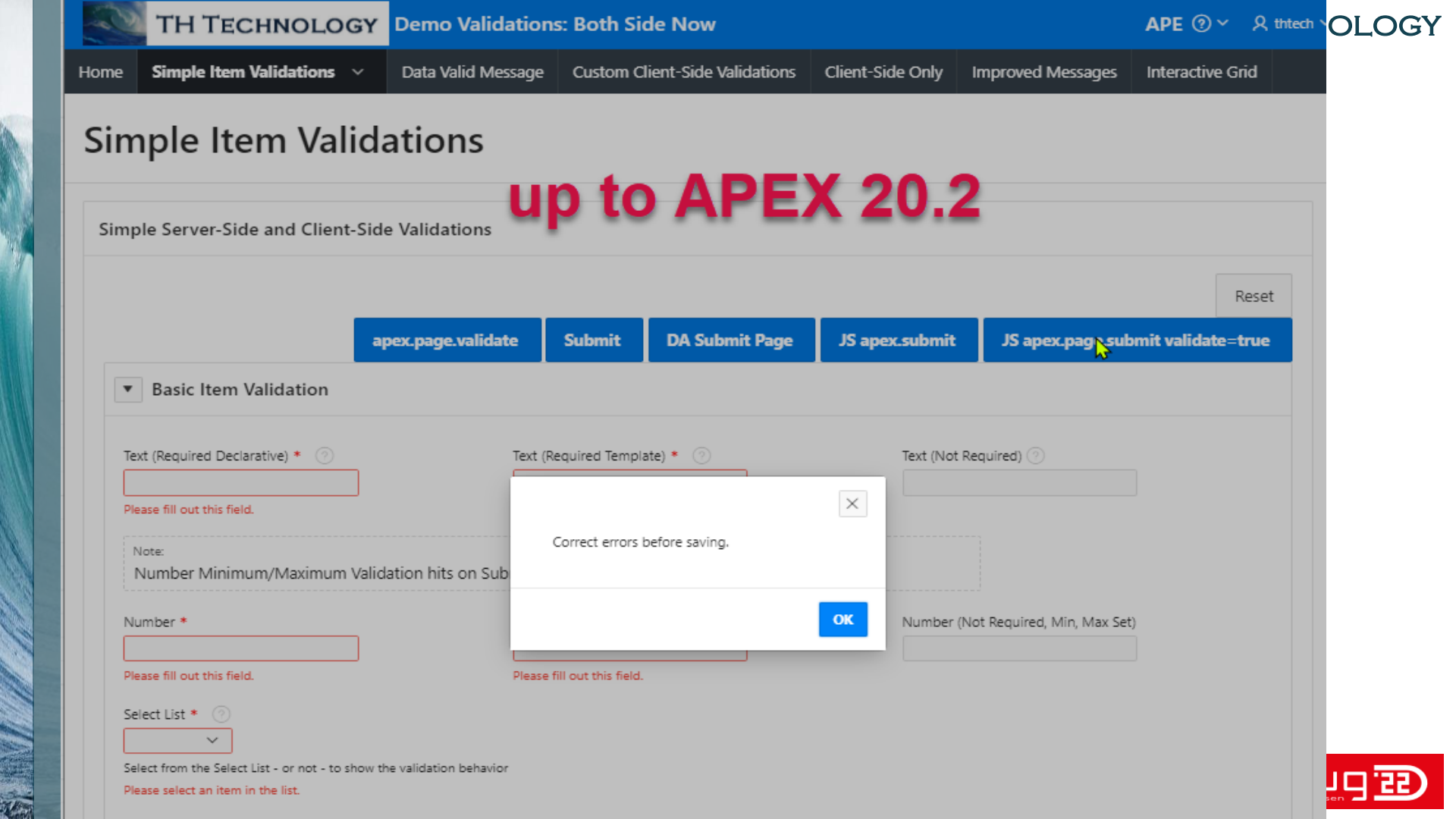
Validations

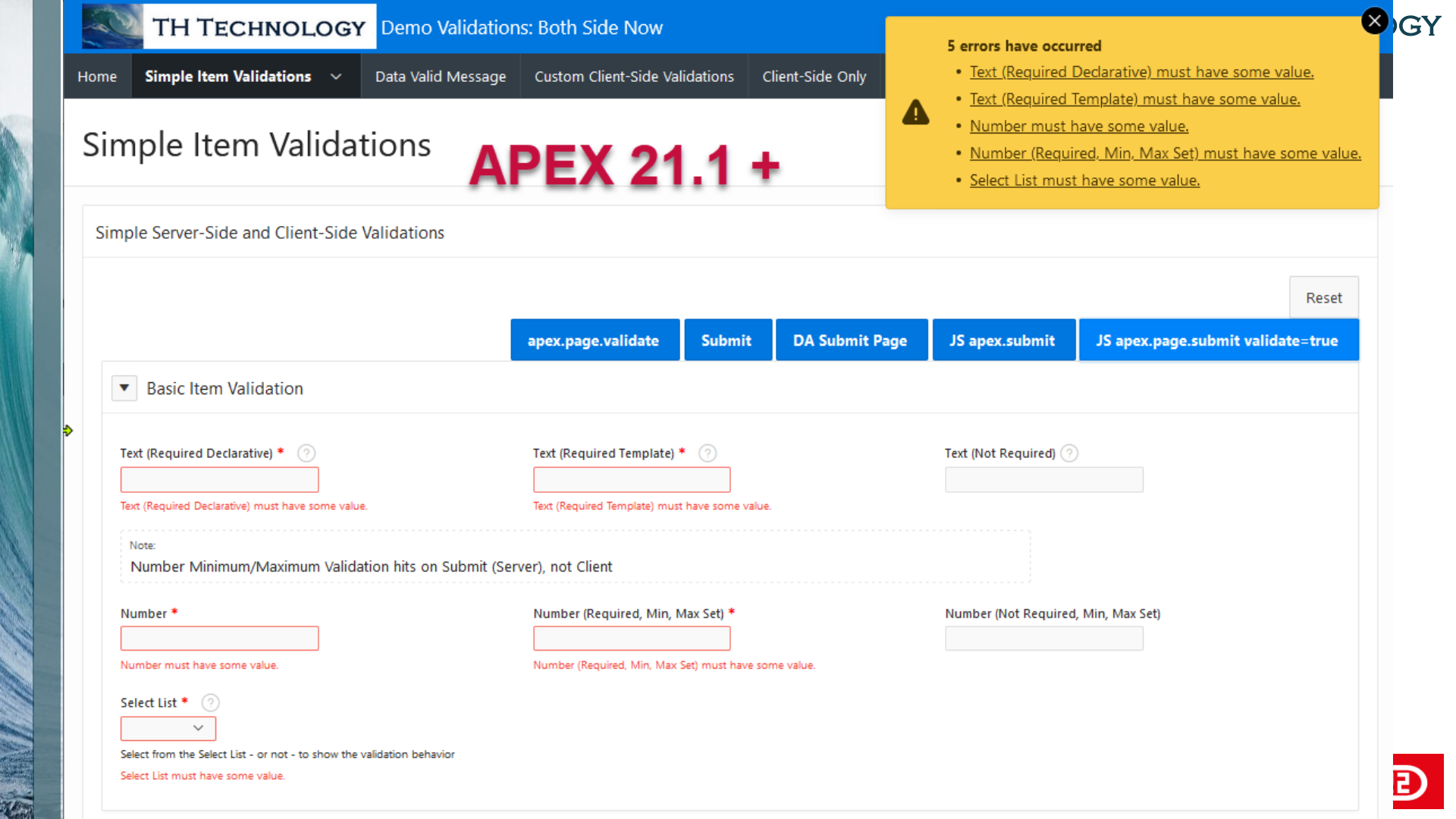
- Client, Server, Database
  - → Cover All Sides
  - → Have a Strategy
- Know How APEX Does Validations
  - Version to Version

Developer Responsibility

Know Your Version

- Read the Release Notes
- Things Change!


TH TECHNOLOGY
Demo Validations: Both Side Now
APE
thtech
Home
Simple Item Validations
Data Valid Message
Custom Client-Side Validations
Client-Side Only
Improved Messages
Interactive Grid
Simple Item Validations
up to APEX 20.2
Simple Server-Side and Client-Side Validations
Reset
apex.page.validate
Submit
DA Submit Page
JS apex.submit
JS apex.page.submit validate=true
Basic Item Validation
Text (Required Declarative) *
Please fill out this field.
Text (Required Template) *
Text (Not Required)
Note:
Number Minimum/Maximum Validation hits on Sub
Correct errors before saving.
OK
Number *
Please fill out this field.
Please fill out this field.
Number (Not Required, Min, Max Set)
Select List *
Select from the Select List - or not - to show the validation behavior
Please select an item in the list.


TH TECHNOLOGY
Demo Validations: Both Side Now
Home
Simple Item Validations
Data Valid Message
Custom Client-Side Validations
Client-Side Only
5 errors have occurred
Text (Required Declarative) must have some value.
Text (Required Template) must have some value.
Number must have some value.
Number (Required, Min, Max Set) must have some value.
Select List must have some value.
Simple Item Validations
APEX 21.1 +
Simple Server-Side and Client-Side Validations
Reset
apex.page.validate
Submit
DA Submit Page
JS apex.submit
JS apex.page.submit validate=true
Basic Item Validation
Text (Required Declarative) *
Text (Required Declarative) must have some value.
Text (Required Template) *
Text (Required Template) must have some value.
Text (Not Required)
Note:
Number Minimum/Maximum Validation hits on Submit (Server), not Client
Number *
Number must have some value.
Number (Required, Min, Max Set) *
Number (Required, Min, Max Set) must have some value.
Number (Not Required, Min, Max Set)
Select List *
Select from the Select List - or not - to show the validation behavior
Select List must have some value.

Why Validate?

Data Quality

Why Validate?

- Data Quality
- Prevent SQL Injection
- User Experience
  - Faster Response for User
  - Preserve Client State

Where Validate?

1. Database
   - Data Types
   - Keys
   - Constraints

Use Your Database!

Consider: a Flag Column

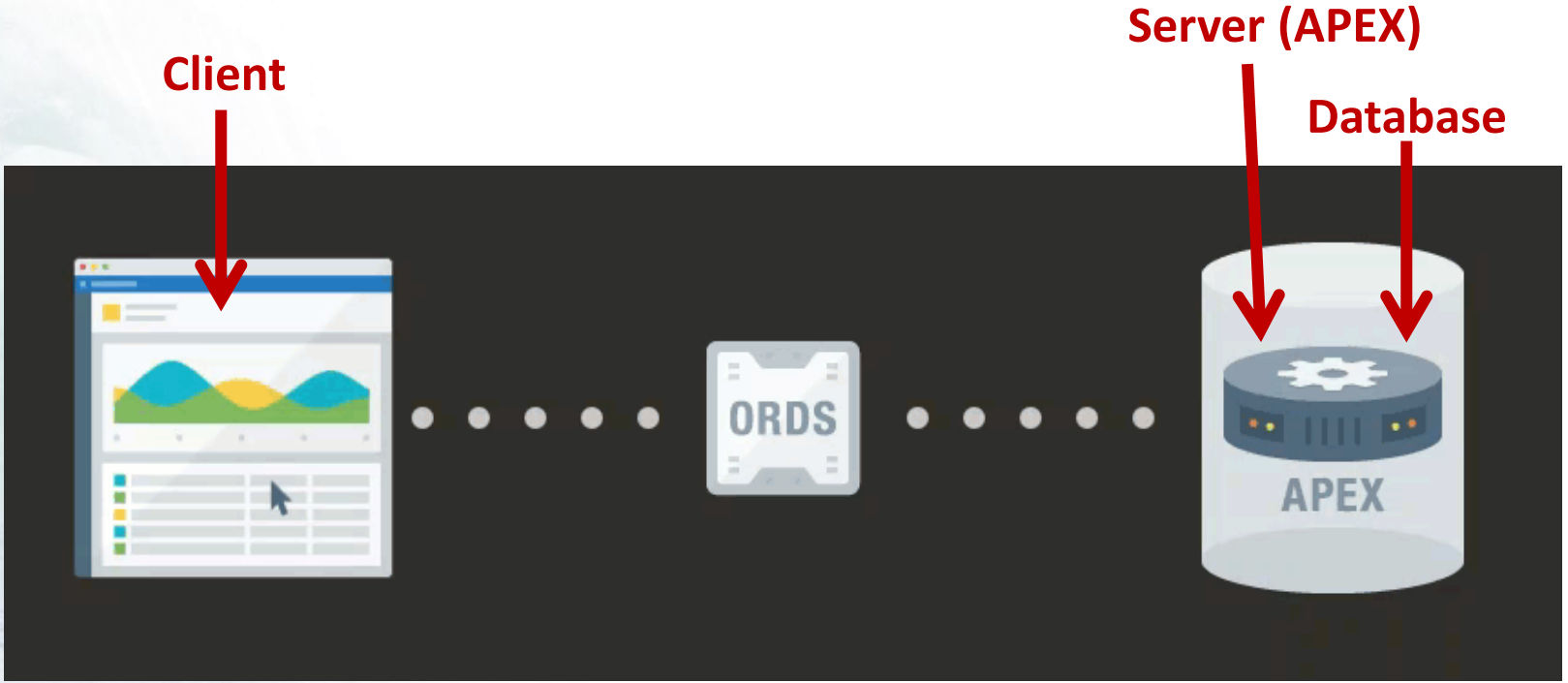
Y/N? Y/N/NULL? 1/0? Yes/No?
CHAR(1) *and* Check Constraint?

Where Validate?

1. Database
   - Data Types
   - Keys
   - Constraints
2. Server – On Submit
3. Client – Upon Entry

Use Your Database!

APEX Architecture



from apex.oracle.com/platform/architecture

What Are Validations?

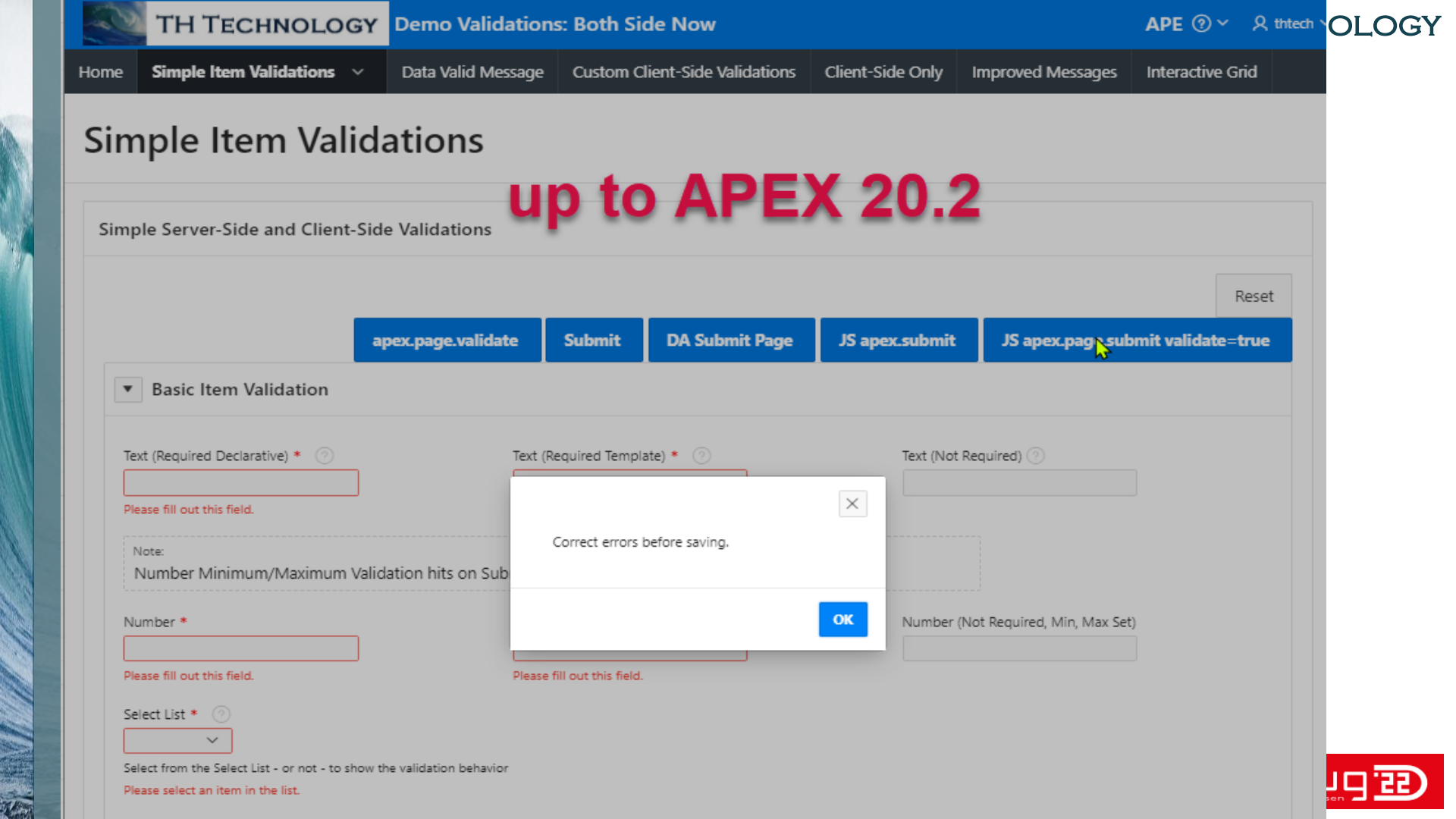
Data Checks In Your APEX App

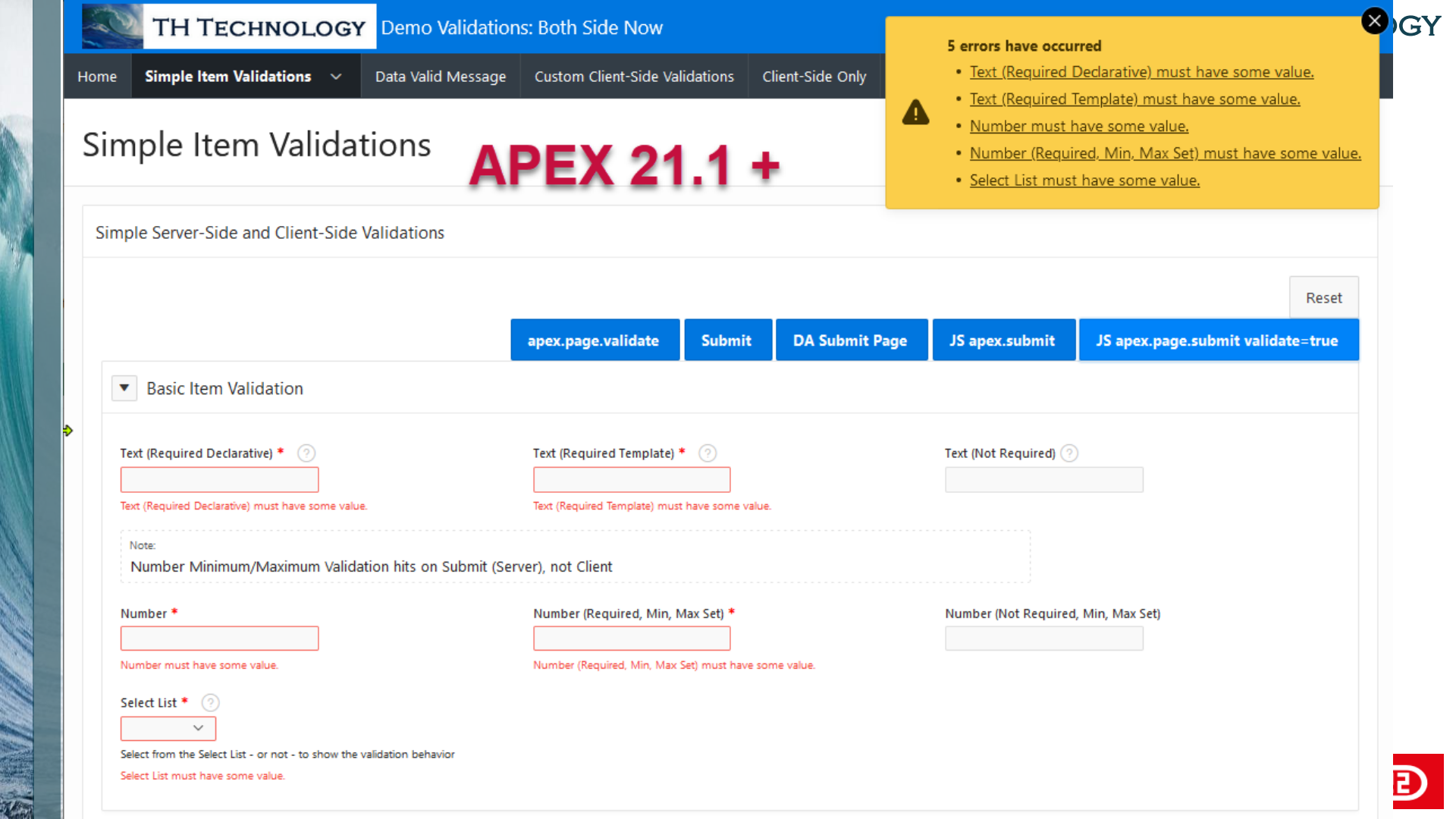
- Page Item
- Page (Multiple Page Items)
- Column
- Row (Multiple Columns)

APEX Client-Side Validation

- Compatibility Mode 5.1+

"... buttons where the Execute Validations attribute is set to Yes also perform some client-side validations (such as item required checks) and will not submit the page until all issues are fixed"


TH TECHNOLOGY
Demo Validations: Both Side Now
APE
thtech
Home
Simple Item Validations
Data Valid Message
Custom Client-Side Validations
Client-Side Only
Improved Messages
Interactive Grid
Simple Item Validations
up to APEX 20.2
Simple Server-Side and Client-Side Validations
Reset
apex.page.validate
Submit
DA Submit Page
JS apex.submit
JS apex.page.submit validate=true
Basic Item Validation
Text (Required Declarative) *
Please fill out this field.
Text (Required Template) *
Text (Not Required)
Note:
Number Minimum/Maximum Validation hits on Sub
Correct errors before saving.
OK
Number *
Please fill out this field.
Please fill out this field.
Number (Not Required, Min, Max Set)
Select List *
Select from the Select List - or not - to show the validation behavior
Please select an item in the list.

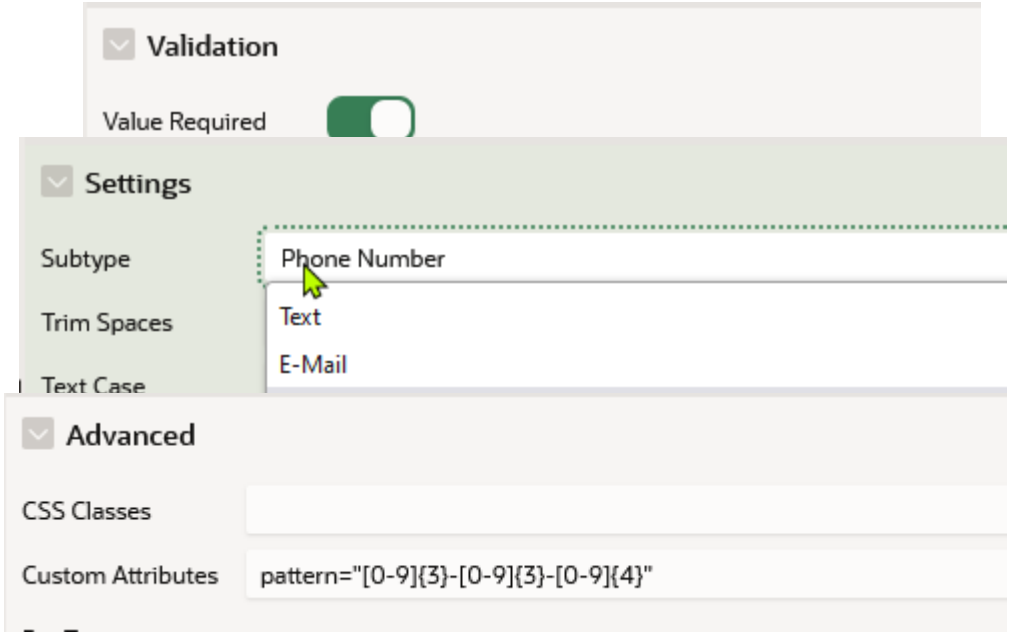

TH TECHNOLOGY
Demo Validations: Both Side Now
Home
Simple Item Validations
Data Valid Message
Custom Client-Side Validations
Client-Side Only
5 errors have occurred
Text (Required Declarative) must have some value.
Text (Required Template) must have some value.
Number must have some value.
Number (Required, Min, Max Set) must have some value.
Select List must have some value.
Simple Item Validations
APEX 21.1 +
Simple Server-Side and Client-Side Validations
Reset
apex.page.validate
Submit
DA Submit Page
JS apex.submit
JS apex.page.submit validate=true
Basic Item Validation
Text (Required Declarative) *
Text (Required Declarative) must have some value.
Text (Required Template) *
Text (Required Template) must have some value.
Text (Not Required)
Note:
Number Minimum/Maximum Validation hits on Submit (Server), not Client
Number *
Number must have some value.
Number (Required, Min, Max Set) *
Number (Required, Min, Max Set) must have some value.
Number (Not Required, Min, Max Set)
Select List *
Select from the Select List - or not - to show the validation behavior
Select List must have some value.

Client-Side Validation in APEX

- HTML5, Built-In
- JavaScript

Built-In Client-Side Validation (HTML5)

- Required
- Max Length
- Min/Max
- Type
  - Text -> Subtype
- Pattern



HTML5 Validity Concepts

- validity
- setCustomValidity
  getCustomValidity
- data-valid-message

Validation / Validity APEX JS APIs

- apex.page.validate
- apex.page.submit({validate:true;});
- apex.page.confirm({validate:true;})

- apex.item getValidity
- apex.item.setCustomValidity
- apex.message

Example JS Validate

```
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  }
24
```

apex.item

Example JS Validate

```
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  }
24
```

apex.item
Validity

Example JS Validate

```
var textitem = apex.item(this.triggeringElement.id),
    errors = [],
    alphanum = /^[0-9a-zA-Z]+$/;
    val = textitem.getValue();

// Perform Validation: Check if alphanumeric
 if (val.length > 0 && val.match(alphanum) ) {
    textitem.node.setCustomValidity(""); // valid
 } else {
    textitem.node.setCustomValidity("Invalid"); // rely on data-valid-message attribute to give a message
 }

// Raise/clear errors based on validity
 if (!textitem.getValidity().valid) {
    errors.push({
        message: textitem.getValidationMessage(),
        location: "inline",
        pageItem: textitem.id
    });
    apex.message.showErrors(errors);
 } else {
    apex.message.clearErrors(textitem.id);
 }
```

apex.message for the uniform message format

Demo: Very Simple Item Validations

Let's Take a Look ...

Validation Strategy

Have One!

Where You Validate Matters

- Client Side
  - Feedback *Before* Submit
  - A Few Declarative Settings
  - Dynamic Actions
- Server Side
  - Feedback *After* Submit
  - Lots of Declarative Options

Use Both!

Consider the Use Case

How You Submit Matters

- Submit
- Dynamic Action → Submit Page
- apex.submit()
- apex.page.submit ({validate:true;})

Know the Difference

Be Consistent

Don't Leave Holes

Do Not Fire Client Side

APEX 21.1 makes it all LOOK the same

How You Say It Matters

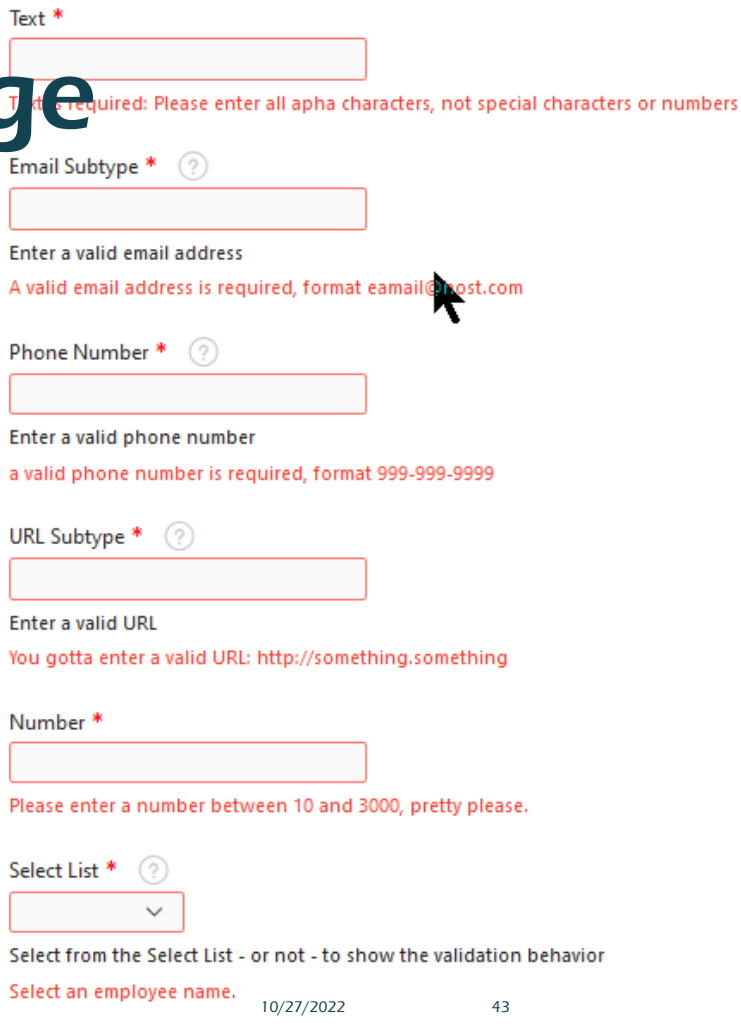
- Default Error Messages
- data-valid-message
- setCustomValidity
- apex.message

Be Informative

Be Consistent!

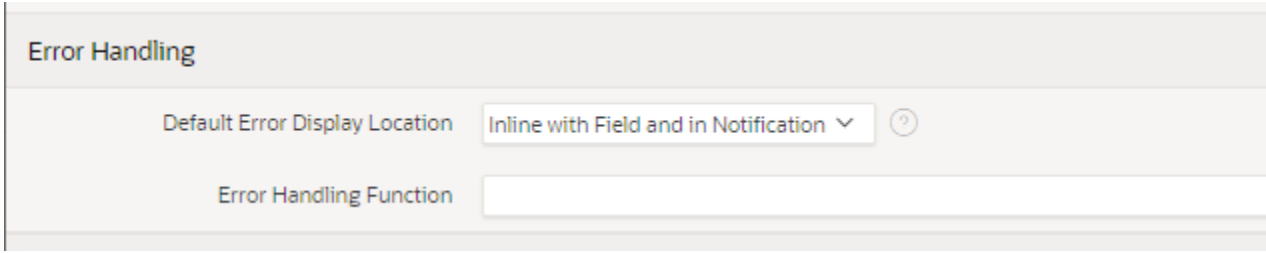
data-valid-message

- Use to Enter a Custom Error Message
  - Instead of "Please fill in <value>."



Where You Say It Matters

- Inline
- Inline and Notification EASIER w 21.1!!!!
- Notification
- Error Page



Be Informative

Be Consistent!

Declarative Validations

Do It Declaratively First ...

Declarative Validations (Easy) Stuff…

- Item Types
  - Text, Number, Date Picker
    - Minimum, Maximum, Format
  - Select List, Popup LOV, Shuttle, Radio Group
- Text Subtypes
  - Email
  - Phone
  - URL

  → Don't Forget to Supplement w a pattern="…"

Use the Declarative (Easy) Stuff...

- Value Required
- Templates
  - Required
  - Optional

Use Conditions to Control When ...

- Validations – (Server Side)
  - Server Side Conditions
  - Client Side Conditions
- Dynamic Actions (Client Side)
  - Server Side Conditions
  - Client Side Conditions

Server-Side Validations

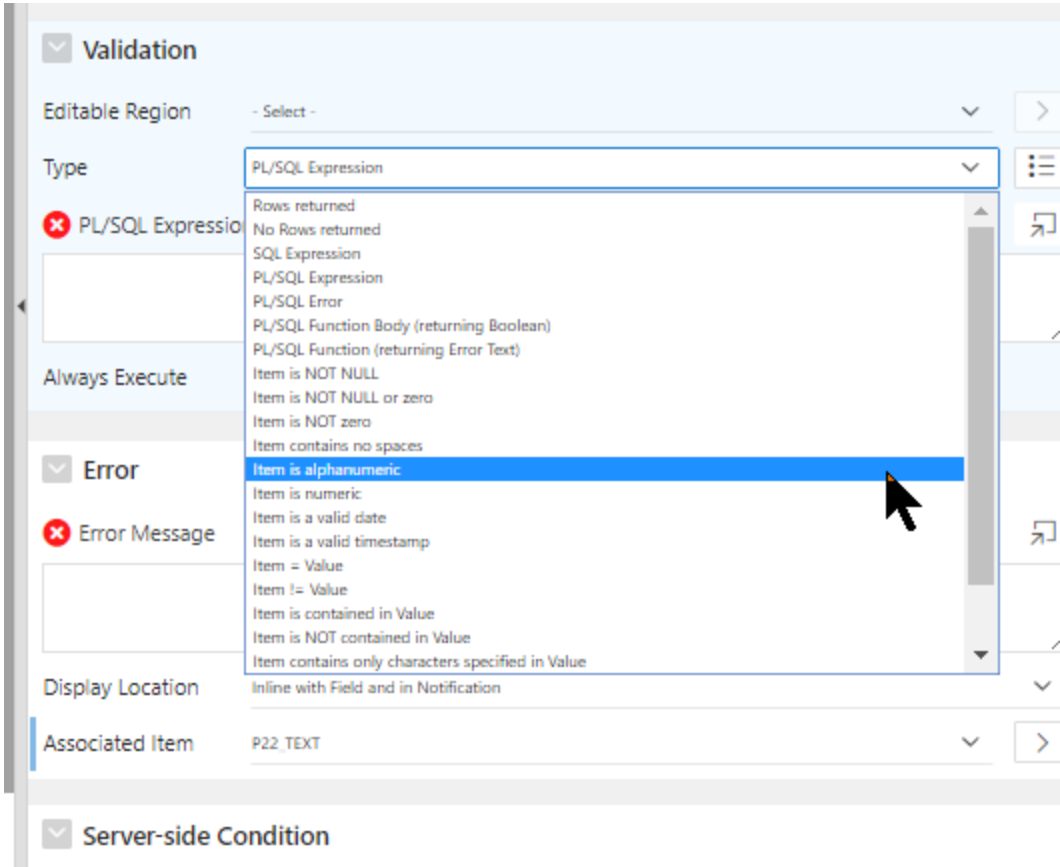
Our Standard APEX Validations ...

Server Side Validations

- Fire On Submit
  - IF all Client Side Validations are OK
- Many Declarative Types
- Many Declarative Conditions

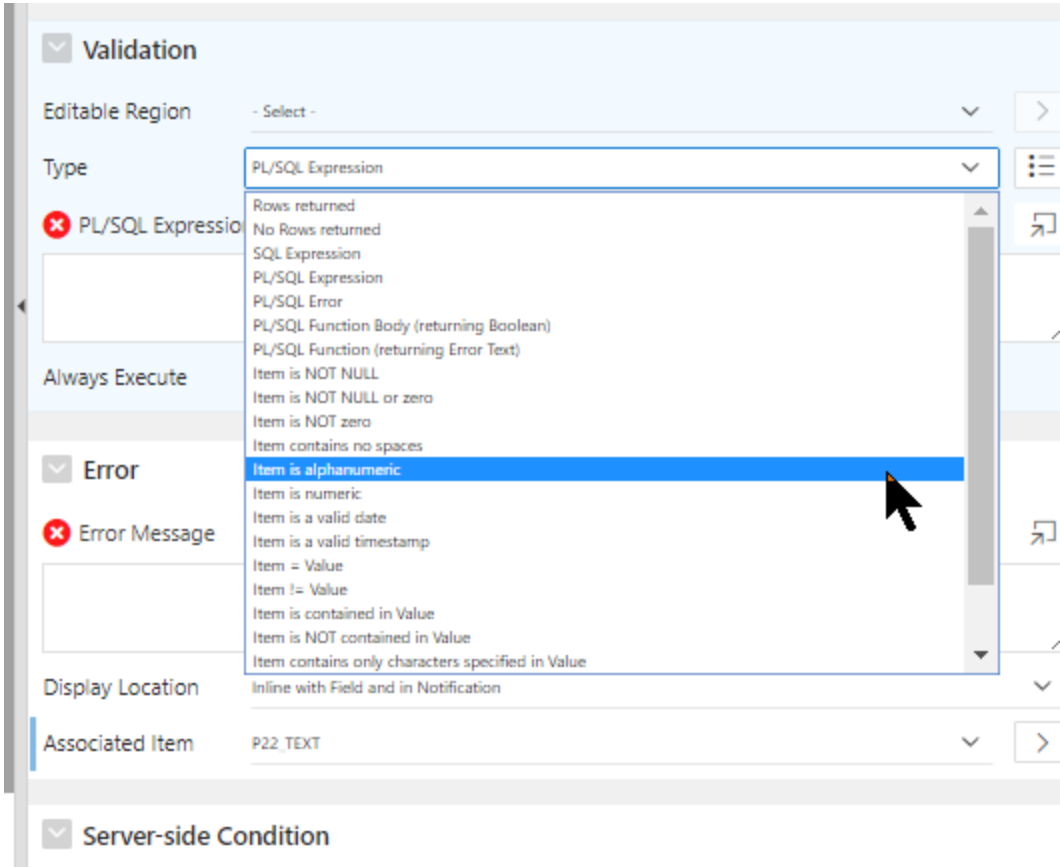
Creating a Server-Side Validation

- Simple Page Item
- Many Validation Types!



Creating a Server-Side Validation

- Use Declarative Types First
- PLSQL Expression, Function, etc. Last



Validation Conditions

- Server-Side, When Button Pressed
- Server-Side, Condition
- Client Side, JS Expression

Use The Conditions!
→ Simplify Your Validation Code

Use The Validation Attributes

- Sequence
- Editable Region (If Validating Item in a Grid)
- Type
- Always Execute

Validation: Error Attributes

- Error Message – Your Consistent Error Message
- Display Location
  - Inline w Field and In Notification
  - Inline w Field
  - Inline in Notification
  - On Error Page
- Associated Item

Client-Side Validations

via Declarative Settings and Dynamic Actions ...

Client-Side Validations

- Declarative HTML5
  - Required
    - Type: Email URL Phone Search
    - Pattern
    - data-valid-message and related validity JS APIs
- Dynamic Action
  - Execute JavaScript

Client-Side All the Time?

- Client-Side Always on Submit?
  - DA, on Page Zero, Before Page Submit
    - Execute JavaScript

    apex.page.validate();

Validate via Dynamic Actions

- On Change vs. On Lose Focus
  - vs Your Requirements
- Use Conditions on the Dynamic Actions

Improve the Message

Use the JS APIs to Tailor the Message and Be Consistent

Improve the Message*

This is the Part You Don't Have To Do as of APEX 21.1

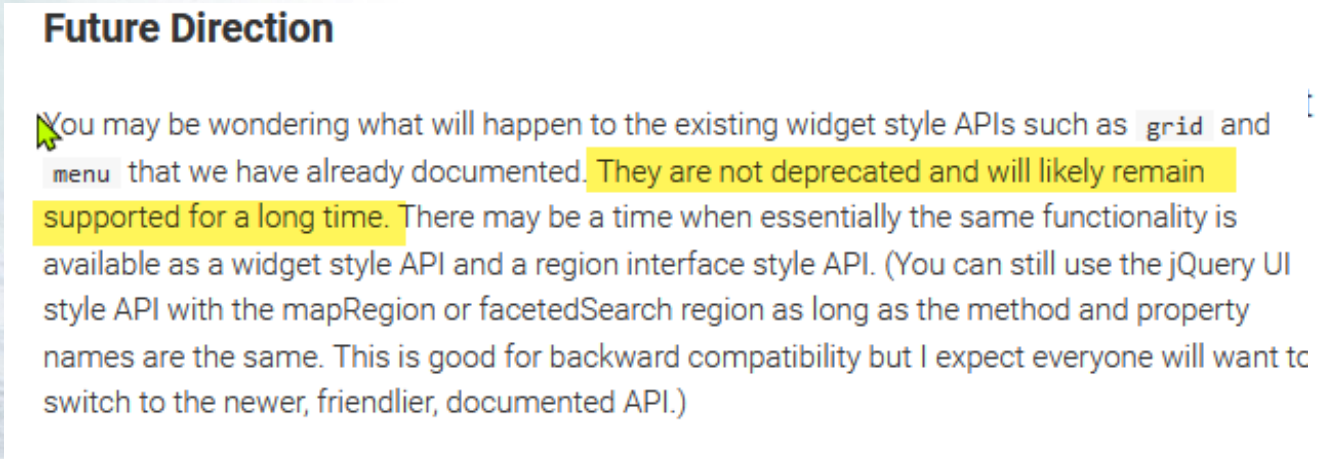
Use the JS APIs to Tailor the Client-Side Message to Look Like the Server Side Message (Be Consistent)

Use the Documented JS APIs

- apex.oracle.com/api → JavaScript APIs
  - apex.item and item
  - apex.message
  - apex.model
  - Interactive Grid APIs

Note on JS API Syntax as of APEX 21.2

- As of APEX 21.2, Moving Away from JQuery UI Widget Syntax to Simpler, Intuitive JS API Syntax...



Future Direction

You may be wondering what will happen to the existing widget style APIs such as `grid` and `menu` that we have already documented. They are not deprecated and will likely remain supported for a long time. There may be a time when essentially the same functionality is available as a widget style API and a region interface style API. (You can still use the jQuery UI style API with the mapRegion or facetedSearch region as long as the method and property names are the same. This is good for backward compatibility but I expect everyone will want to switch to the newer, friendlier, documented API.)

Example JS Validate

```
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  }
24
```

apex.item

Example JS Validate

```
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  }
24
```

apex.item
Validity

Example JS Validate

```
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  }
24
```

apex.message for the uniform message format

Too Much JavaScript?

- If Lots of JS in Dyn Actions ... CONSIDER Putting All in a JS File
  - More JS Control
  - Less Declarative

Validate Your Validations: Part II

Grids
Strategy
Thoughts?

Validation in Interactive Grids

Apply the Same Server-Side/Client Side Strategy ...

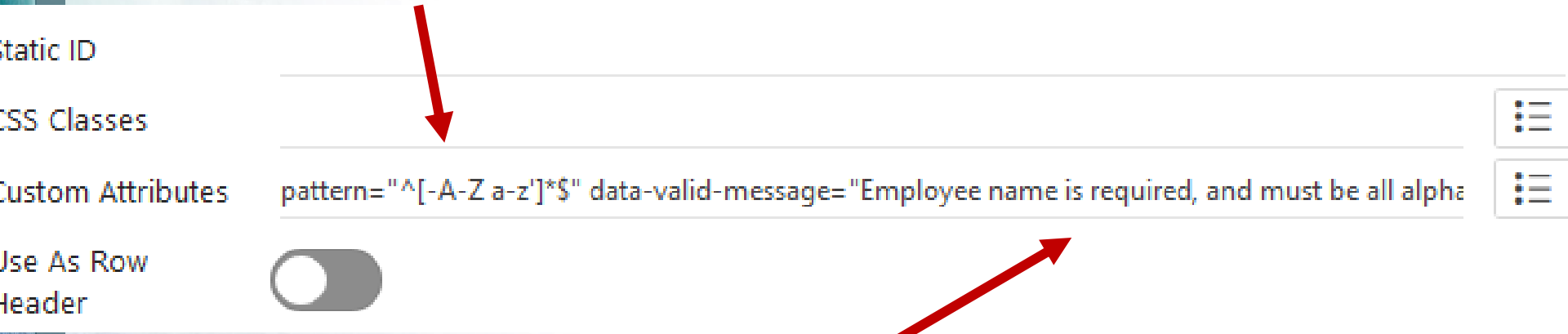
In General

- Each Column Is An Item
- Validations and Dynamic Actions Work the Same as Page Items
- :COLUMN_NAME Bind Reference

The Same Stuff Applies!

Grid Validation - EName

- EName Must Be Alpha

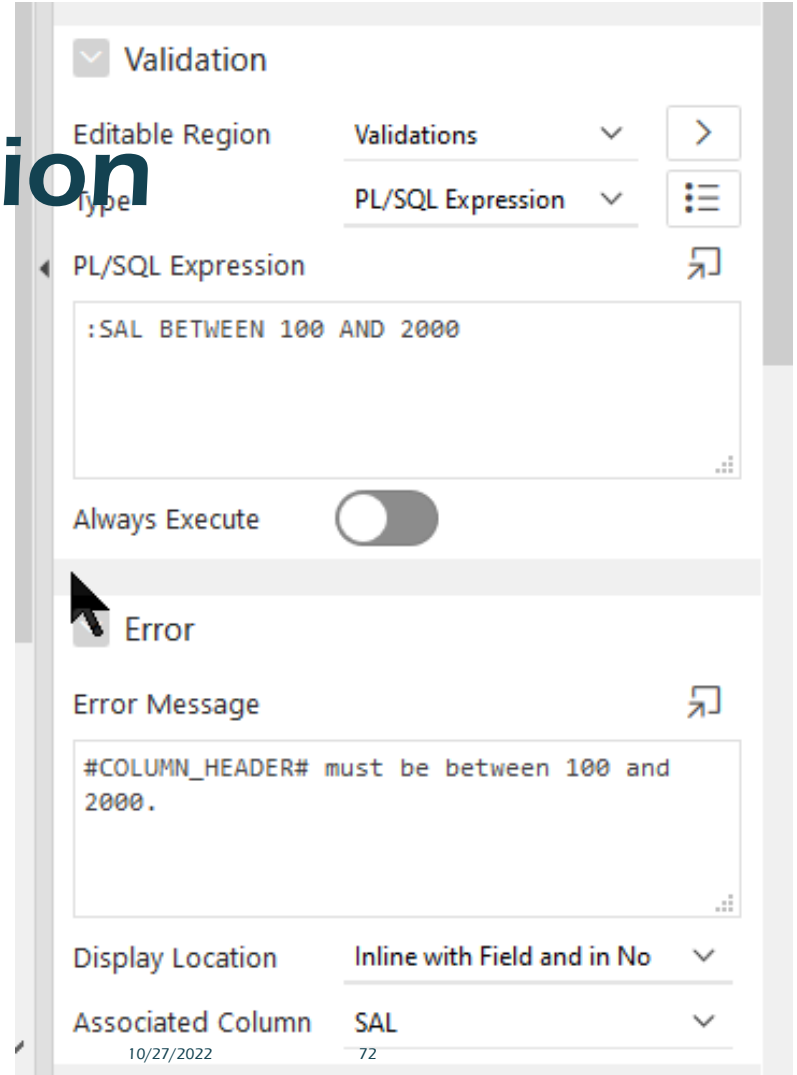


- Custom Message

Example Grid Validation

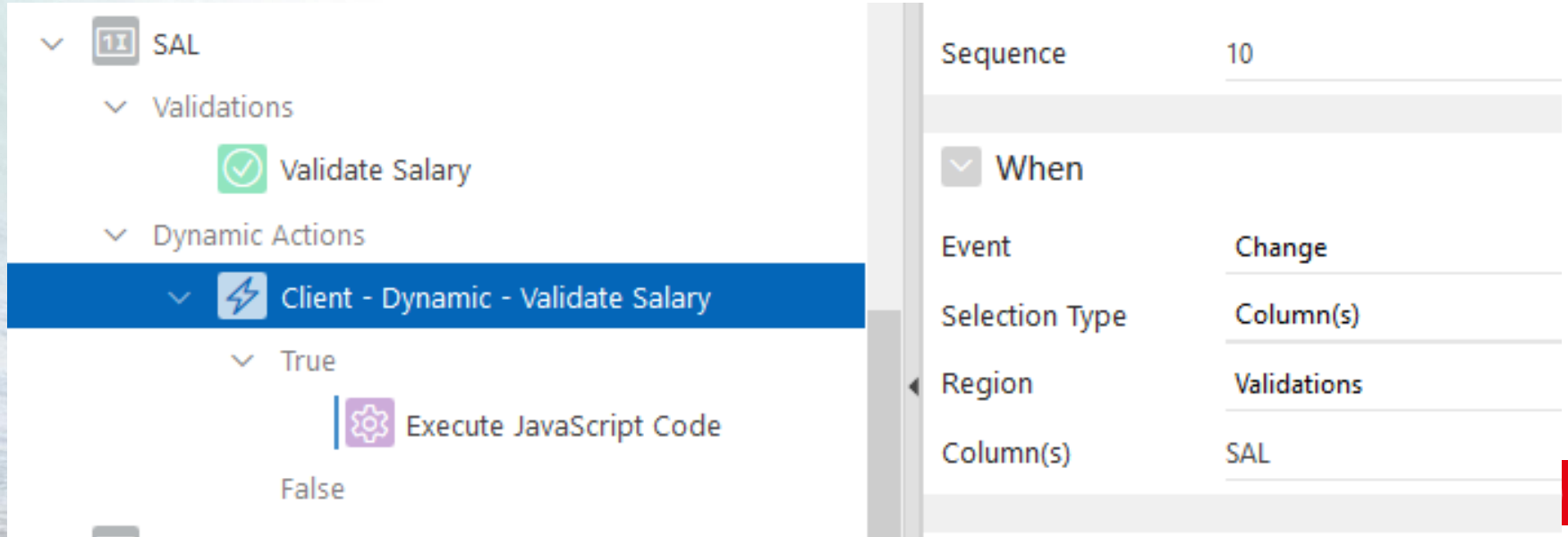
- Server-Side

Salary Between 100 and 2000



Client-Side Grid Validation - Item

- Dynamic Action
  - On Change



Grid Validation - Salary

- Client-Side Salary Betw100 and 3000

```
var sal = apex.item('salary'),
    num = sal.getValue();
if ( num !== ""
          && (parseFloat(num) != num || num < 100 || num > 3000)) {
    // this msg only used if there is no HTML
    //data-valid-message attribute on the column
    sal.node.setCustomValidity("Salary must be between 100 and 3000");
} else {
    sal.node.setCustomValidity(""); // clear the error
}
```

Grid Validation - Salary

- Client-Side Salary Betw100 and 3000

Code Editor - Code

```
1  var sal = apex.item('salary'),
2      num = sal.getNativeValue();
3  if ( num !== "" && ( num < 100 || num > 3000)) {
4      // this msg only used if there is no HTML data-valid-message attribute on the column
5      sal.node.setCustomValidity("Salary must be between 100 and 3000");
6  } else {
7      sal.node.setCustomValidity(""); // clear the error
8  }
```

`getNativeValue() → {number}`

Return the current value of the item as a JavaScript number. If the value is not a valid number returns NaN. This is useful when working with numbers because the item#getValue method always returns a string.

Returns:

Type
number

Example

In this example, page items P1_COST and P1_TAX are added together and then formatted and displayed in an alert dialog

```
var total = apex.item( "P1_COST" ).getNativeValue() + apex.item( "P1_TAX" ).getN
apex.message.alert( "Total is: " + apex.locale.formatNumber( total, "L999G999G9
```

Grid Validation – Dyn Action Salary

- Salary Betw100 and 3000

Grid Validation – Dyn Action Salary

- Salary Betw100 and 3000

```
var sal = apex.item('salary'),
    num = sal.getValue();
if ( num !== ""
          && (parseFloat(num) != num || num < 100 || num > 3000)) {
    // this msg only used if there is no HTML
    //data-valid-message attribute on the column
    sal.node.setCustomValidity("Salary must be between 100 and 3000");
} else {
    sal.node.setCustomValidity(""); // clear the error
}
```

Grid Validation – Dyn Action Salary

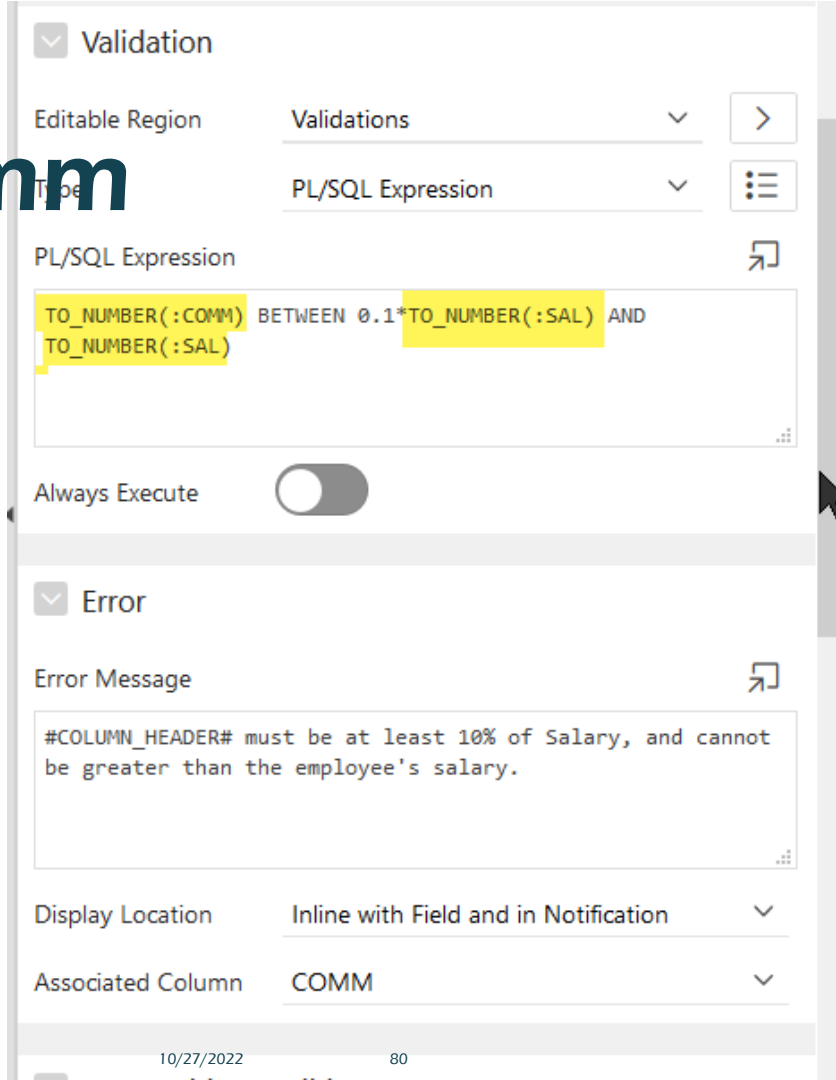
- Salary Betw100 and 3000

```
var sal = apex.item('salary'),
    num = sal.getNativeValue();
if ( num !== "" && ( num < 100 || num > 3000)) {
    // this msg only used if there is no HTML data-valid-message attribute on the column
    sal.node.setCustomValidity("Salary must be between 100 and 3000");
} else {
    sal.node.setCustomValidity(""); // clear the error
}
```

Grid Validation - Comm

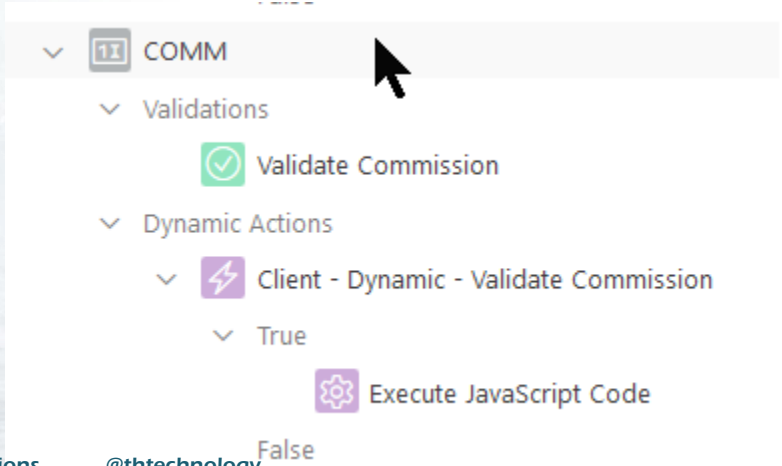
- Server-Side - Row

At Least10% of Salary and Not > Salary



Grid Validation – Client Side -Row

- Dynamic Action
  - On Change
    - Execute JavaScript



Grid Validation – DA - Commission

- 10% Sal < Commission < Salary

```
var comm = apex.item("commission"),
    sal = apex.item("salary"),
    numComm = comm.getValue(),  // same as $v("static id")
    numSal = sal.getValue();

if ( numComm !== "" &&
    (parseFloat(numComm) != numComm || numComm < 0.1*numSal || numComm > numSal)) {
    // this message used iff no data-valid-message attribute on the column item
    comm.node.setCustomValidity("Commission must be at least 10% of Salary, and cannot be more than
} else {
    comm.node.setCustomValidity(""); // clear the error
}
```

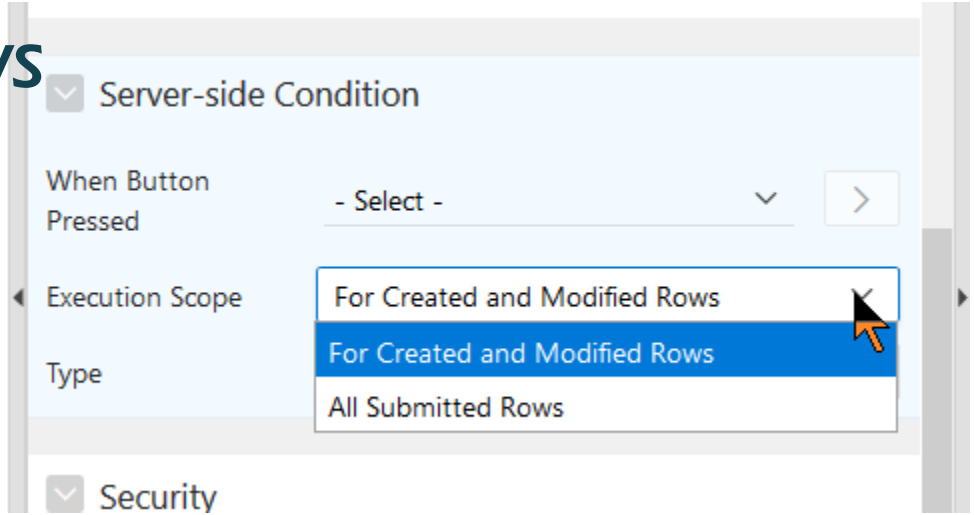
Grid Validation – DA - Commission

- 10% Sal < Commission < Salary

```
var comm = apex.item("commission"),
    sal = apex.item("salary"),
    numComm = comm.getNativeValue(),  // get native JS number, the parseFloat stuff not needed
    numSal = sal.getNativeValue();
//alert( numComm );
//alert ( 0.1*numSal);
if ( numComm !== "" &&  (numComm < 0.1*numSal )) {
   // (parseFloat(numComm) != numComm || numComm < 0.1*numSal )) {
    // this message used iff no data-valid-message attribute on the column item
    comm.node.setCustomValidity("Commission must be at least 10% of Salary.");
} else {
    comm.node.setCustomValidity(""); // clear the error
}
```

Why Did My Grid Validation Not Fire?

- Grid Validation Settings
  - Created and Modified Rows
  - All Submitted Rows



"Submitted Rows"
... Not All Rows Get Submitted!

Grids Are Designed to Validate **Submitted Rows**

- What Is Your Use Case?
- Do you really want validations to fire on *All* rows?
- How Many is *All*?

All Submitted Rows ?

- Do you really NEED validations to fire on *all* rows?
- "Touch" Each Row
  - Add a Dummy Column, Touch, Hide That
- Right Before the "Save" – Best Time ?

General Validation Strategy

Cover *All* the Bases ...

General Data Validation Strategy

- Have a Data Model
- Use Keys
- Use Constraints
- Be Consistent !
  - Same Checks
  - Same Informative Messages
  - Help Gives Same Messages

General APEX Validation Strategy

- Do It Declaratively First
- Create Server Side
- Create Client Side
- Be Consistent !
  - Same Validations
  - Same Informative Messages
  - Help Gives Same Messages

That's A Lot of Work!

There's a Plugin for That ...

United Codes Plugins Pro

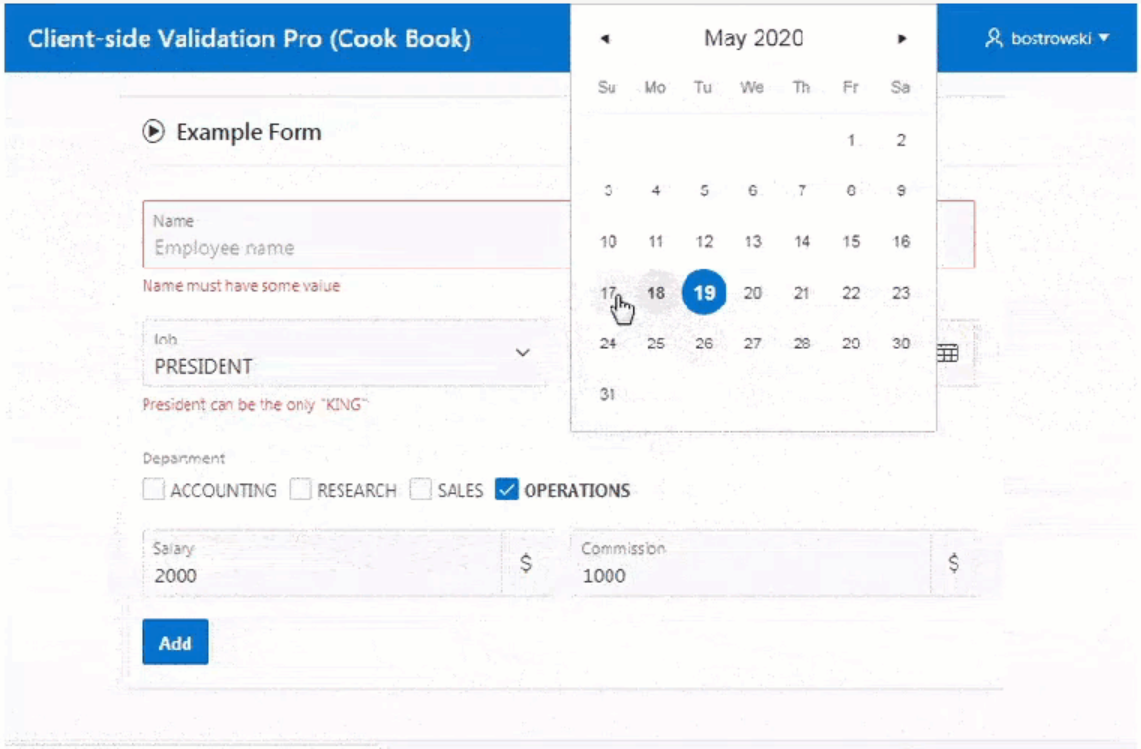
Others?

Validations Plugin ~ Free Version!

THINK: Should This Be Automatic?

Client-side Validation Pro

Native APEX validations with instant results!



More Validations == More Testing

- Check for Requirements
  - Is All Validated That Should Be?
- Check Everything Fires When It Should
  - Validation Holes?
- Check User Experience

Validations ~ Summary

- Database Design – Final Data Check
- Server Side – Final App Check
- Client Side – Immediate Feedback

Take The Time to

Secure All Fronts!

Some APEX Validations Homework ...

APEX Validations ~ Homework

- References
- Sample DB App
- Sample Grids App
- IG Cookbook
- Grids – Learn the APIs

- **Know What Fires When and Why**
- Adjust the Samples For Your Requirements

Standardize

Stay Sane

Keep Your Users Happy

Know Your Tool
Use Your Tool

Know How APEX Does Validations

Version to Version

Questions?

Thank You

Karen Cannell
kcannell@thtechnology.com
@thtechnology

Email for Demo App & Slides

Validations Resources

- Oracle APEX Documentation

  https://docs.oracle.com/en/database/oracle/application-express/20.2/htmdb/validating-user-input-in-forms.html#GUID-8BA81FD5-E138-494D-9F8E-35D5A370E31A

- APEX 21.1 New Features
- J Snyders, Client Side Validation

  https://hardlikesoftware.com/weblog/2017/05/10/apex-client-side-validation/

- M Mulvaney Client Side Validation

  https://explorer.co.uk/client-side-validations/

- M Mulvaney Grid Validations

  https://explorer.co.uk/the-numerous-options-for-apex-18-1-interactive-grid-validations/

- Translate Client Side Messages

  http://apexbyg.blogspot.com/2017/02/apex-51-translate-html5-client-side.html

Validations Resources

- J Snyders, APEX 21.2 New JS APIs

  https://hardlikesoftware.com/weblog/2021/10/11/apex-21-2-new-javascript-apis-joelkallmanday/APEX 21.1 New Features

- J Snyders, Interactive Grid Validation

  Interactive Grid Validation – HardLikeSoftware

  https://hardlikesoftware.com/weblog/2022/04/25/interactive-grid-validation

- …

  …

Where to Put JS Files

- https://www.talkapex.com/2017/01/how-to-reference-javsscript-and-css-files-for-entire-application/js-storage.gif

New

X 21.2

- ...

New Item Interfaces

For the longest time item interfaces returned by apex.item() all had the exact same methods and properties. Even if some of those methods such as `addValue` only applied to a very few item types. Starting a few release ago item plug-in implementations could extend the base methods and properties. So in the same way that the mapRegion extends the region interface we now have specific items like the `numberFieldItem` item that extend the item interface. A `numberFieldItem` is a more specific kind of item. It has the same methods and properties of an item except that the ones that don't apply are not documented/used.

When you call `apex.item("itemid")` you get back an item interface that is specific to the type of item it is. This was true even in release 21.1 where, for example, the interface for a Number Field item had the extra method `getNativeValue`. But now these extras are documented.

The additional method offered by the `numberFieldItem` is `getNativeValue` which returns a JavaScript number rather than a string like `getValue` does. This method is very useful if you want to do some client side math on the value of a Number Field.

The additional methods offered by the `colorPickerItem` are:

- `getNativeValue` which returns an object with properties for the red, green, blue, and alpha values and
- `contrastWith` which is used to compare the color value of this item with another color and return contrast information.

The still newish Date Picker item also has a `getNativeValue` method but there wasn't time to add it to the documentation.

For Demo Application:

https://github.com/thtechnology/APEXWorldValidations

kcannell@thtechnology.com
@thtechnology