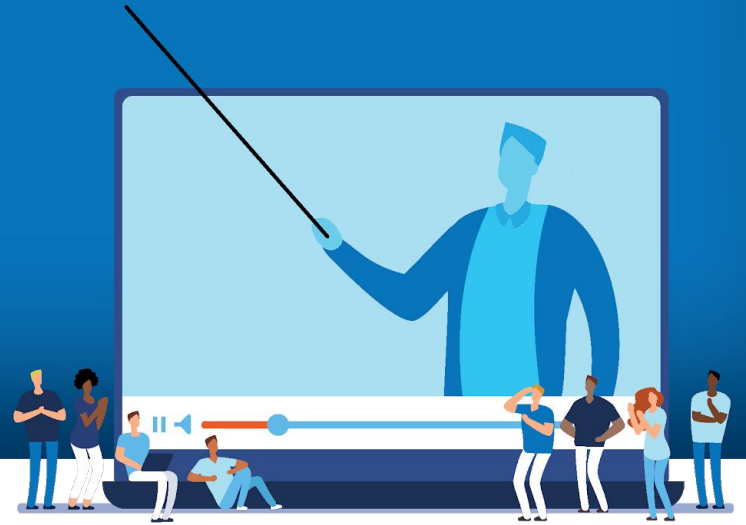


# Best Practices for Building Smart Controls







*Presenter:*  
**Jonathan Hsu**



*Moderator:*  
**Stella Park**



# GoToWebinar Control Panel

-  Hide/Collapse Control Panel
-  Microphone Status: Muted (red)
-  Make Webinar Full Screen
-  Raise Hand or Ask Presenter for Audio Rights

File View Help

Audio

Sound Check

Computer audio

Phone call

**MUTED**

Microphone (HD Webcam C510)

Speakers (High Definition Aud...)

Questions

[Enter a question for staff]

Send

Multi sessions different registrants

Webinar ID: 980-960-603

GoToWebinar

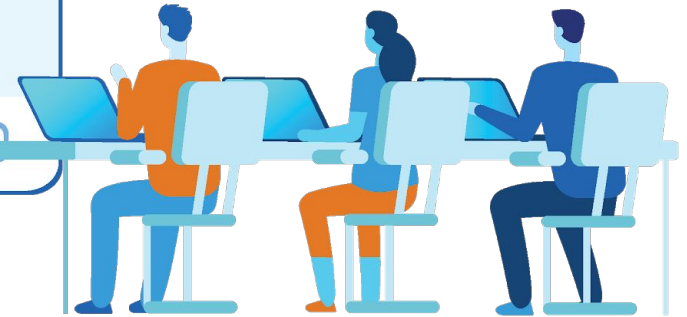
Type questions or comments anytime during the webinar here



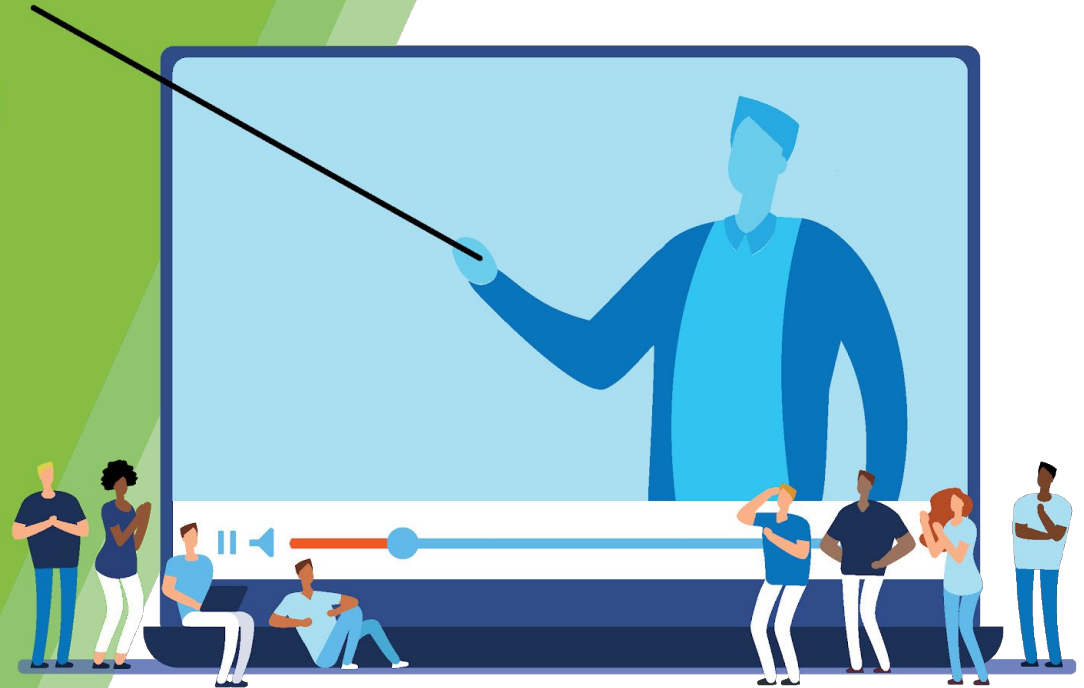
# POLL

# Agenda

1. How App Data is Stored
2. How to Avoid Data Caching
3. Referencing Subform Elements
4. Setting Up Page-Level Javascript
5. Configuring Lookup Element Filters



# How App Data is Stored and Avoiding Data Cache Issues



App data is stored in variables. If a Text element is named 'attendee', then there will be an 'attendee' variable. Record data is stored in a form variable that matches the name of the form.

If our form is named 'smart\_controls' then the first name field will exist in two places:

```
attendee
smart_controls.attendee
```

A form with attendee and age will have the following structure:

```
attendee
age
smart_controls.attendee
smart_controls.age
```

Why is this important?

Even though `attendee` and `smart\_controls.attendee` may have the same value, only one represents the true value of the record.

## Before and After submitting a record

attendee	=>	"Jonathan"	first_name	=>	"Jonathan"
age	=>	35	age	=>	35
registration.attendee	=>	"Jonathan"	registration.attendee	=>	
registration.age	=>	35	registration.age	=>	

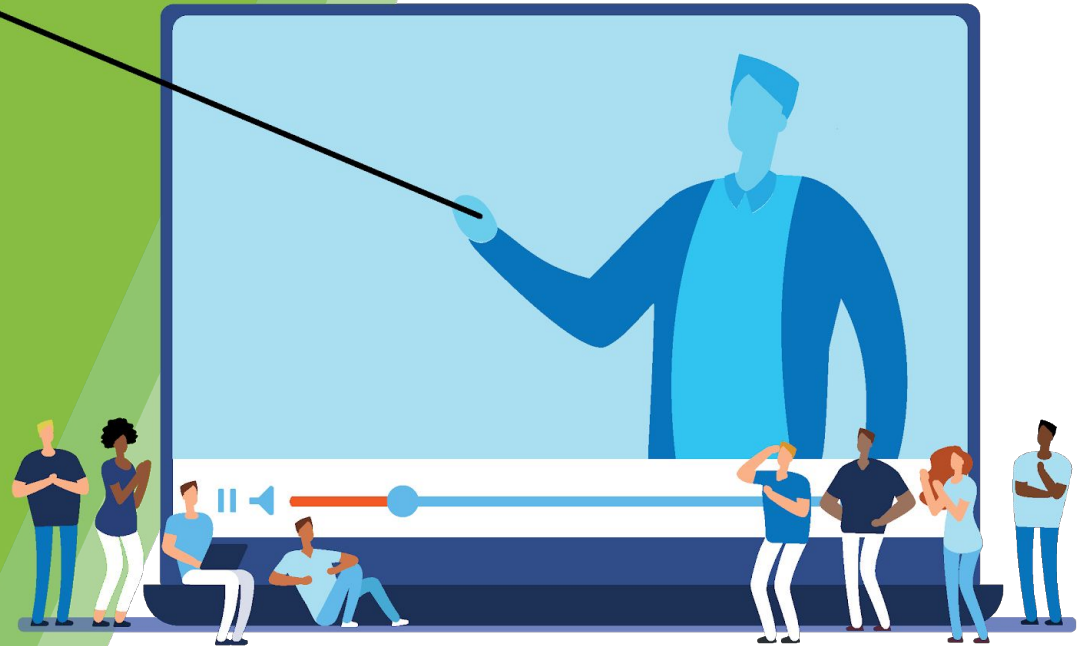
***To guarantee Smart Controls are referencing the current record's value, always use the full form path.***

In a multi-paging subform, each new record will overwrite first\_name.

score	=>	9	score	=>	9	score	=>	8
[0] score	=>	9	[0] score	=>	9	[0] score	=>	9
			[1] score	=>		[1] score	=>	8

Variables outside the form path are shared across the entire app — both record-to-record and even form-to-form.

# Referencing Subform Elements







# POLL

Subforms are organized as a list of records — also referred to as an array of objects.

Arrays use an integer to identify the arrangement of each object — called an index.

A single-paging subform is still an array. If you are the only person in line at the grocery store, you're still in line.

The index of an array is always a whole number, starting at zero and counting upwards.

***The first object in an array always has an index of zero.  
The last object in an array always has an index equal to the length minus 1.***

There are two ways to specify the index of a subform record:

### **Hard-coded integer**

```
registration.attendees[0].name
```

This method is simple, but limiting because the reference is to a fixed point.

Use this when passing a single-paging subform value to the parent or performing single-paging subform calculations.

### **The `.index` subform property**

```
registration.attendees[smart_controls.attendees.index].name
```

This method is longer, but necessary for dynamic references in multi-paging subforms

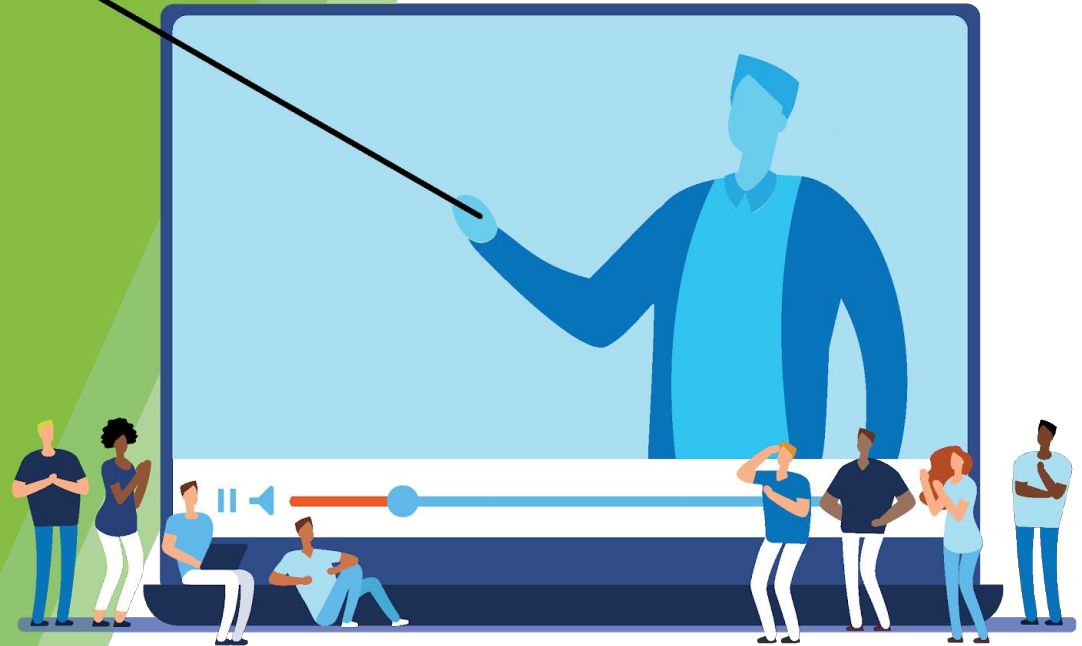
Use this when performing multi-paging subform calculations.

## What About Grandchild Forms?

Here's a strategy for complicated subform references.

1. Write the reference with empty square brackets  
`-registration.attendees[].emails[].address`
2. Copy everything up to the square bracket, then paste inside the square bracket  
`-registration.attendees[registration.attendees].emails[].address`
3. Add the `.index` property  
`-registration.attendees[registration.attendees.index].emails[].address`
4. Repeat for the next subform  
`-registration.attendees[registration.attendees.index].emails[registration.attendees[registration.attendees.index].emails].address`  
`-registration.attendees[registration.attendees.index].emails[registration.attendees[registration.attendees.index].emails.index].address`

# Setting Up Page-Level JavaScript





Page-Level JavaScript (PLJS) allows form builders to define functions that can be used in any of the Smart Control fields.



A function is a sequence of commands that begin with inputs and ends with a single output.



Use PLJS functions for complex, multi-step commands or calculations that are performed multiple times.



**POLL**

# Writing Lookup Filter Logic





The Lookup Element is the latest addition to the iFormBuilder toolkit. This element is the spiritual successor to Smart Table Search and allows for guided, advanced filtering of lookup records in the Form Builder.

Apply a Dynamic Attribute to begin defining a lookup filter.

**Assign lookup** **Clear lookup**

Dynamic Attributes

Filter

Filter

Restrict the selectable records based on criteria

Field Name	Condition
owner_name	Equal to

Field Value

owner\_text\_element

Cancel **Confirm**

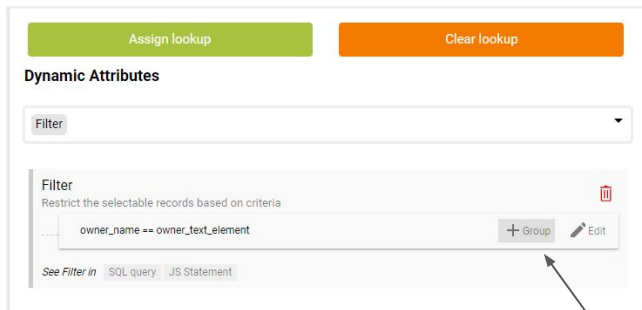
See Filter in [SQL query](#) [JS Statement](#)

**Lookup Data Field**

**Filter Condition**

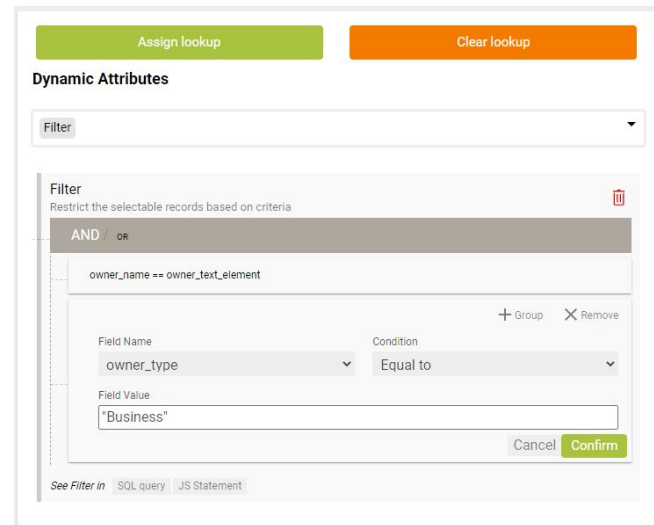
**Form Field**

Once the expression has been configured, the appropriate logic is saved.



To create more complex filters, convert a single expression to a group expression.

Group expressions are connected with either "AND" or "OR".





**POLL**

# THANK YOU!



/zerion



Zerion



@zerionsoftware