## Contents



# Goal of the Workshop

Guide participants through the process of creating a SAP Build Apps Application that can connect to the Advanced Event Mesh and demonstrate the ability to publish a message.

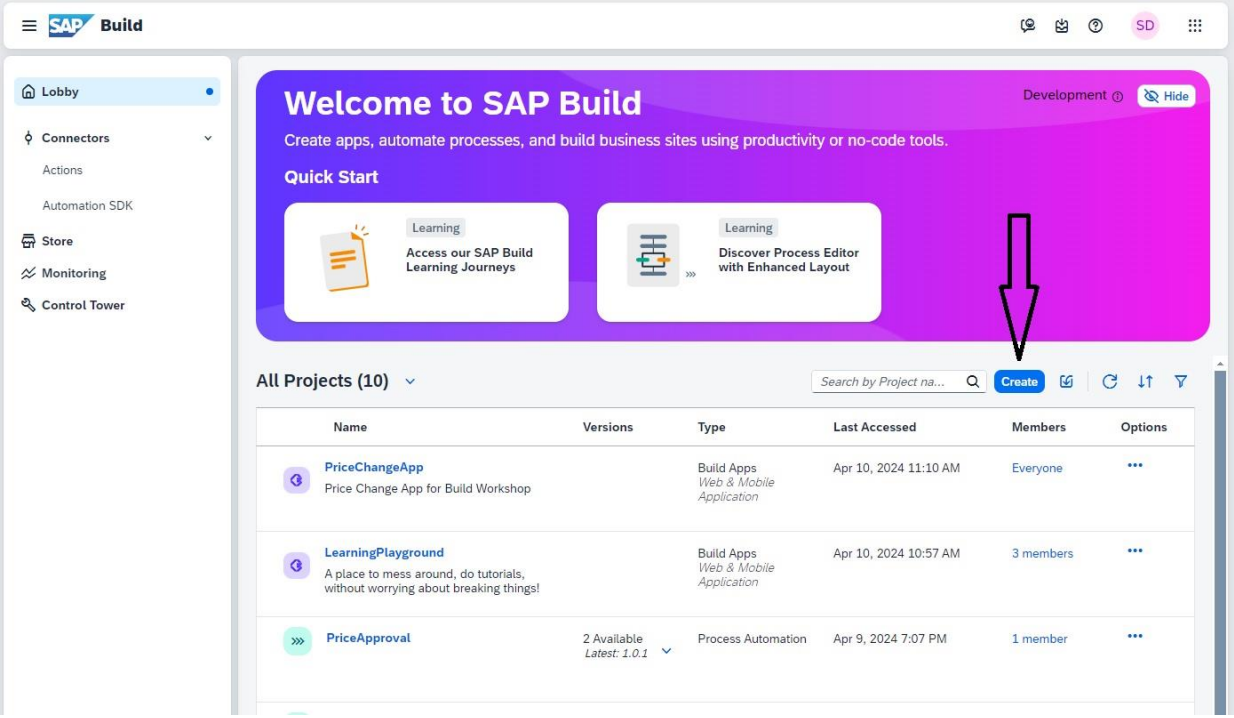
# Pre-requisites

Access to the SAP Build Apps environment and access to a Advanced Event Mesh Broker.

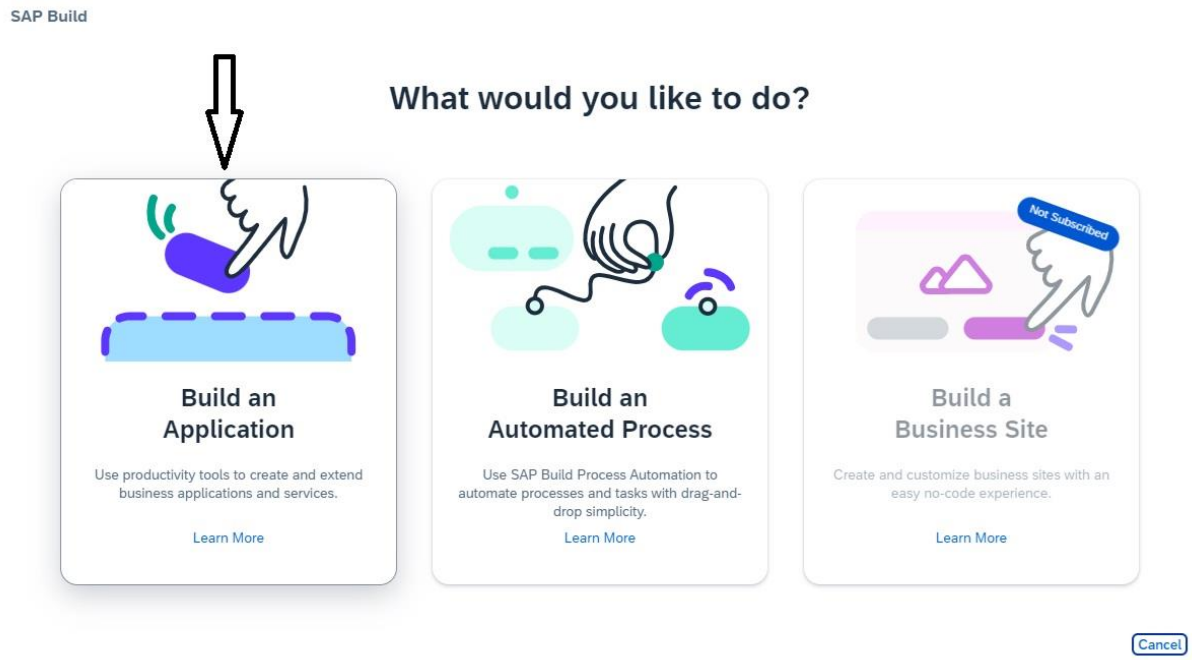
# Creating the Build App Project

## Step 1 – Create a new Build App Project

From the Lobby of the SAP Build, we will select the create option.

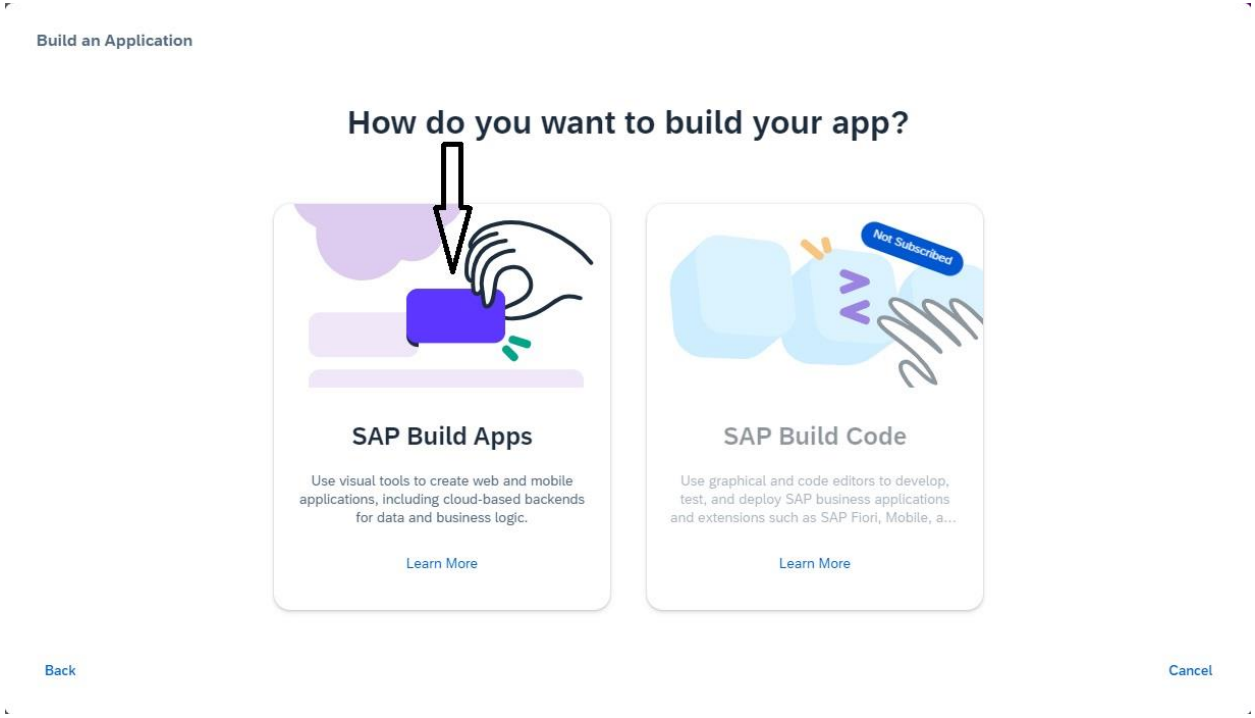


From there select “Build an Application”

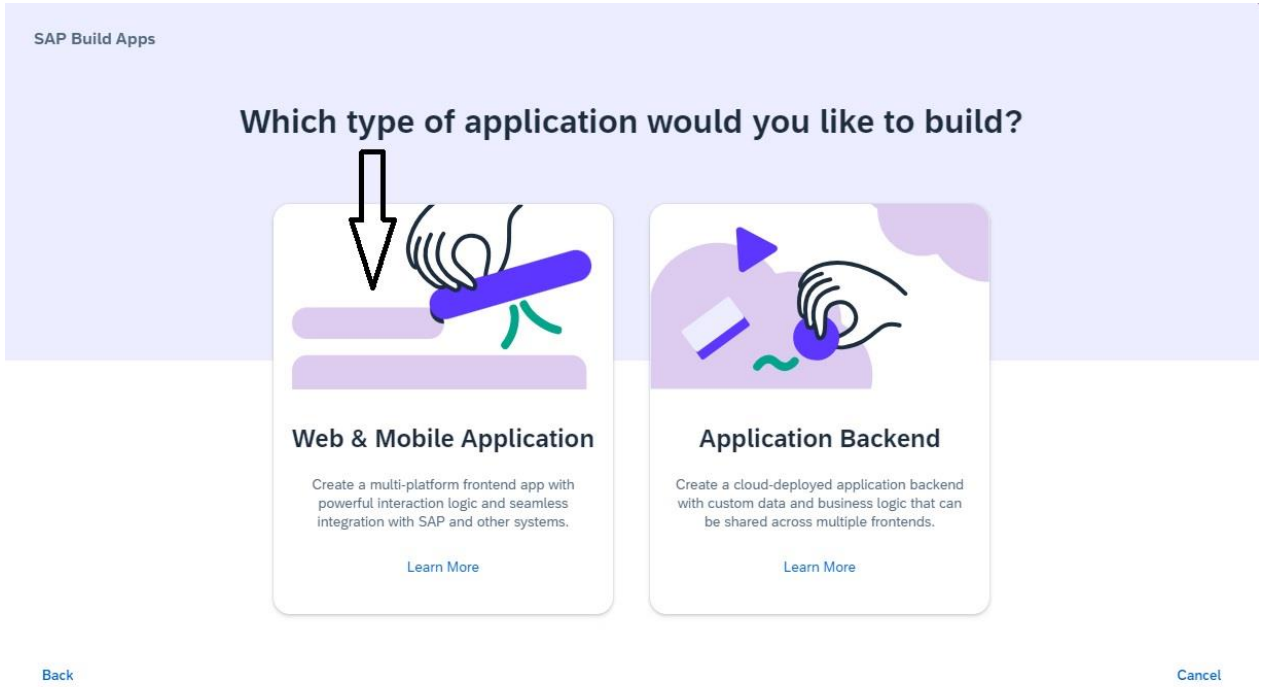


From there select “SAP Build Apps”

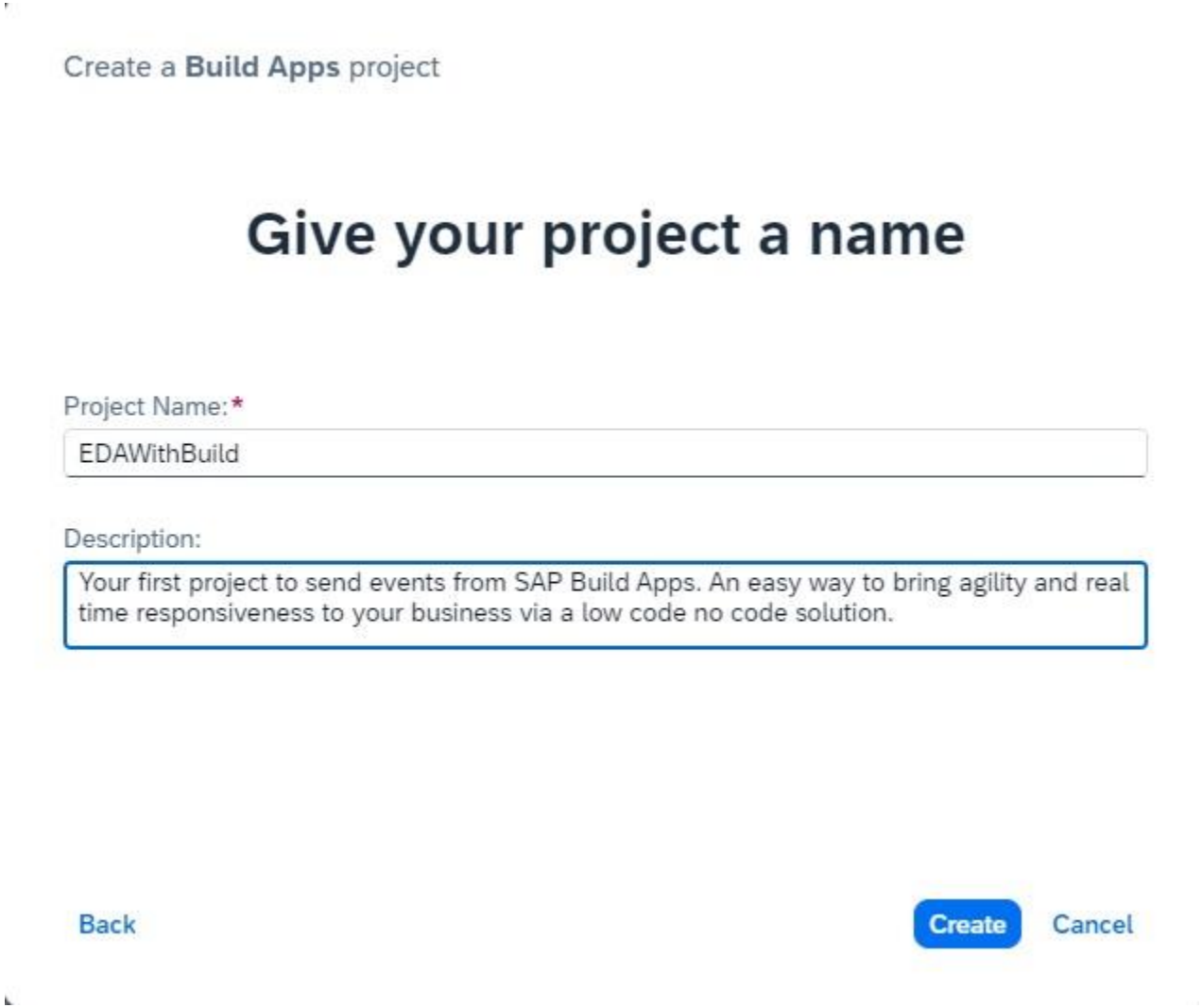
(***My option for SAP Build Code is grayed out because I have not activated this service***)



Select “Web & Mobile Application”

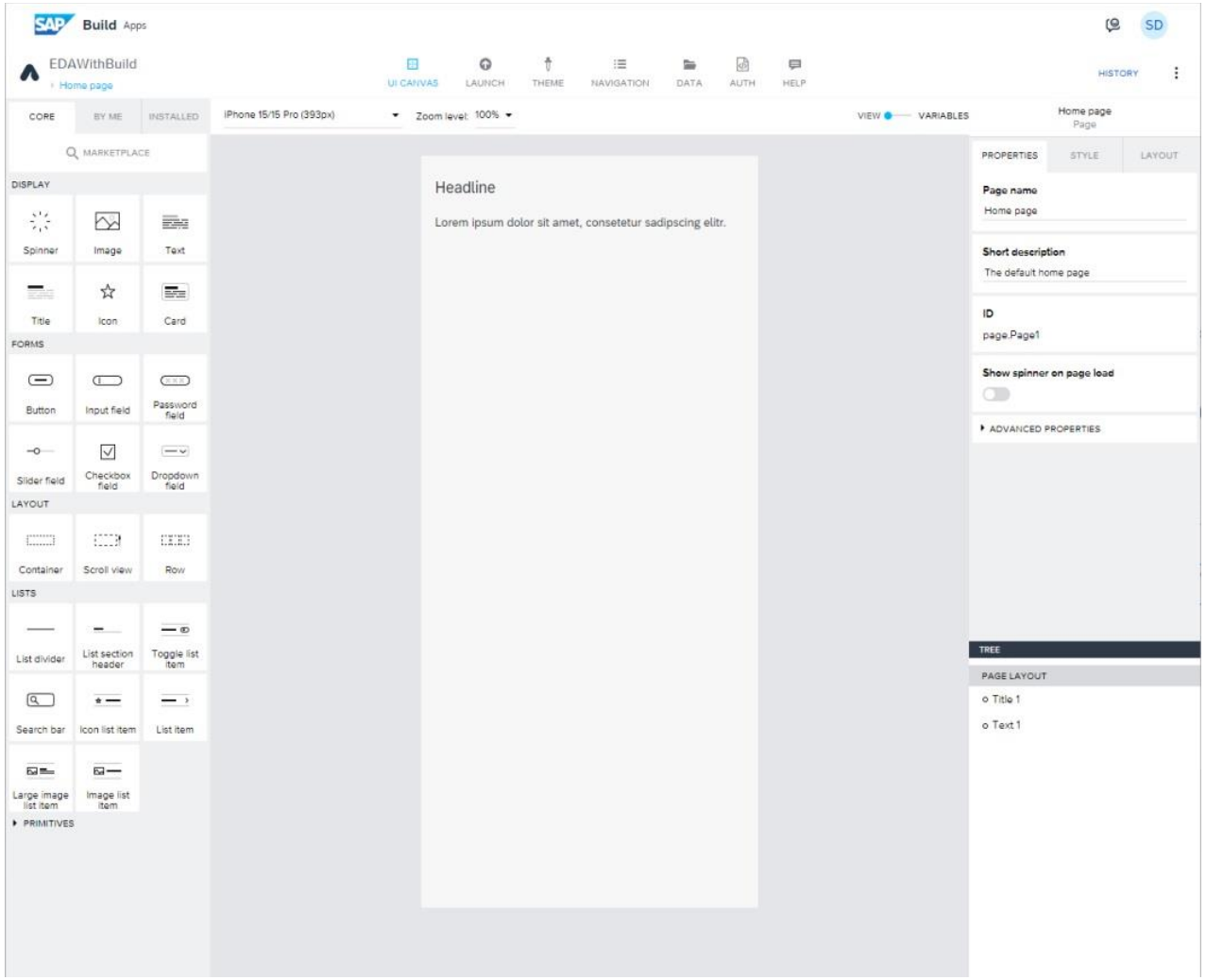


Give your name a project and an amazing description.



It may take a few minutes for the project to be created. When ready, click on the project in the list to open it.

Your starting point for the project should look like this.

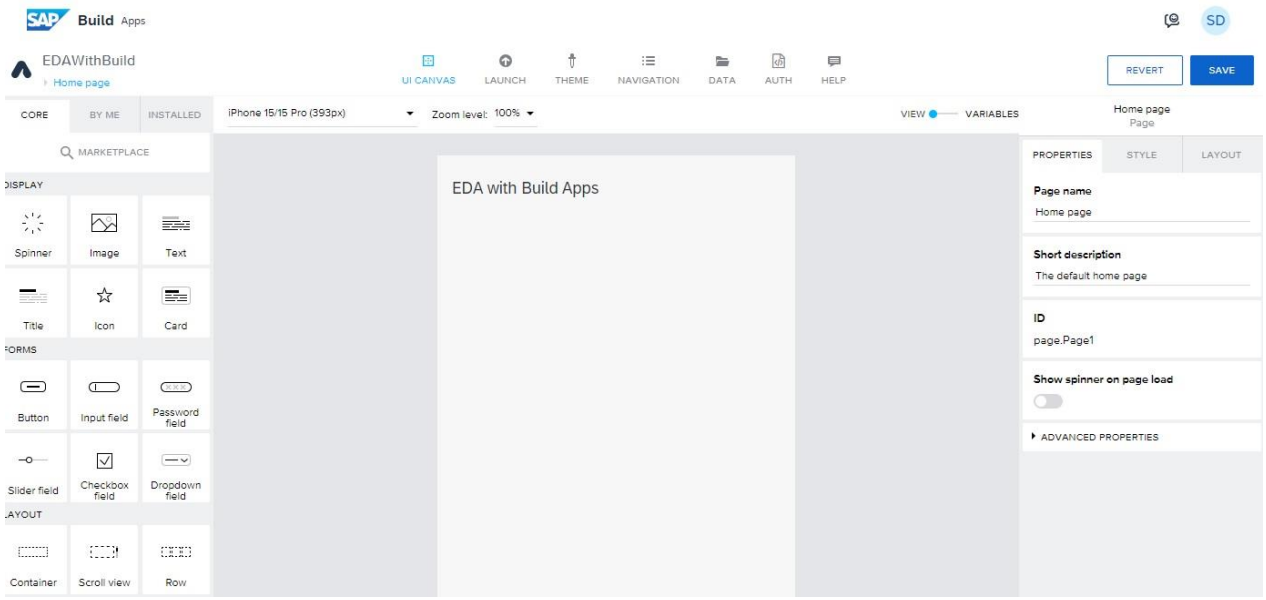


## Step 2 – Design the Product Pricing Form

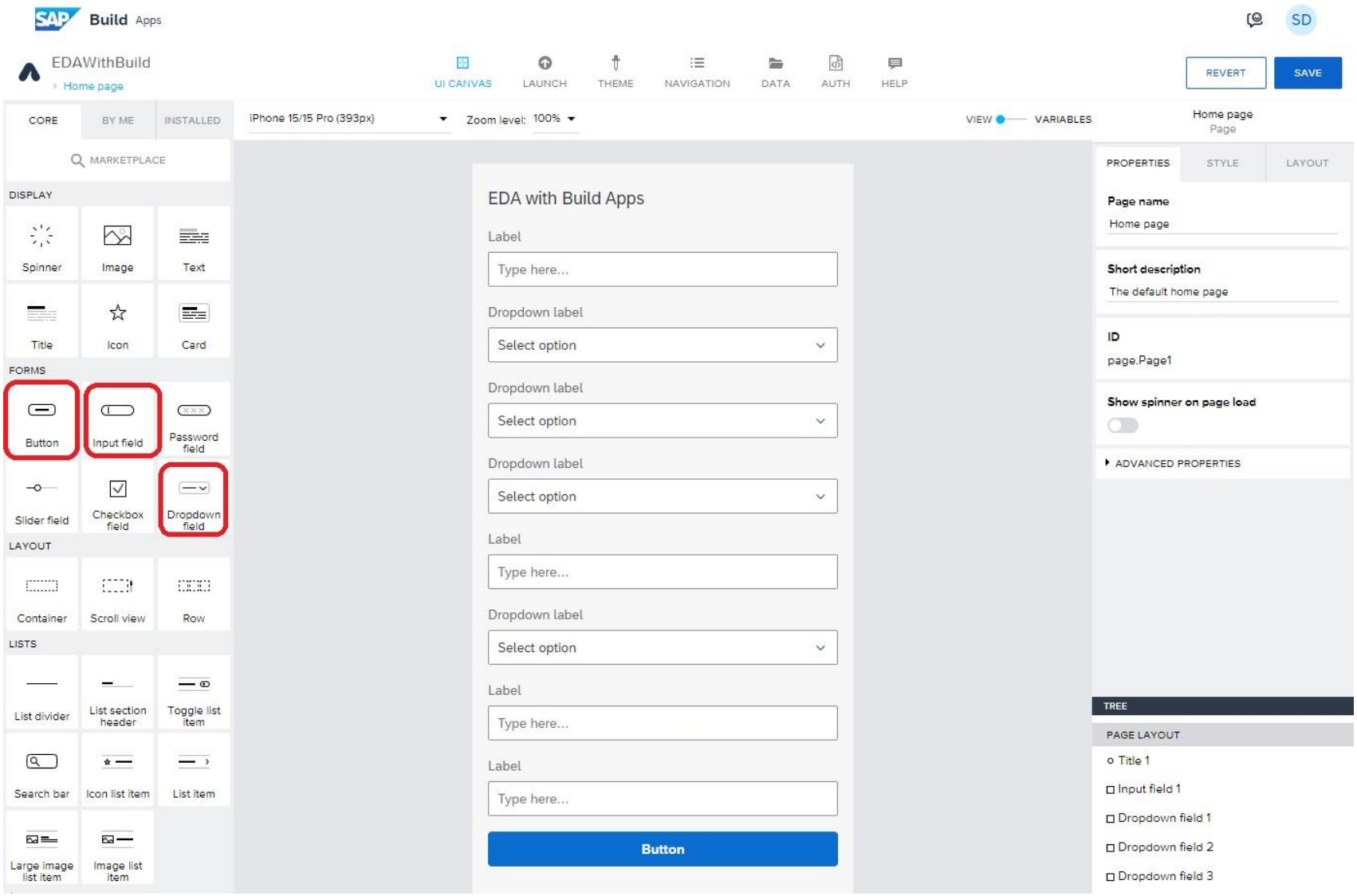
The first thing we will do is rename the headline and remove the 2nd line of text.

Simply click on the Headline which will highlight the properties on the right hand side of the screen. Change the name to a title of your choice or “EDA with Build Apps” by editing the properties in the pane on the right.

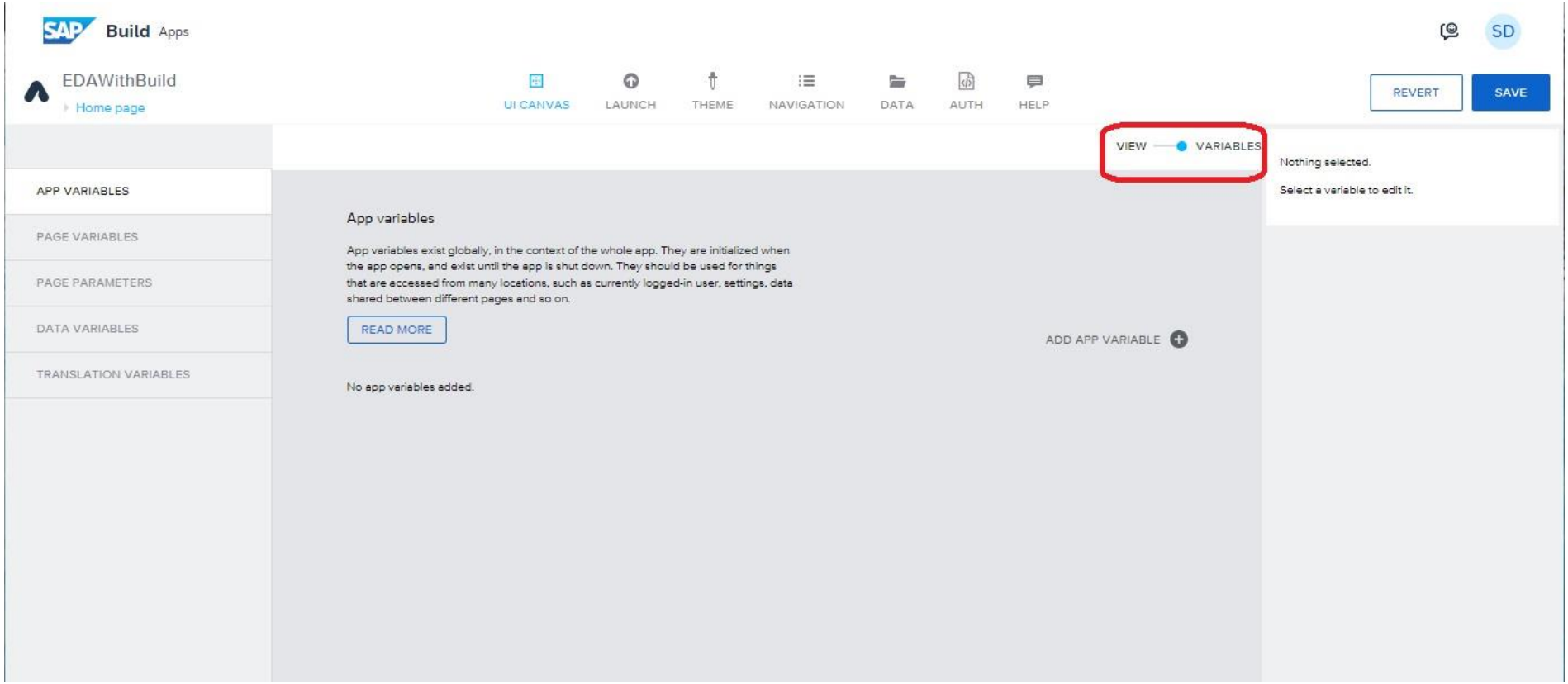
Next, click the 2nd link of text and remove it by clicking on the x in the blue field that shows up.



At this point, you will see a component palette on the left side with the various widgets you can add to your form. Now we will add 4 “Input Field”, 4 “DropDown Field” and 1 “Button” by dragging and dropping these field from the palette on the left to your form in the middle (one by one) so the next iteration of your form will look like in the next screen shot.
Arrange the fields in the order shown on the screen.

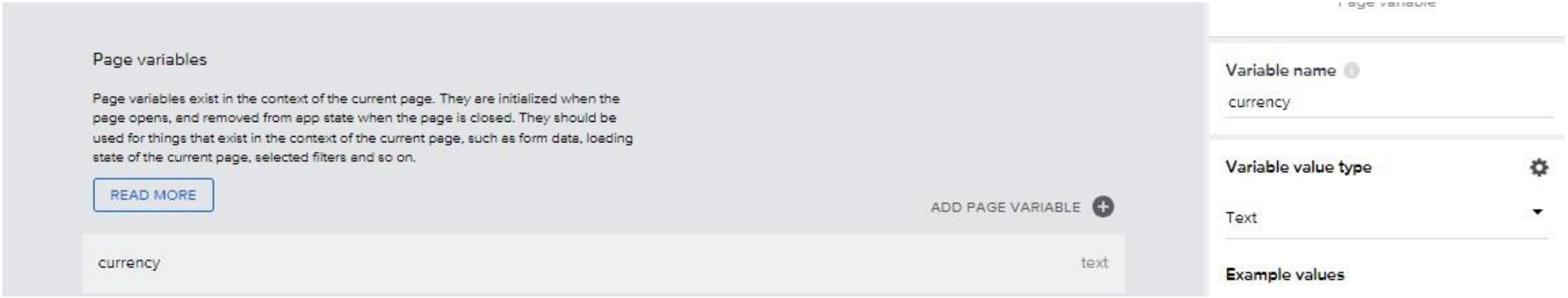


Now before we make any further changes to the form, we will switch our view to create variables to which we will bind the various input fields. On the top right of the screen, there is slider bar and it's currently sitting on "View" -> Click on it to slide it to the right to access the variables screen.




Now we will add all the variables required for the application. On the left side of the page, click "PAGE VARIABLES". Start by clicking the "ADD PAGE VARIABLE". From here you will be creating a series of variables (10 in total).

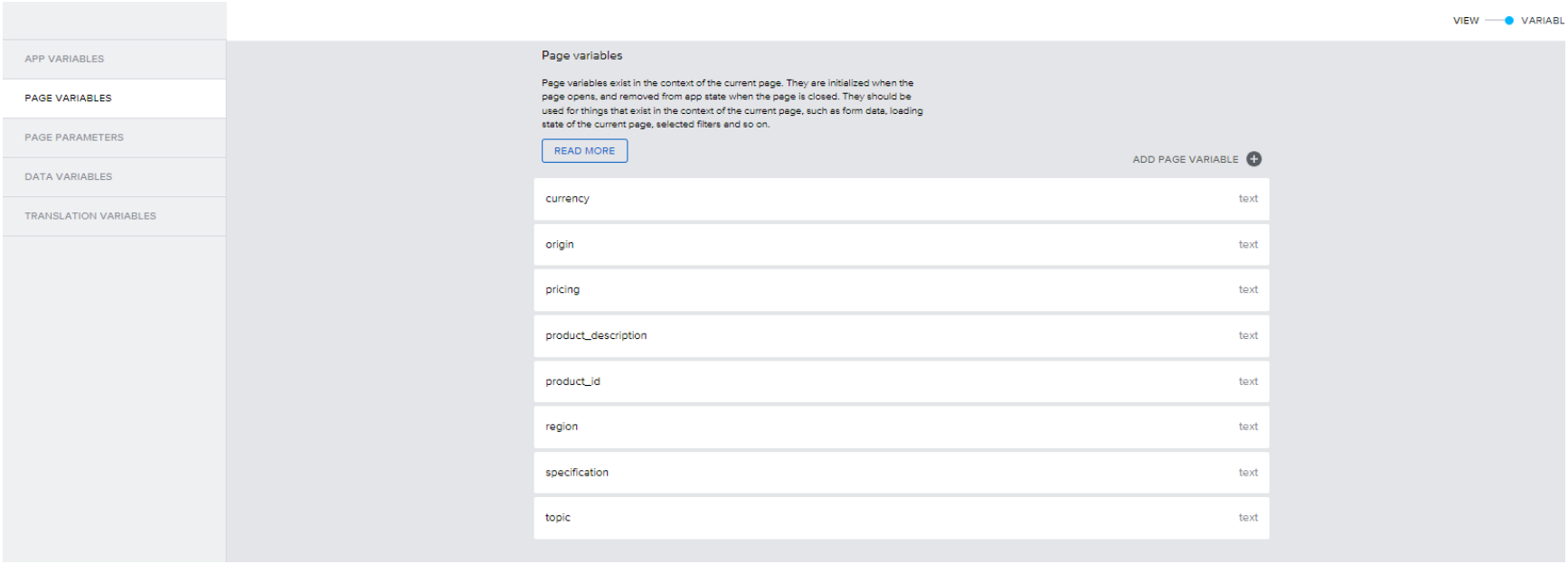
The first one is called “currency” (rename the variable in the pane on the right).




Now you will create the next 9 variables….all with the “Text” Data Type (you should have 10 in total at the end of this) and rename each of them to match the following names. *important* make sure the spelling is exactly the same for the below variables, lowercase and uppercase matter. Use the same spelling every time they are referenced as well.

- topic
- origin
- pricing
- product_description
- product_id
- region
- specification

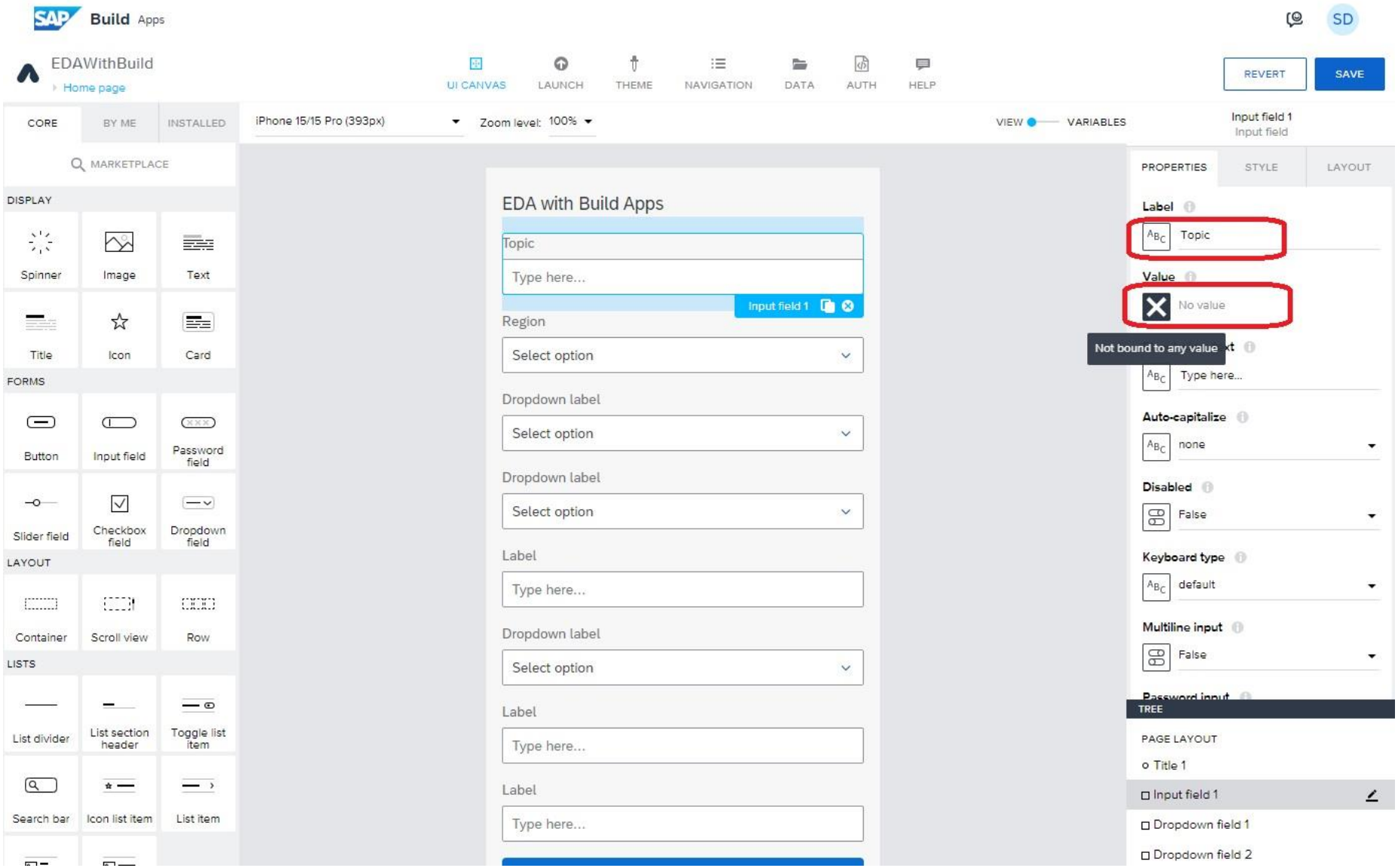
- Be sure to hit the “SAVE” button in the top right when you are done.



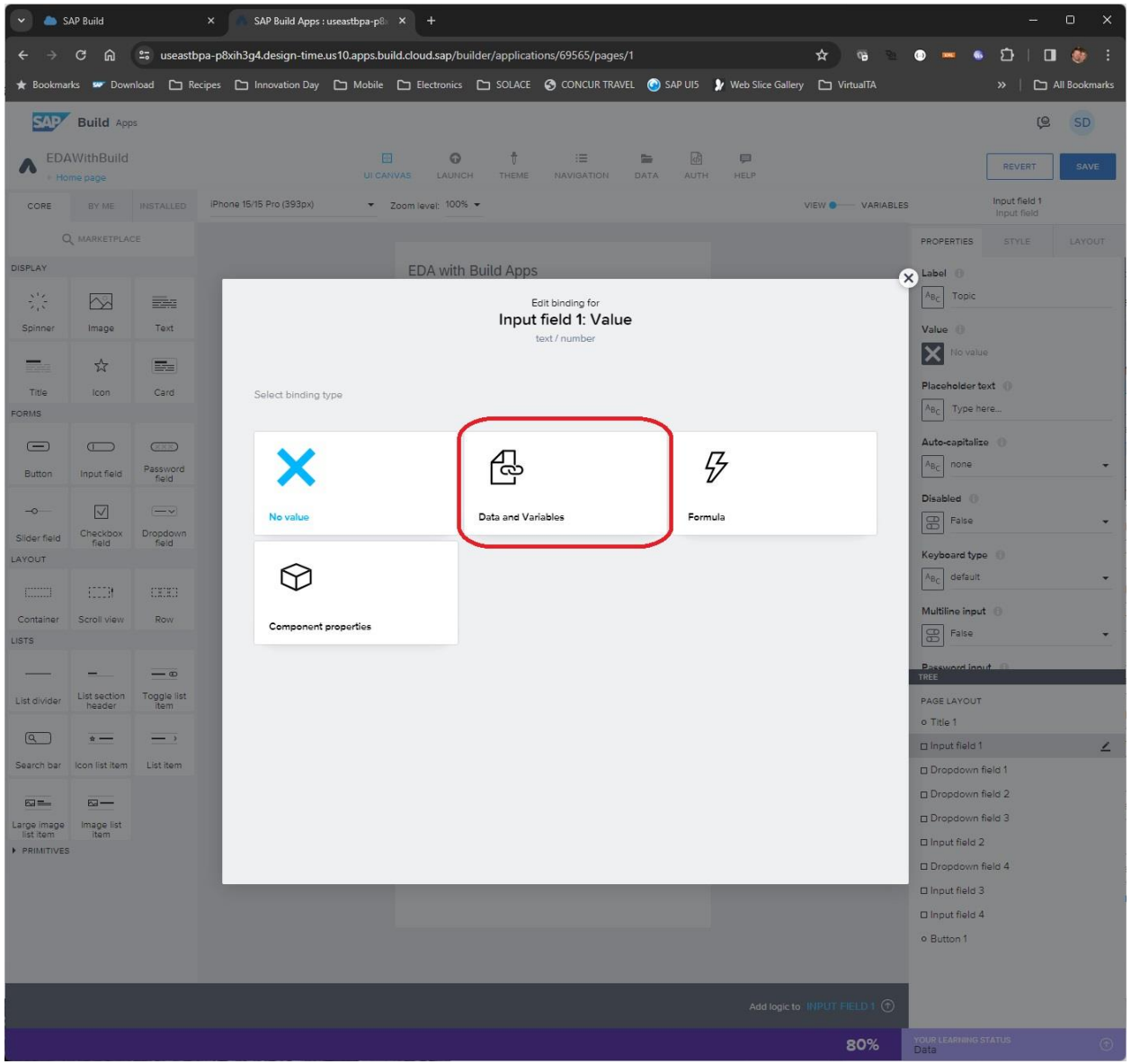
After saving, we will return to the form and configure our Form Fields and Map them to these App variables. From this screen, move the slider back to the “VIEW” position by clicking on it.

From here, you will now label our first field and map it to the corresponding page variable.

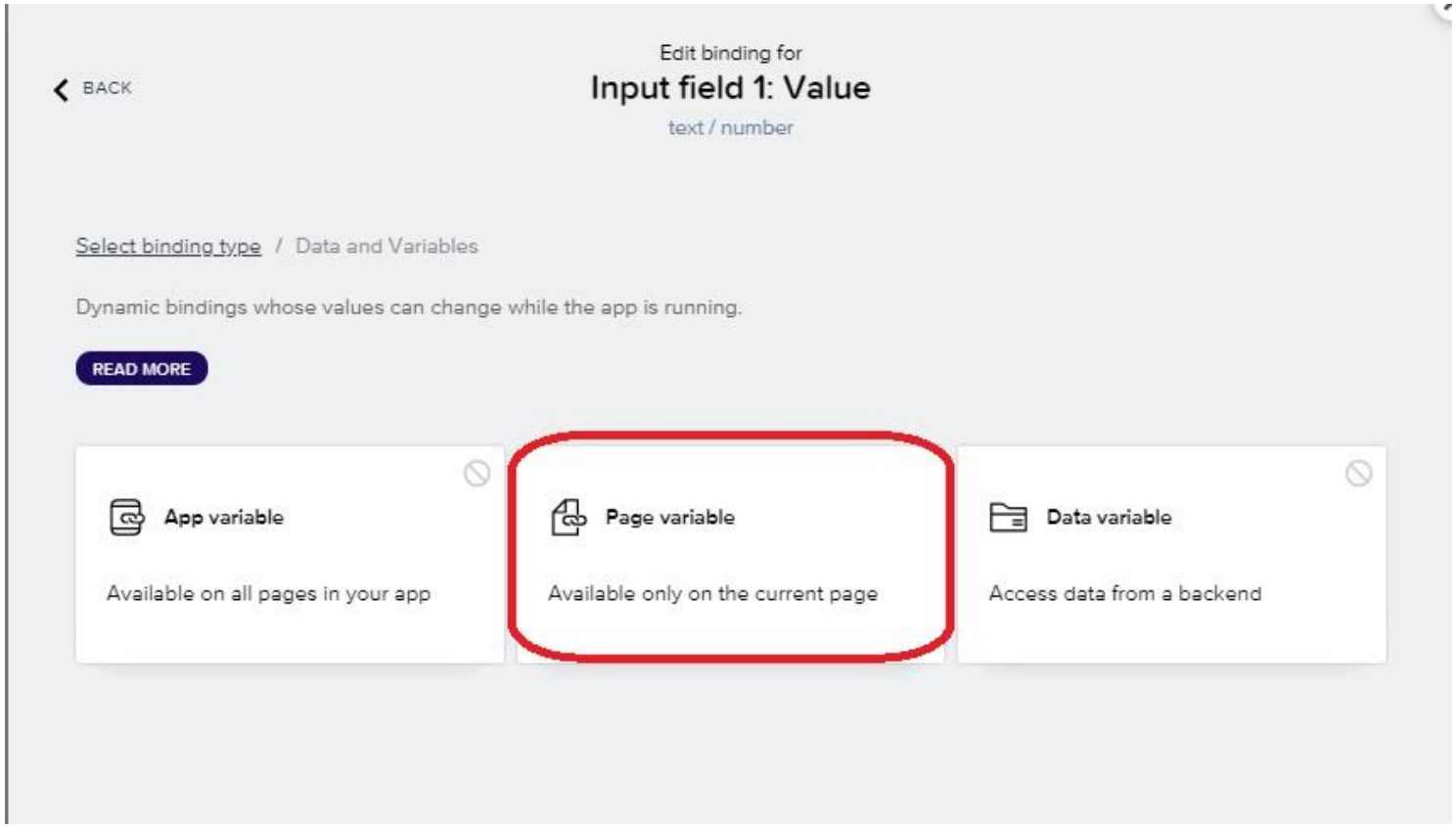
Click on the Label Property and Specify Topic. Next you will click on the Value and now we will bind it.



When you click on the field (Click on the X), you will choose what the field is being bound to, we will choose “Data and Variables”.

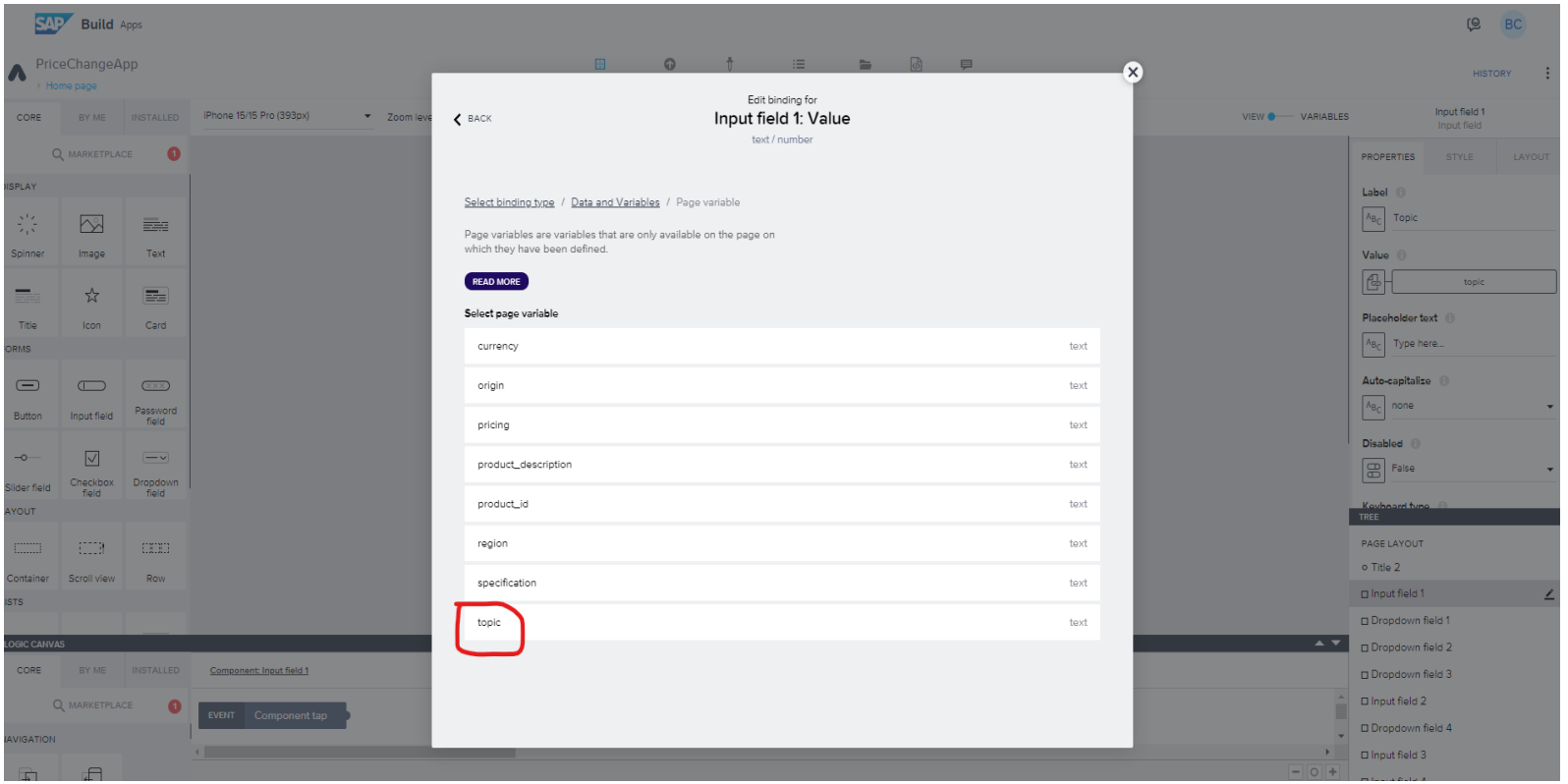


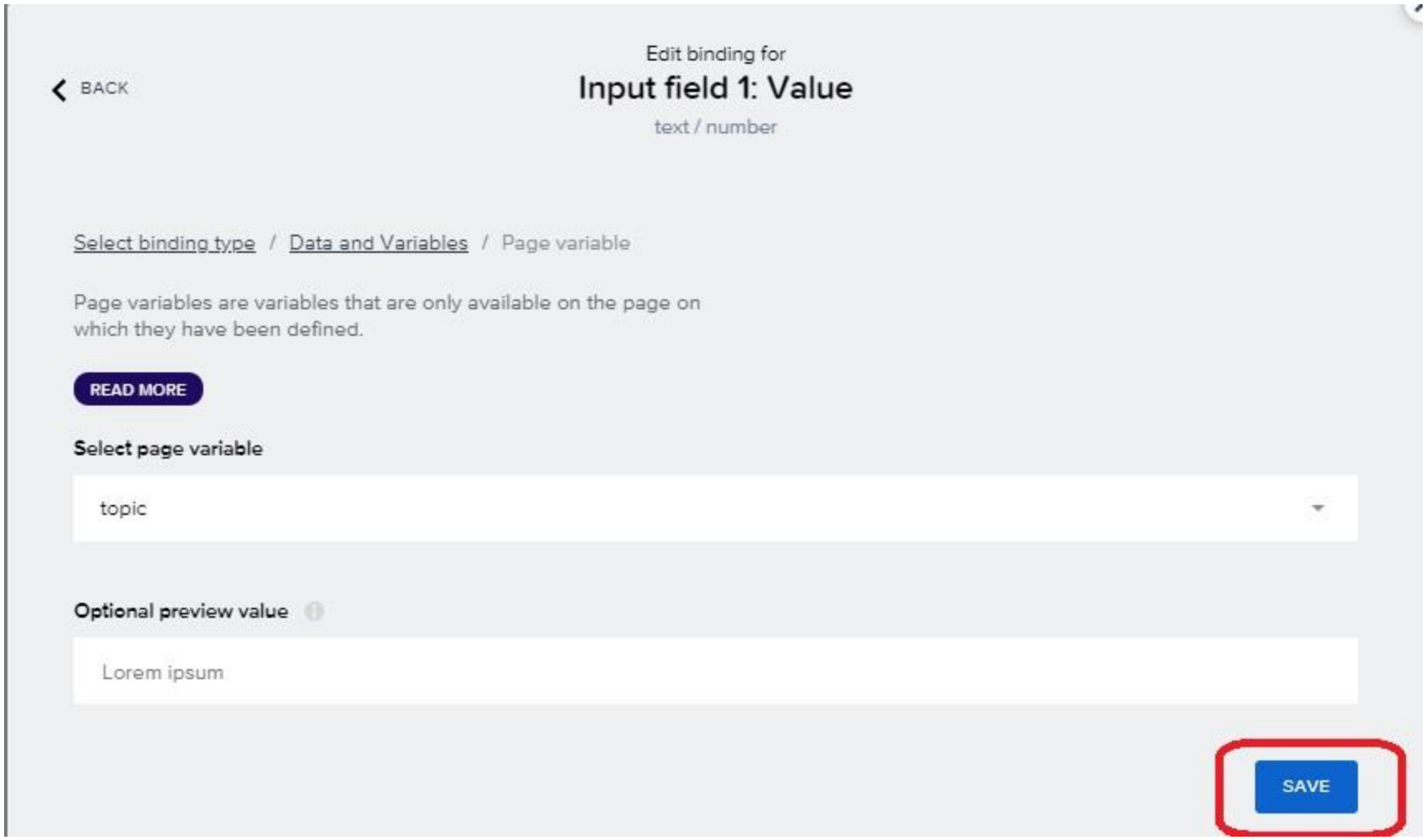
There are several types of variables but as per the earlier section, we created Page Variables so click “Page Variable”.



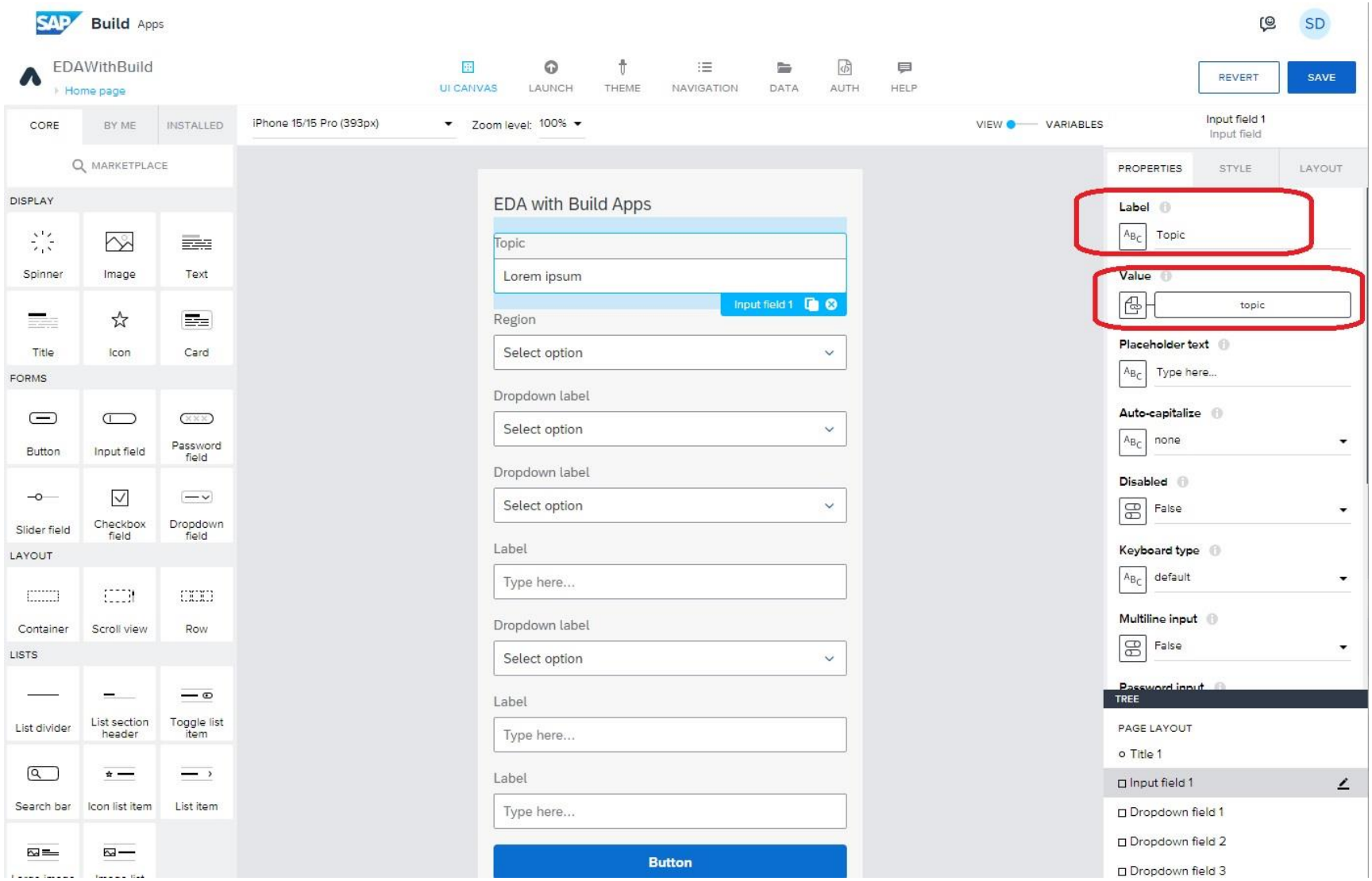
You will then be prompted with the list of all the variables you created earlier, Select “Topic” for this first binding.

Build Apps will then show you your selection, if you’ve picked the correct variable, Push the “Save” button at the bottom.




BACK
Edit binding for
Input field 1: Value
text / number
Select binding type / Data and Variables / Page variable
Page variables are variables that are only available on the page on which they have been defined.
READ MORE
Select page variable
topic
Optional preview value
Lorem ipsum
SAVE

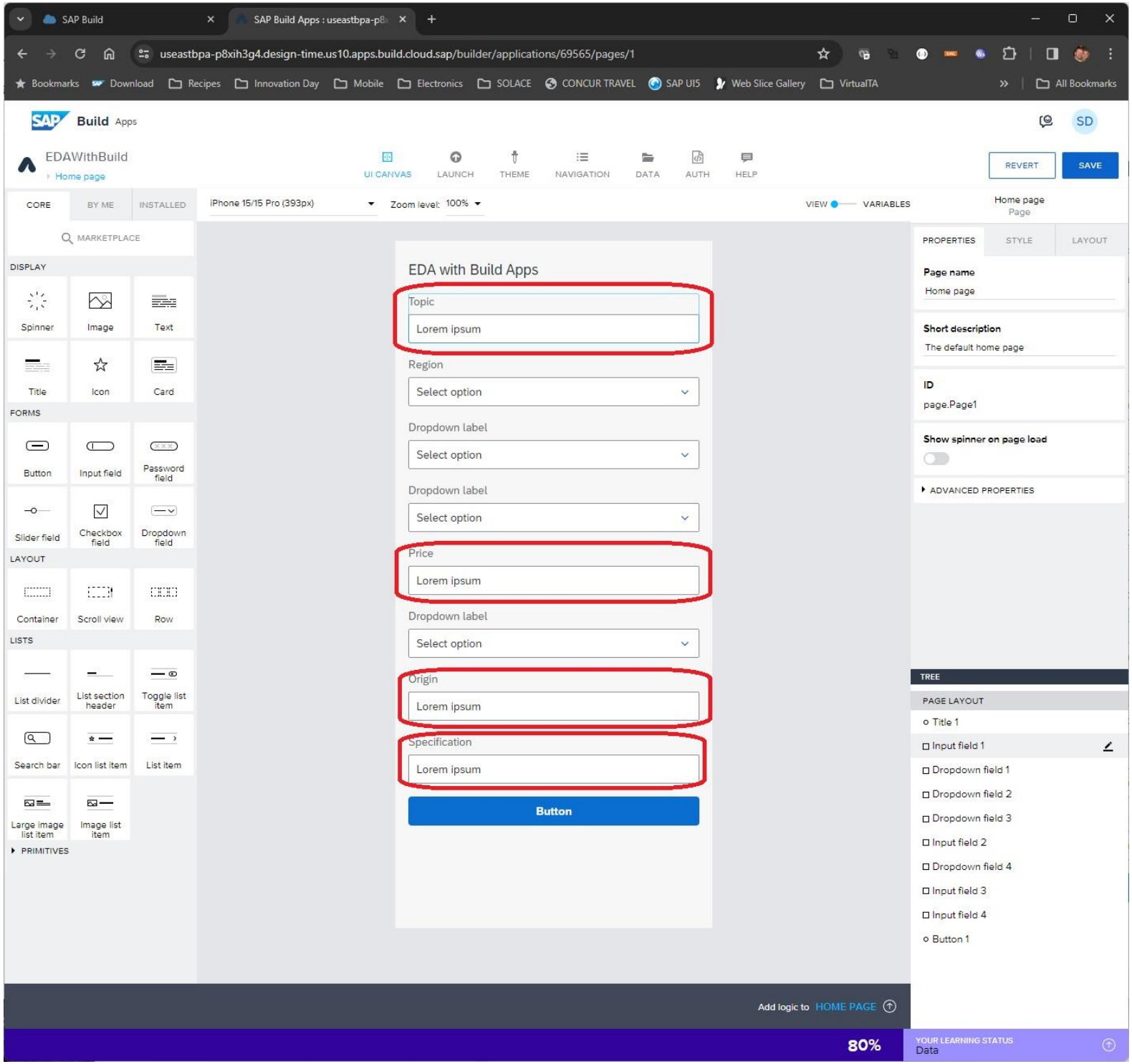
Now you should see that the Value of the Input Field is now bound to the “topic” page variable.



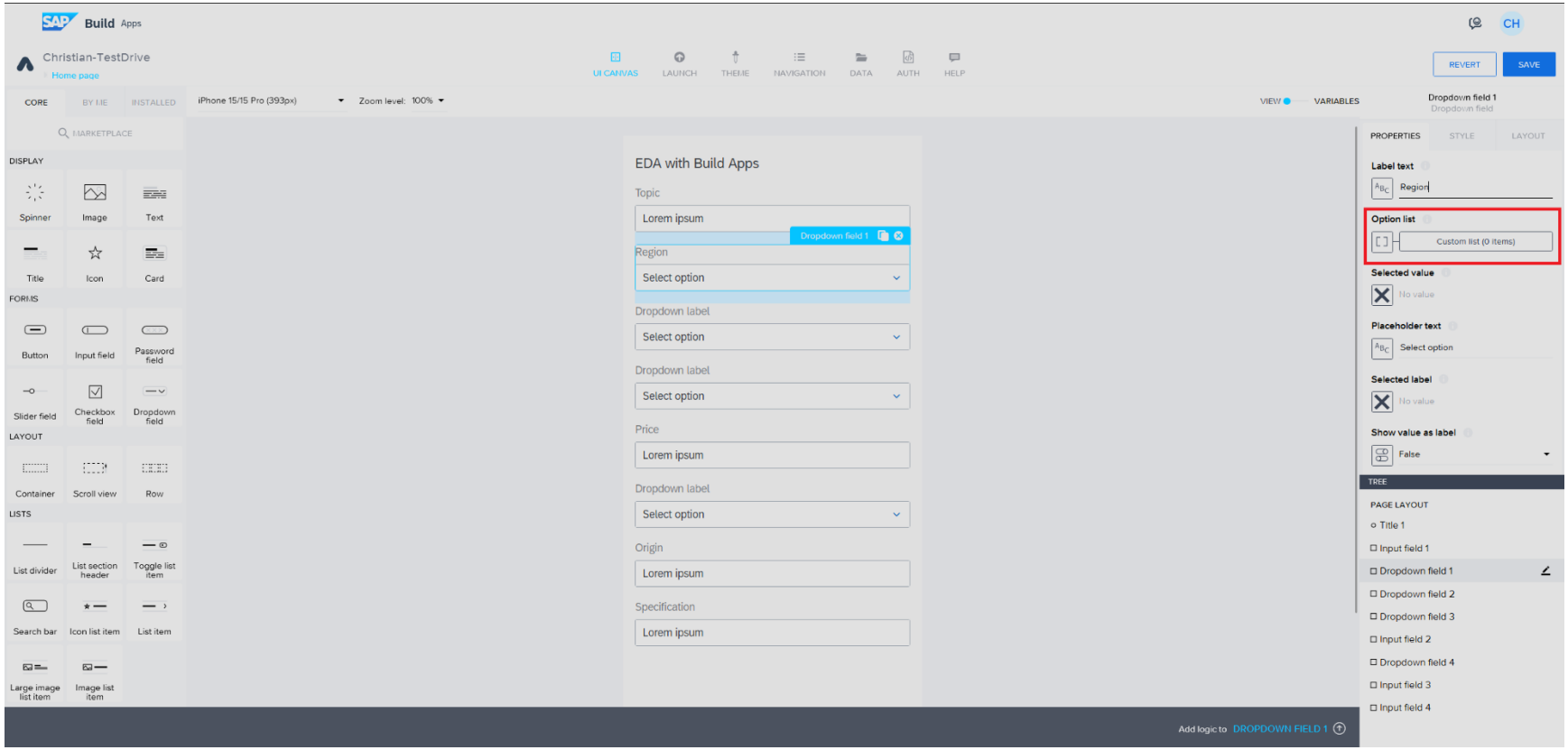
Now you will repeat the same process for the other 3 Input Fields (the ones that say “Label”):

- Price
- Origin
- Specification

So, once you are finished labelling and binding those input fields on the form, you should have something that looks like this:

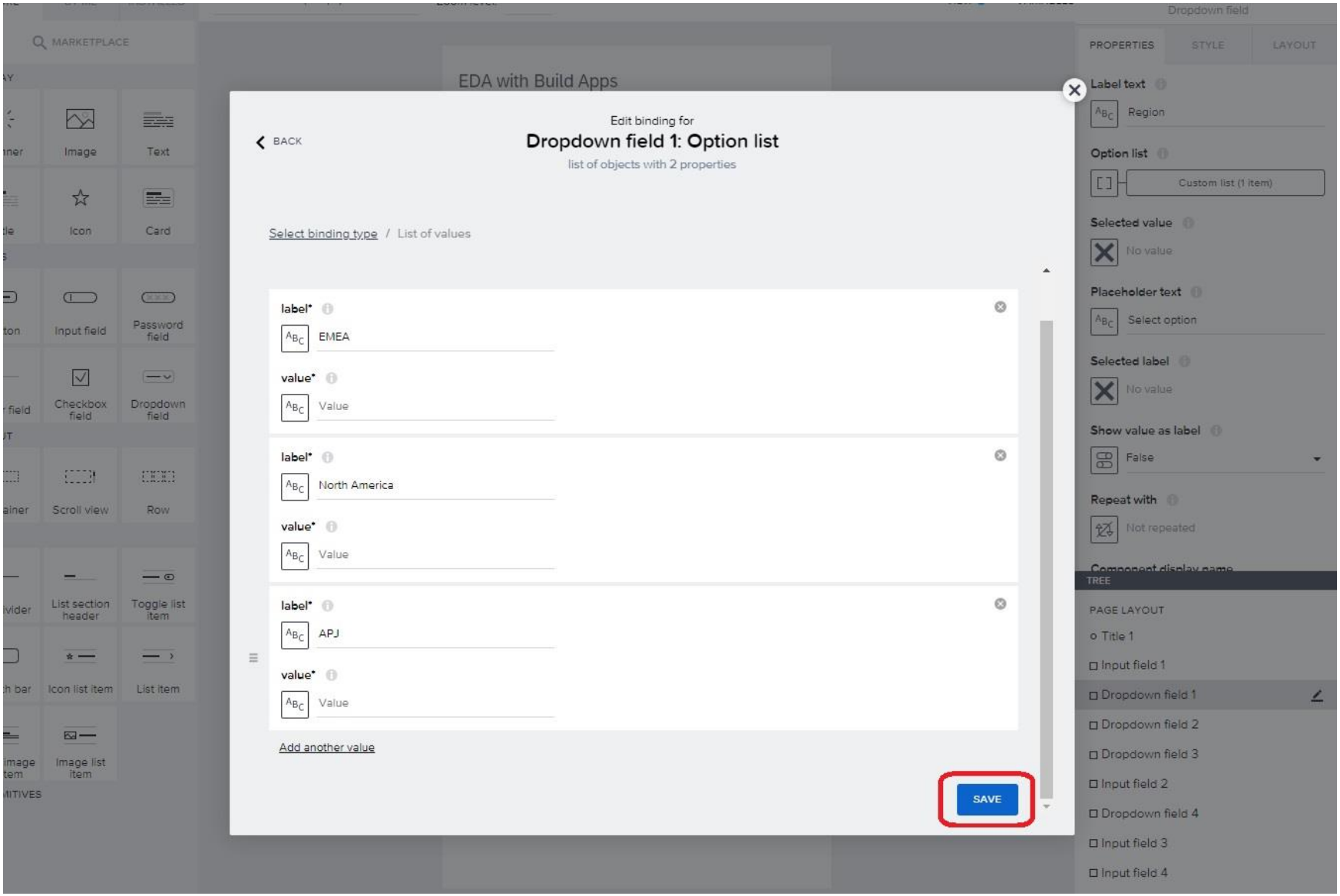


Now we will take care of the dropdown fields. Change the label to Region for the first field and then click to set the “Option List”. Click on the Box where it says “Custom List( 0 Items)…

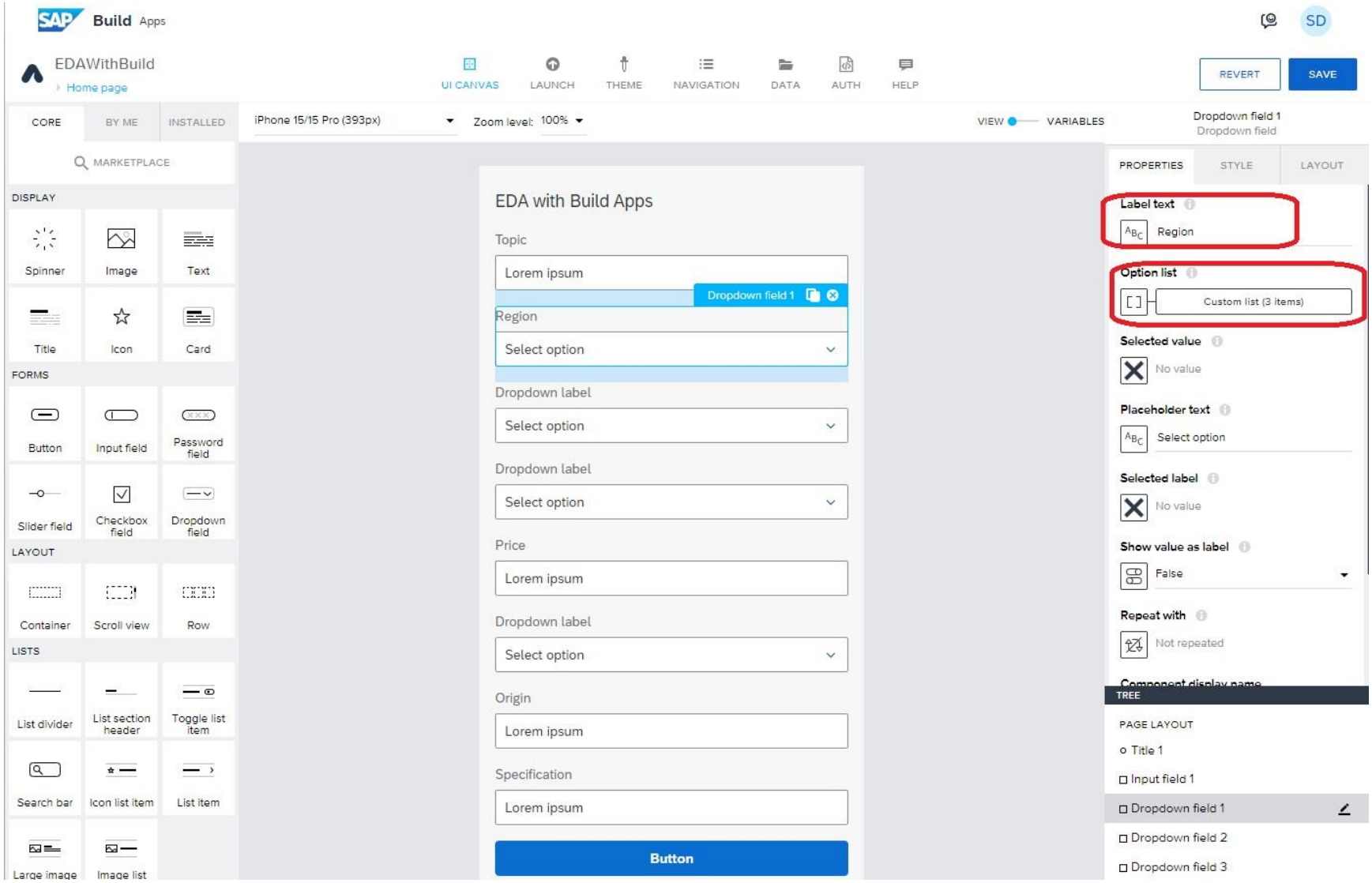


Once you click on the custom list, you will be presented with this screen where you can add your options. You will create 3:

- EMEA
- North America
- APJ

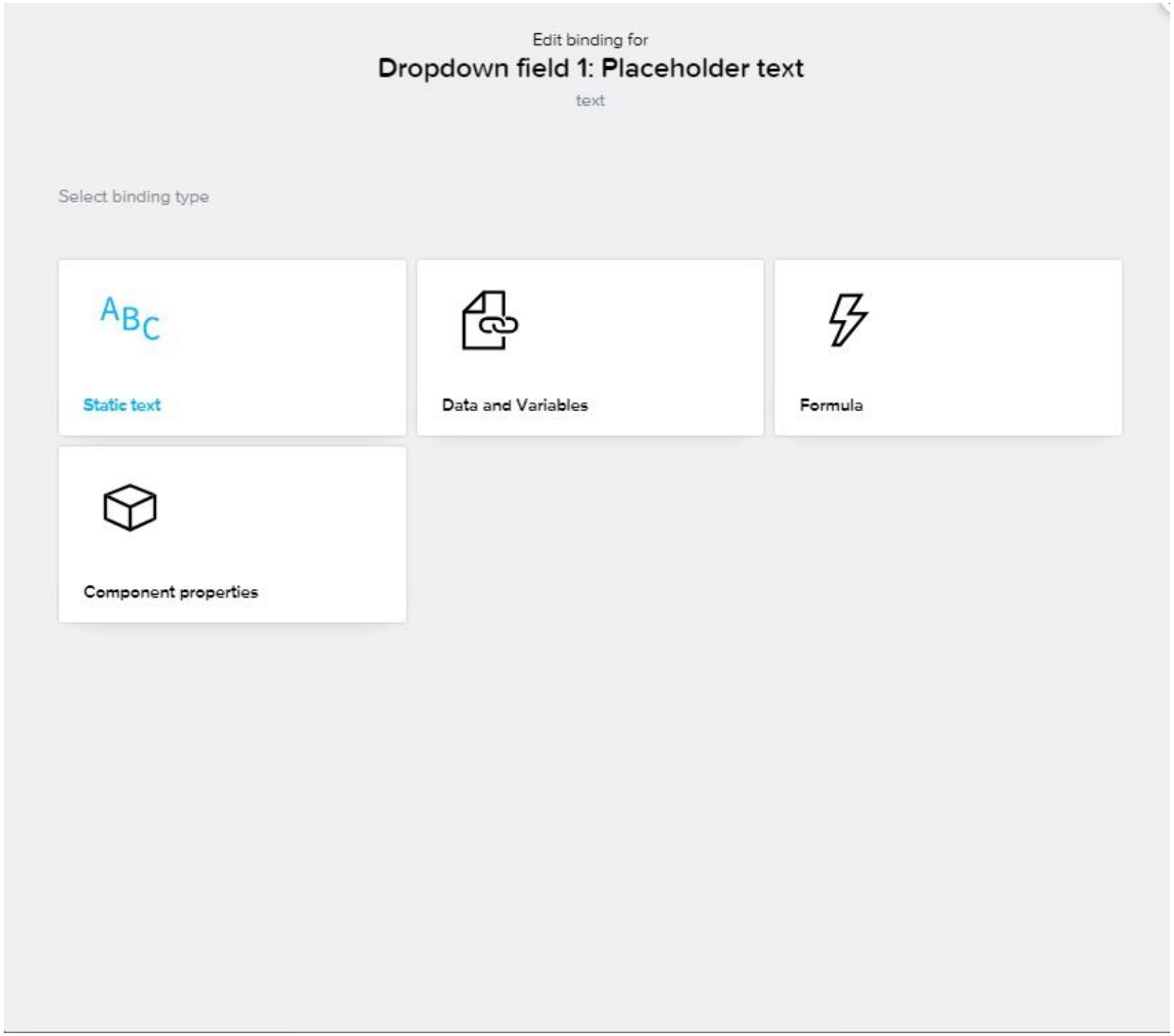


Once you click Save, you will be returned back to the previous screen and if you remembered to Save, you will now also see “Custom list (3 Items)” as in the this screenshot.

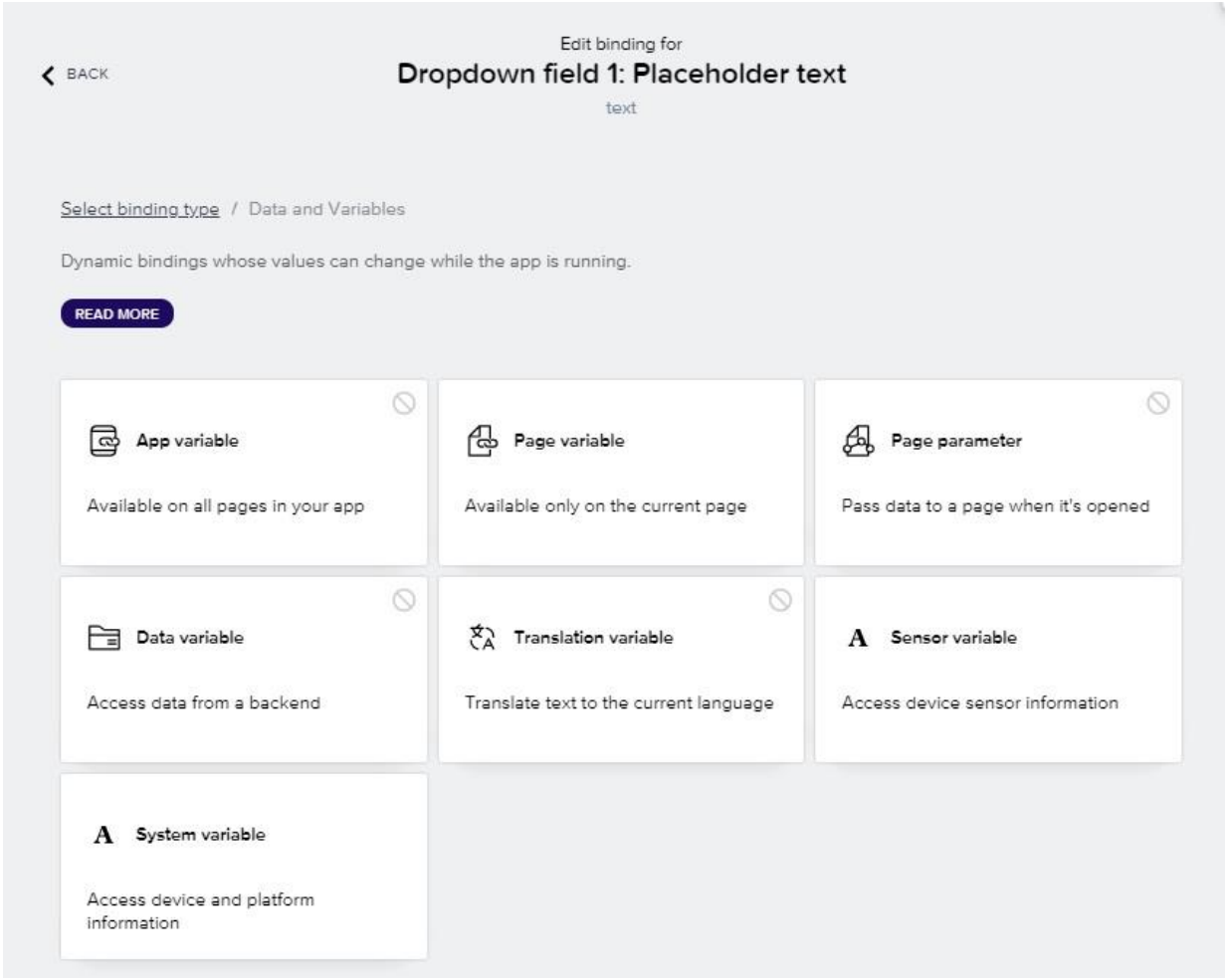


Now you will be following a similar process for setting the Placeholder Text and mapping it to the Region Page Variable.

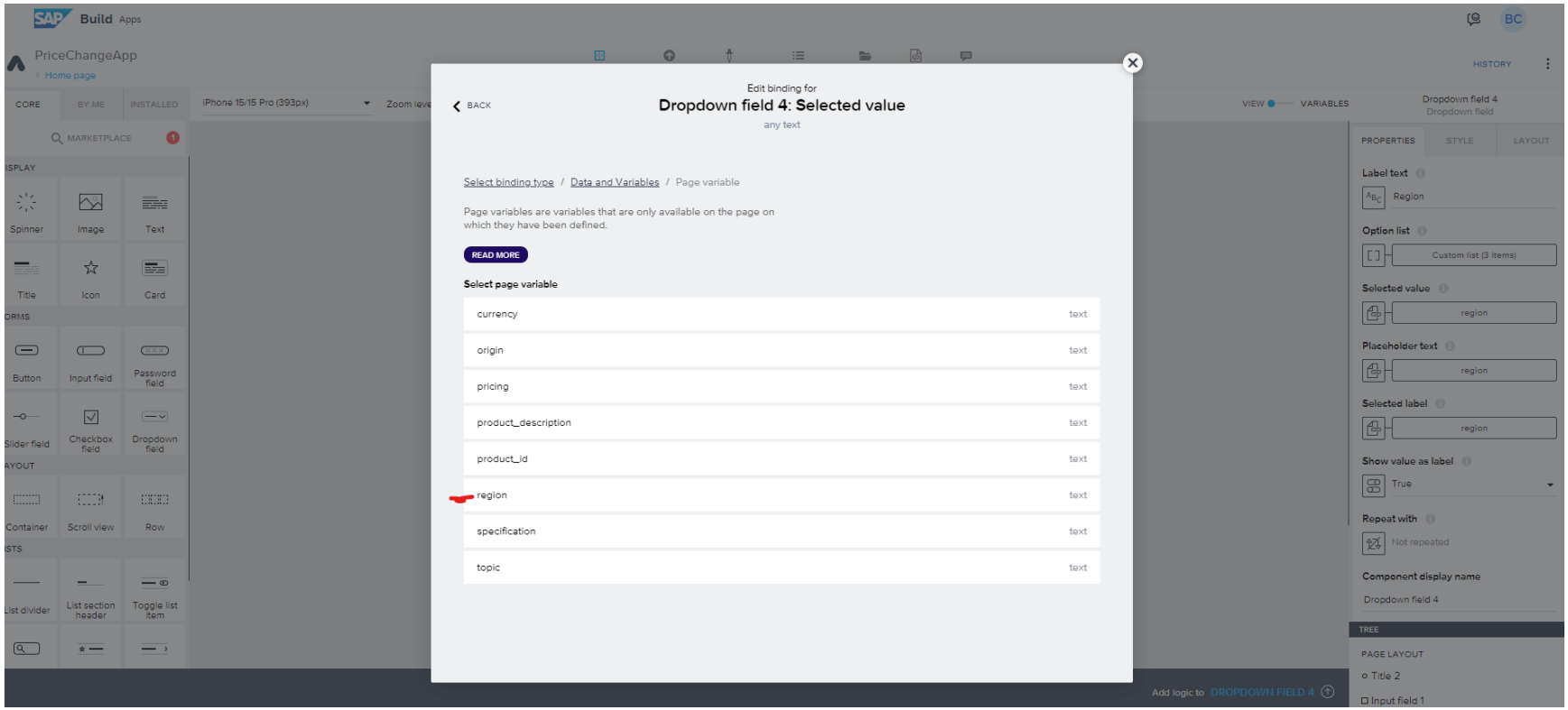
Just notice as you click through, there are a few more options but your path will be the same.
Start by clicking on the “ABC” box in front of “Select Option” under “Placeholder text” in the properties pane.
You will then see the following 4 options, Choose “Data and Variables”.



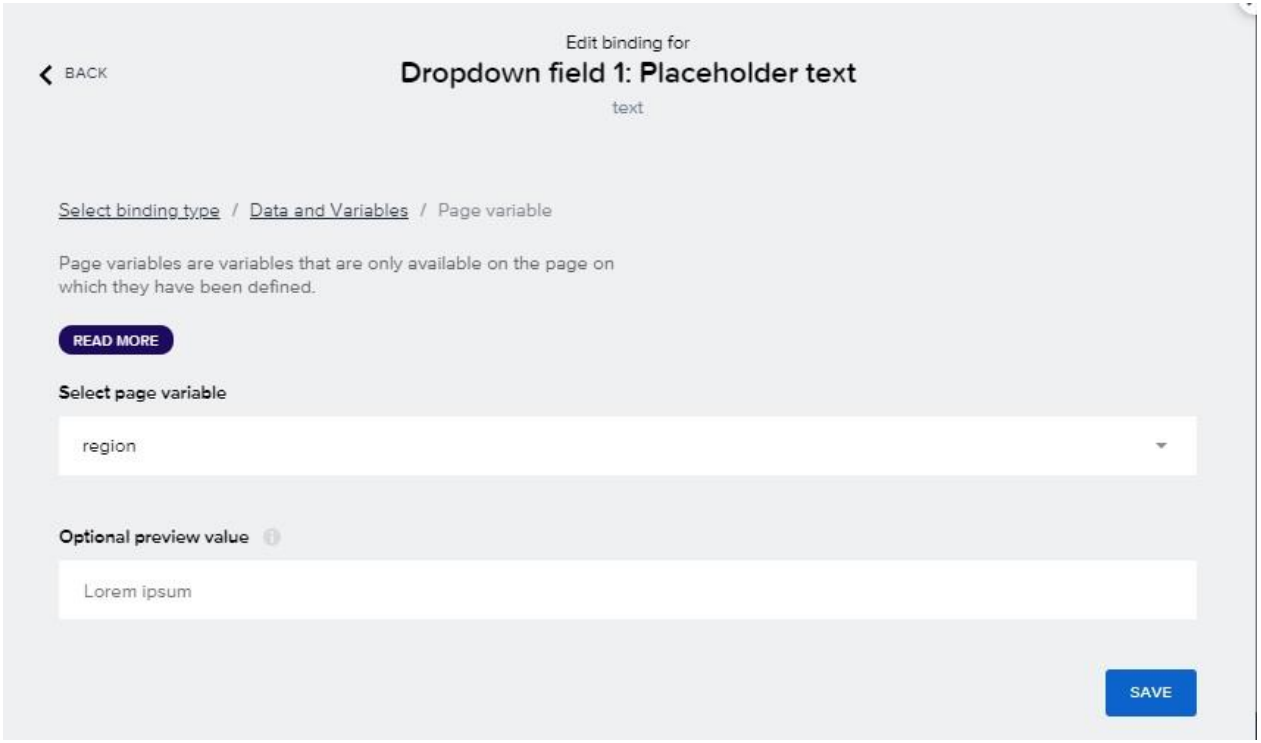
Select “Page variable”



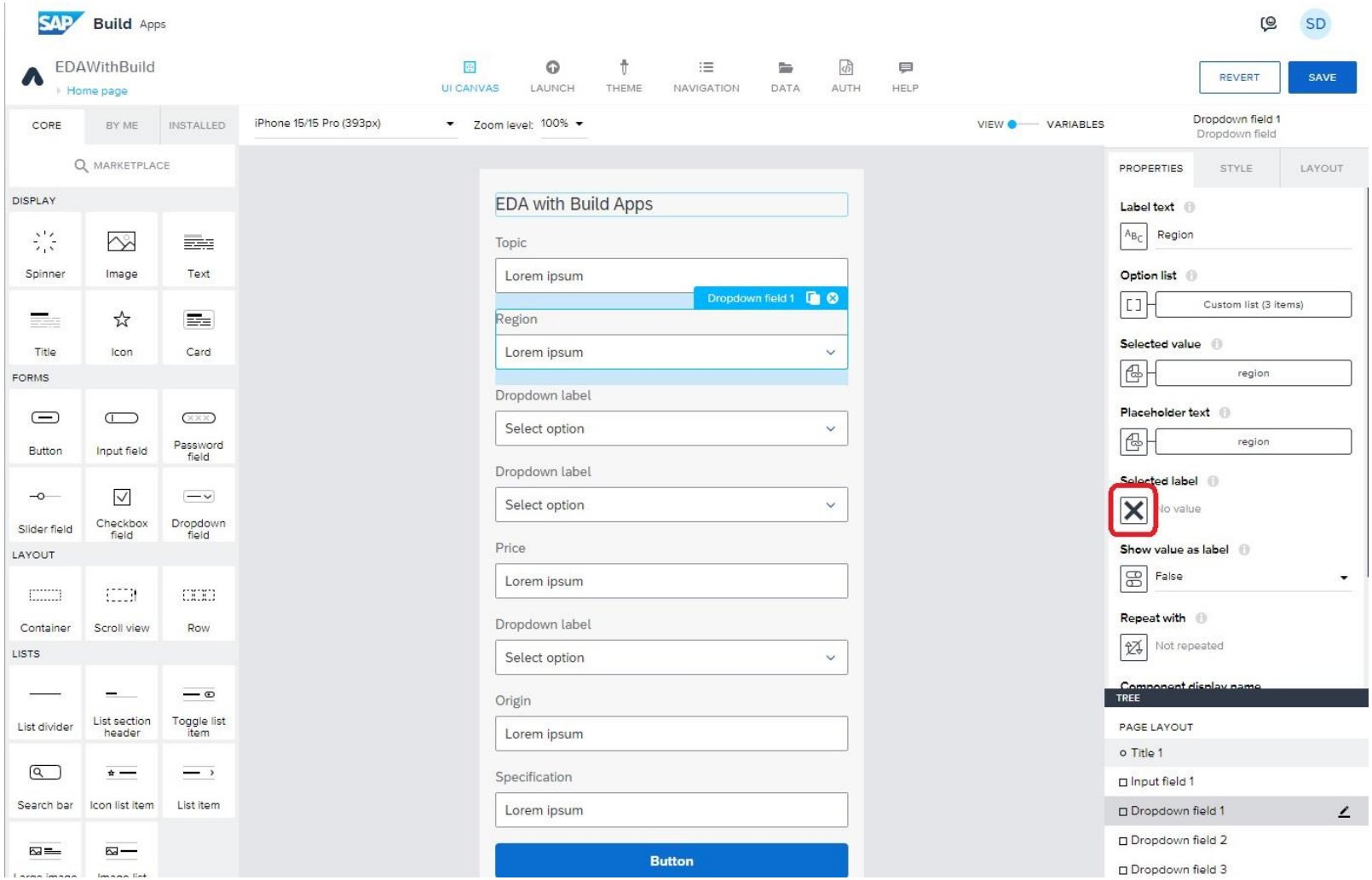
Again you will see your page variables, select “region”.



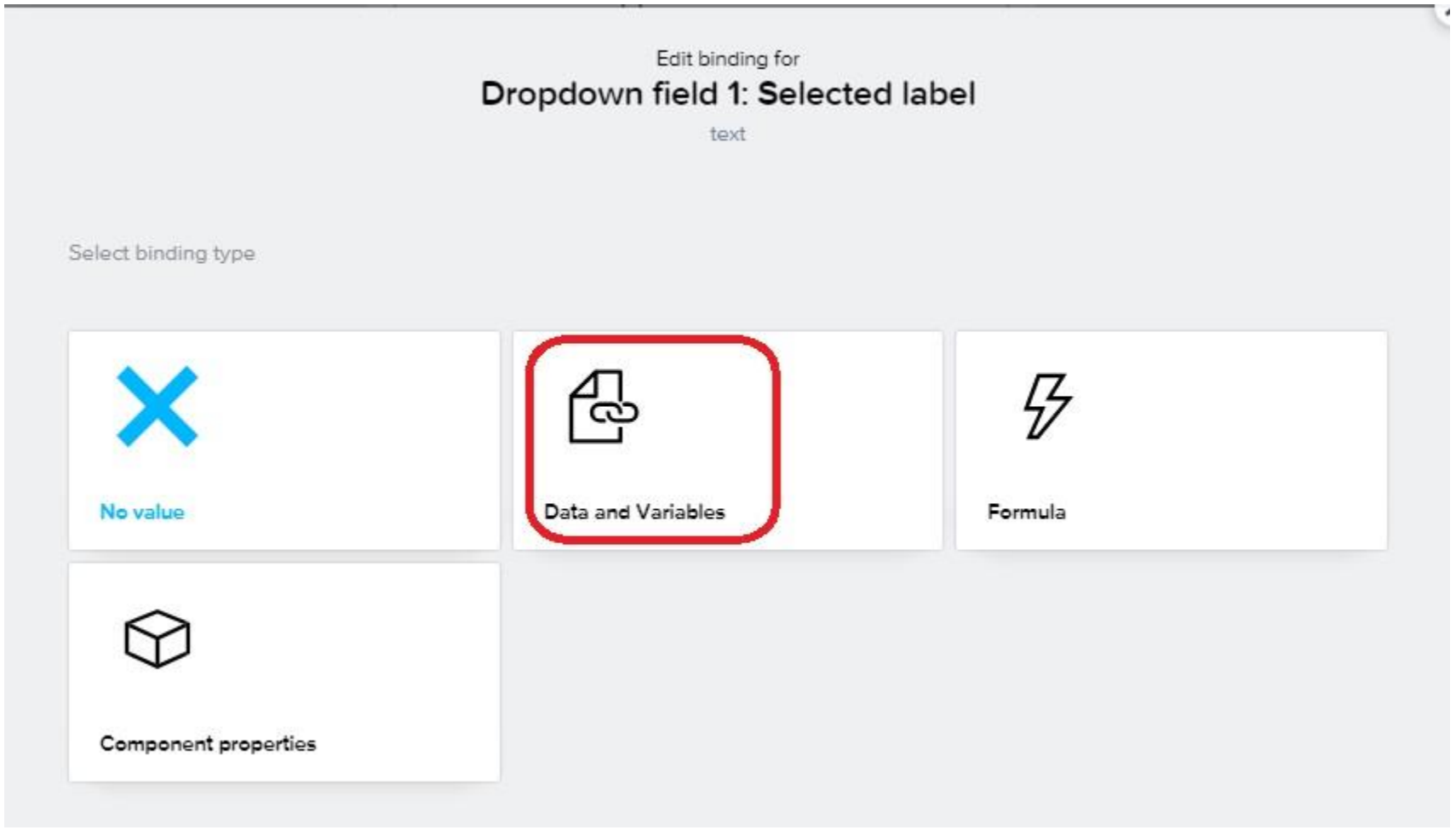
Build Apps will show you the selected option, Click “SAVE”.



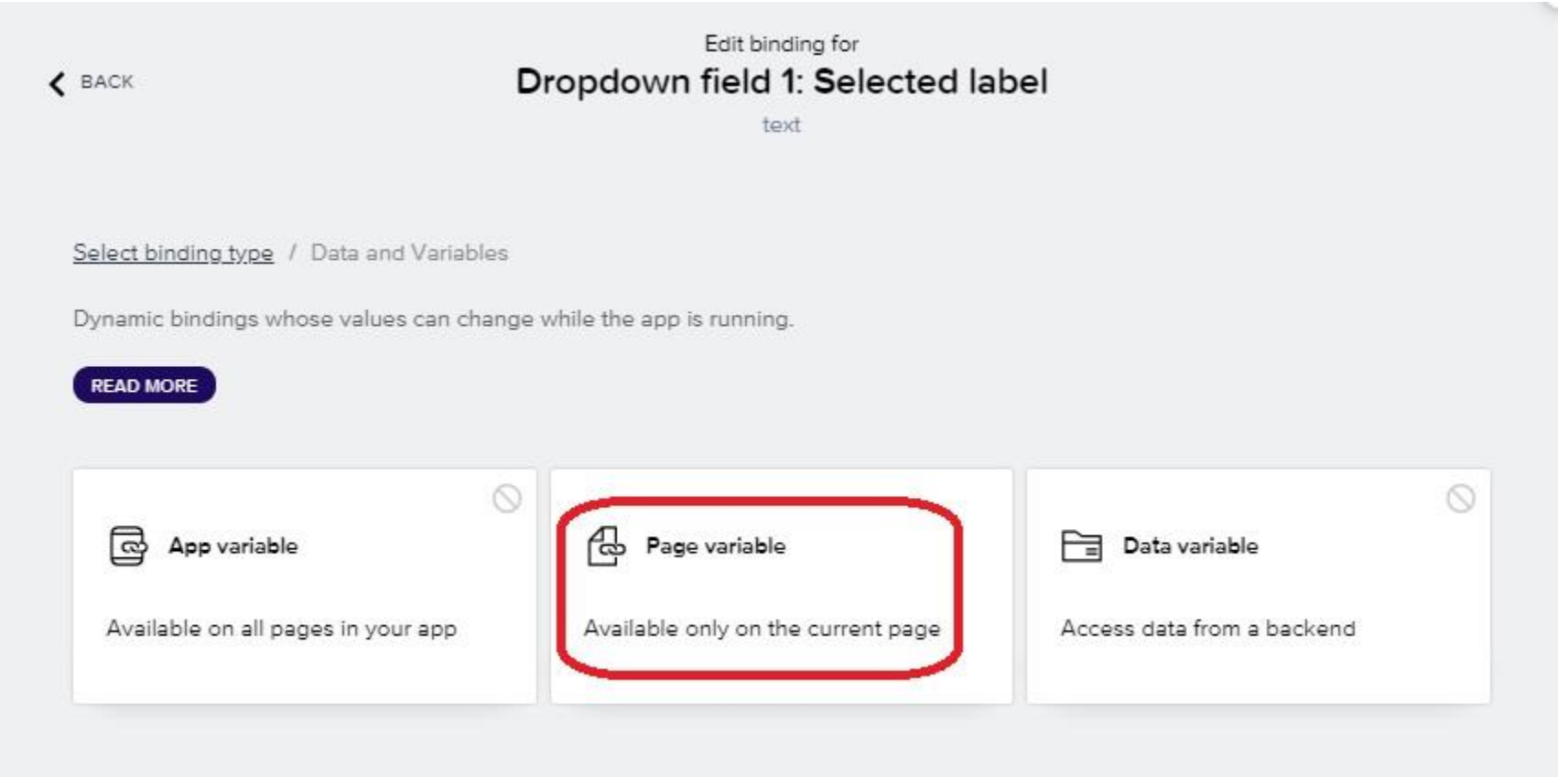
Now we will assigned the “Selected Label”, using the same process as above to find our Region Page Variable. Click on the X for Selected Label.



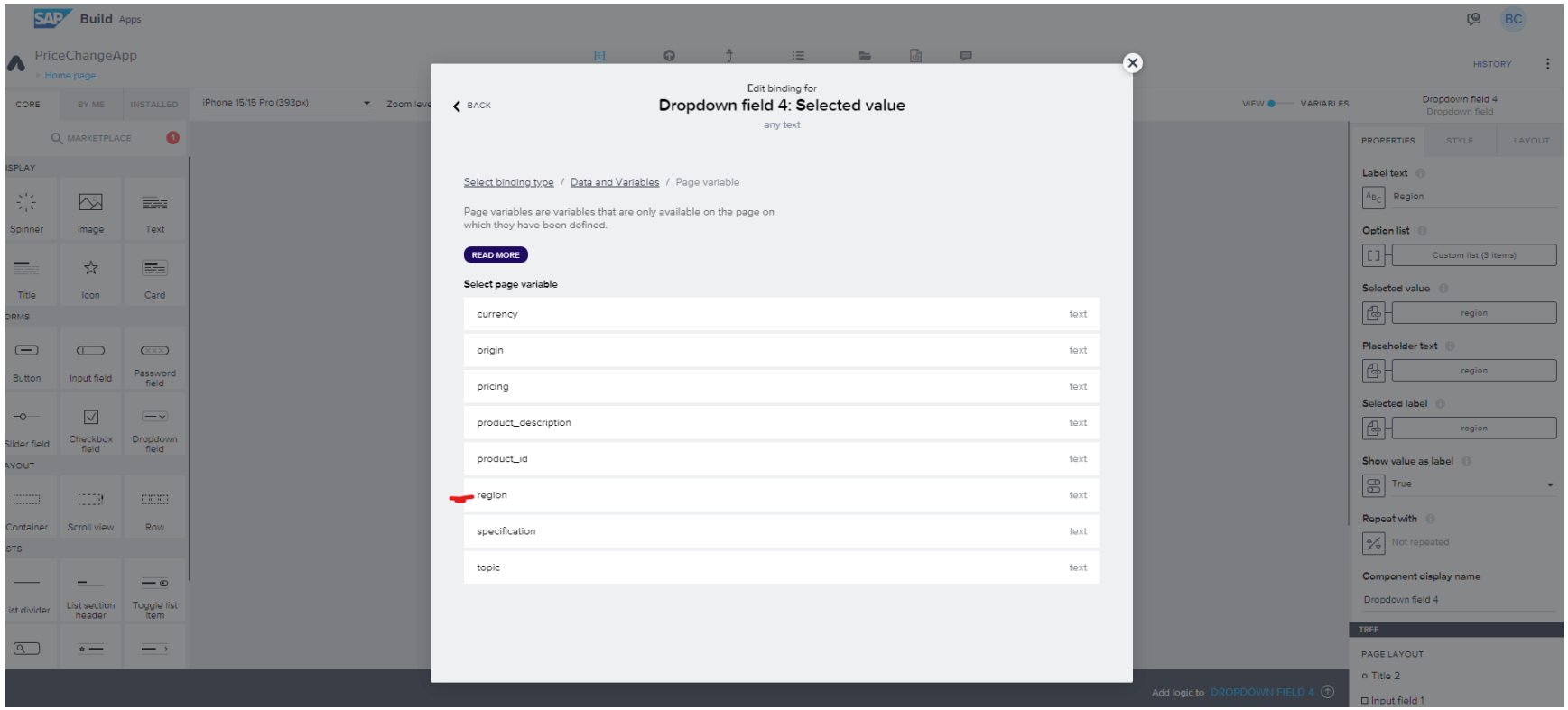
Click “Data and Variables”



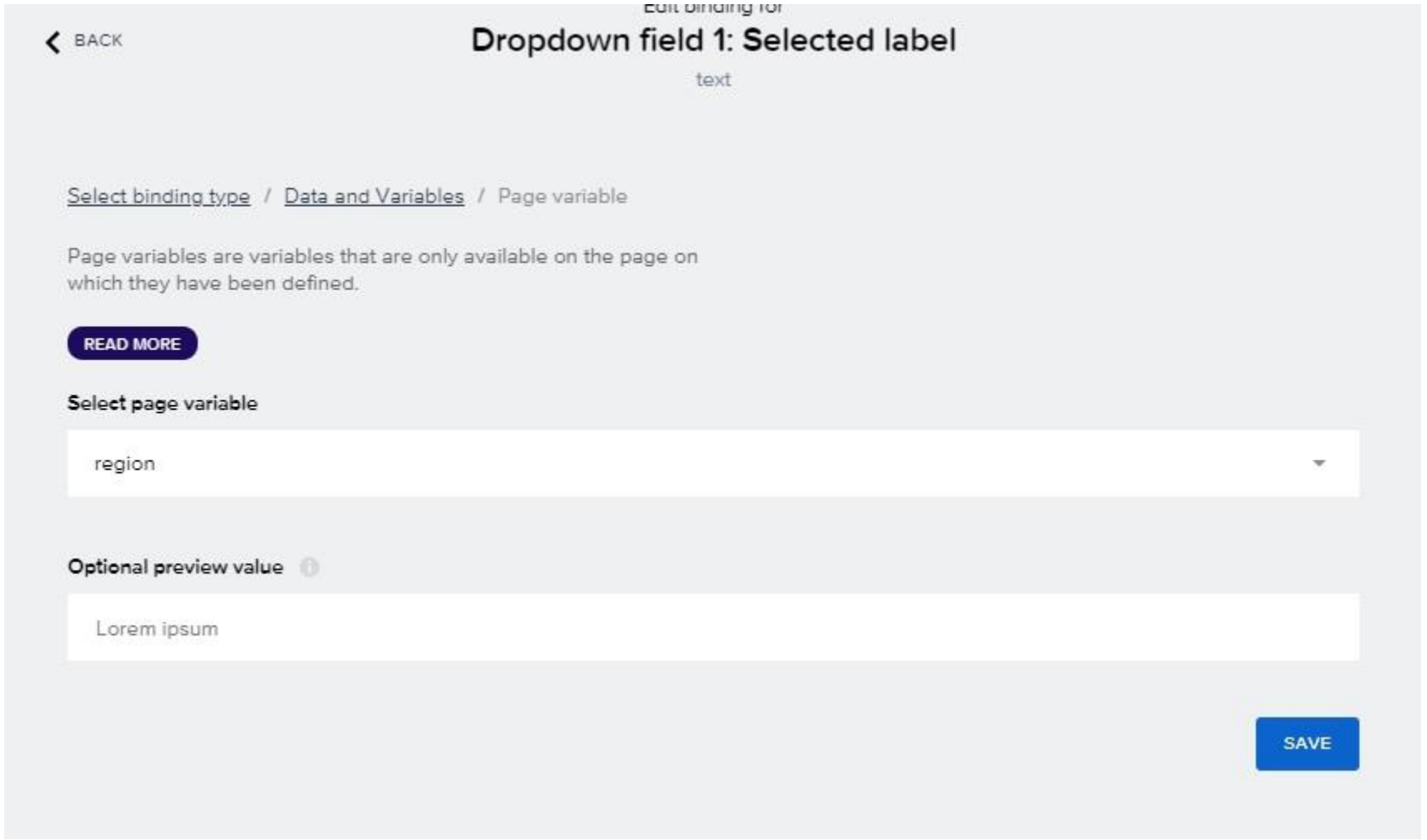
Click on “Page variable”



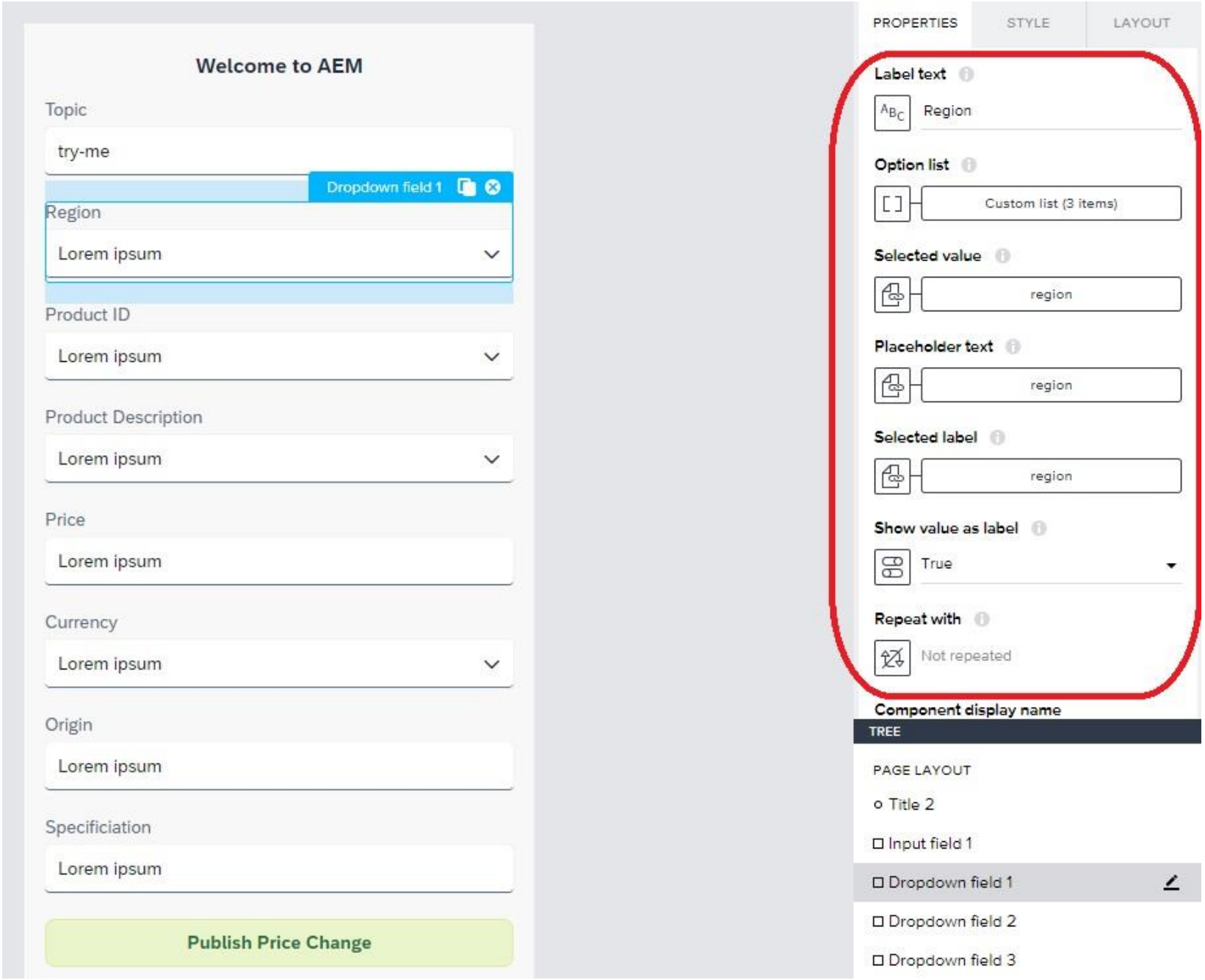
Click on Region



Once “Region” is selected, click on “SAVE”



Make sure you set “Show value as label” to “True”.
Your properties should now look similar to what is displayed below in RED, double check that you have formatted them properly.



Now you will follow the same process to populate the next dropdown:

Label – Product ID

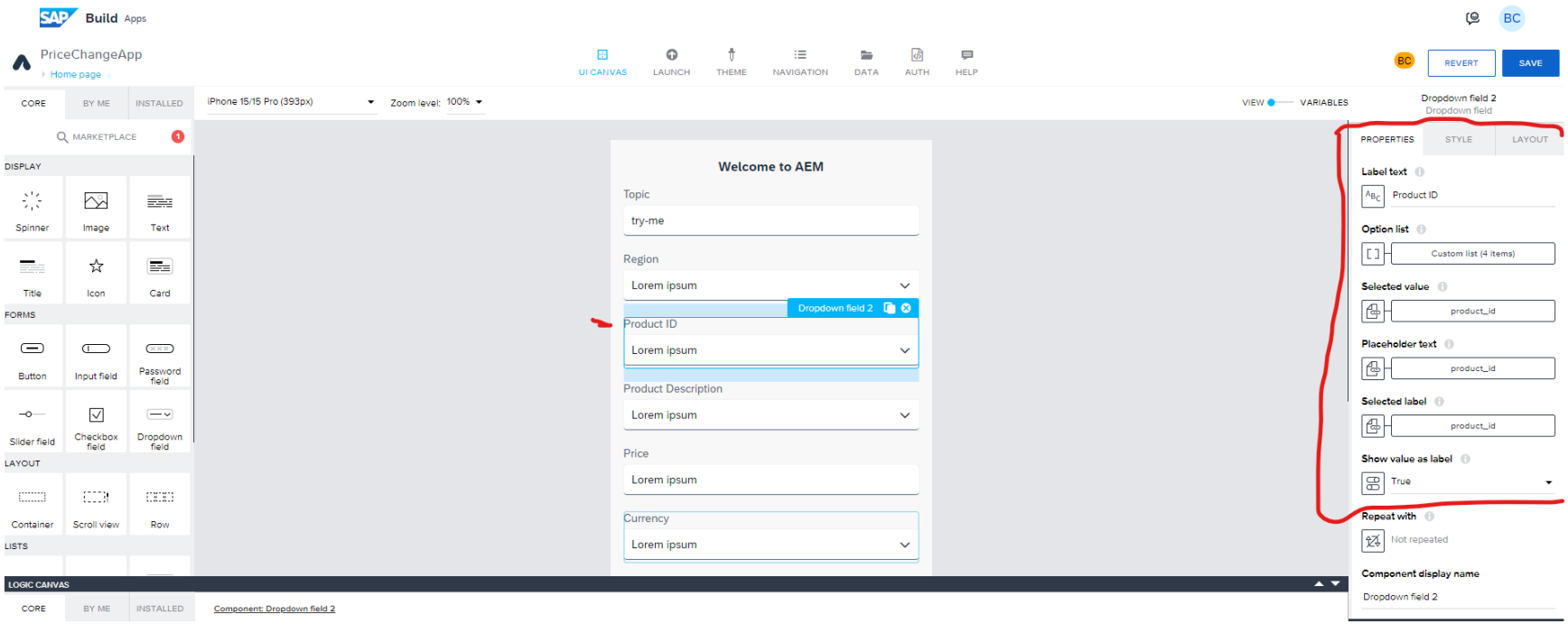
Option List

- 51001
- 52005
- 62009
- 95004

And bind all the various fields to your product_id page variable as before for region.

And also set “Show value as label” to “True”.

Double check that your configuration looks like this:



Now you will follow the same process to populate the next dropdown:

Label – Product Description

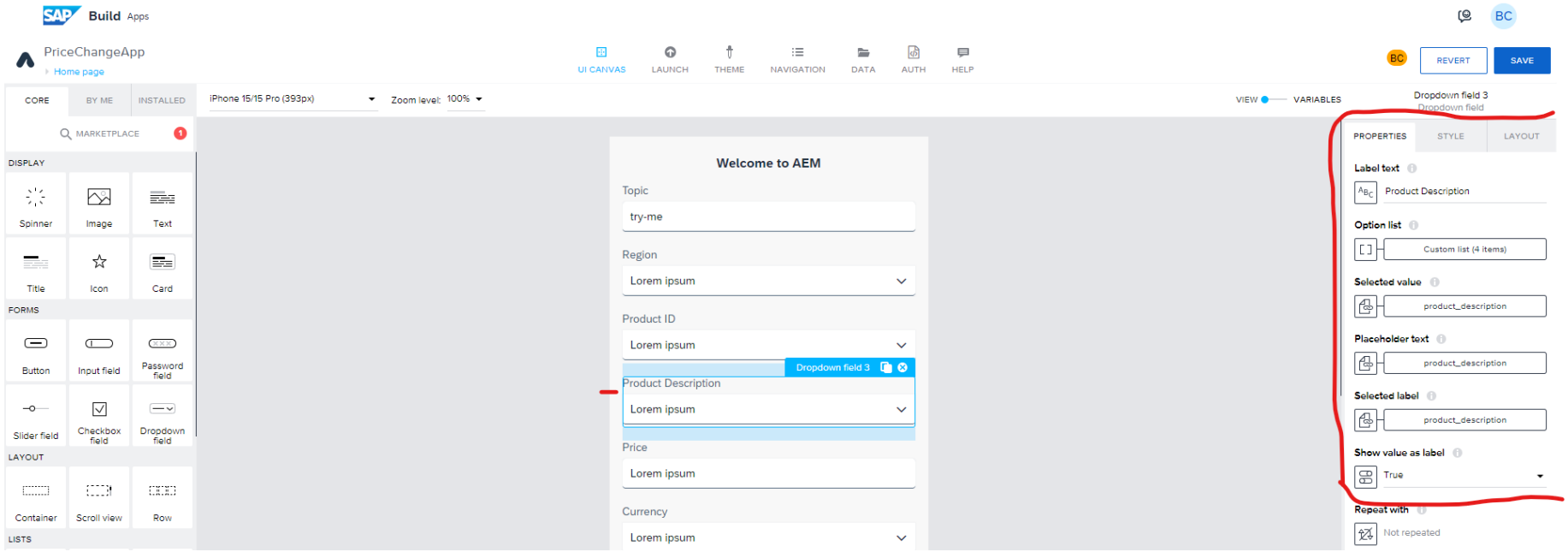
Option List

- Milk Chocolate
- Apple
- Bread
- Cheese

Again, bind all the various fields to your product_description page variable as before for region & productid.

And also set “Show value as label” to “True”.

Double check that your configuration looks like this:



Now you will follow the same process to populate the next dropdown:

Label – Currency

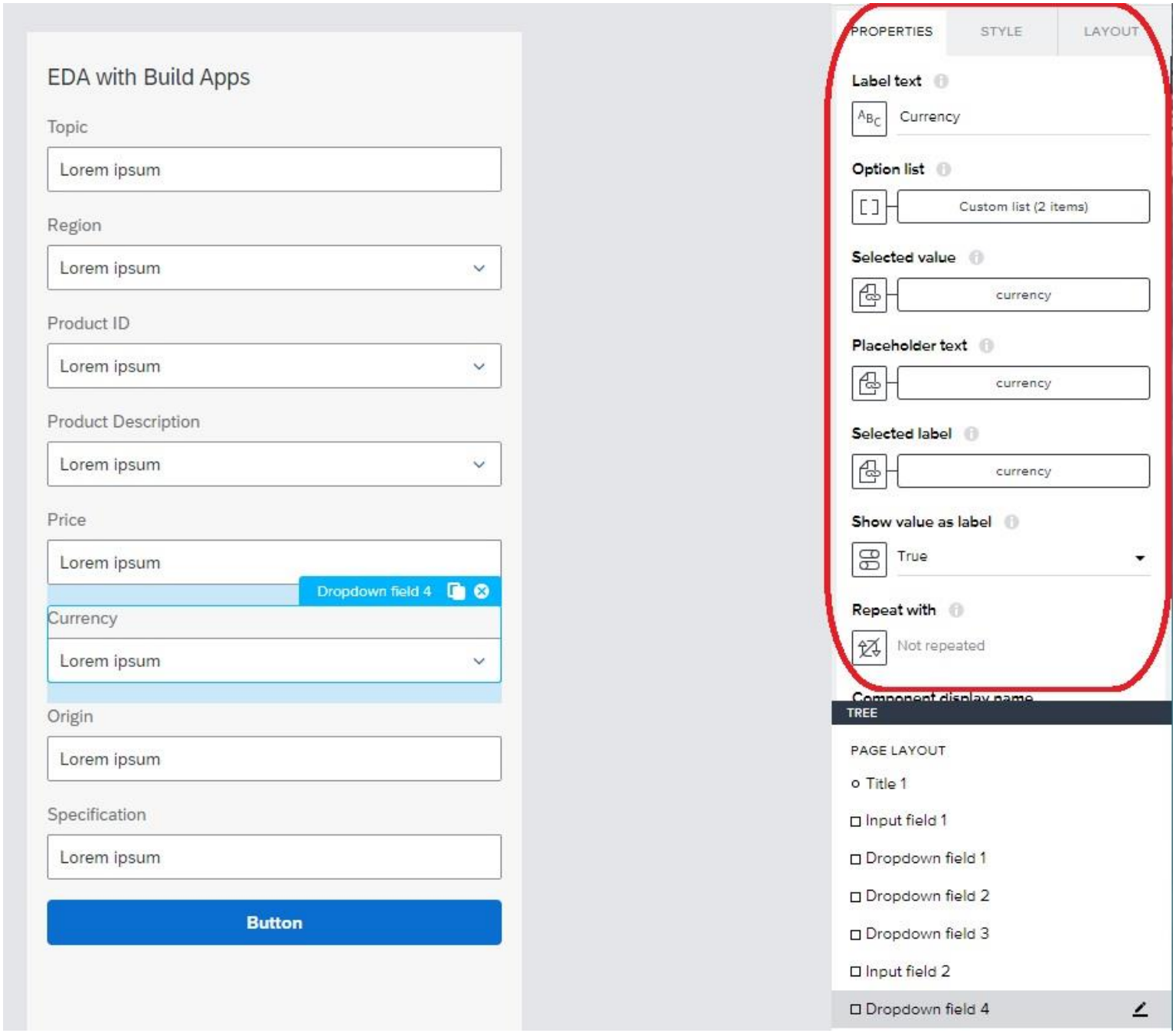
Option List

- GBP
- EUR
- USD
- YEN

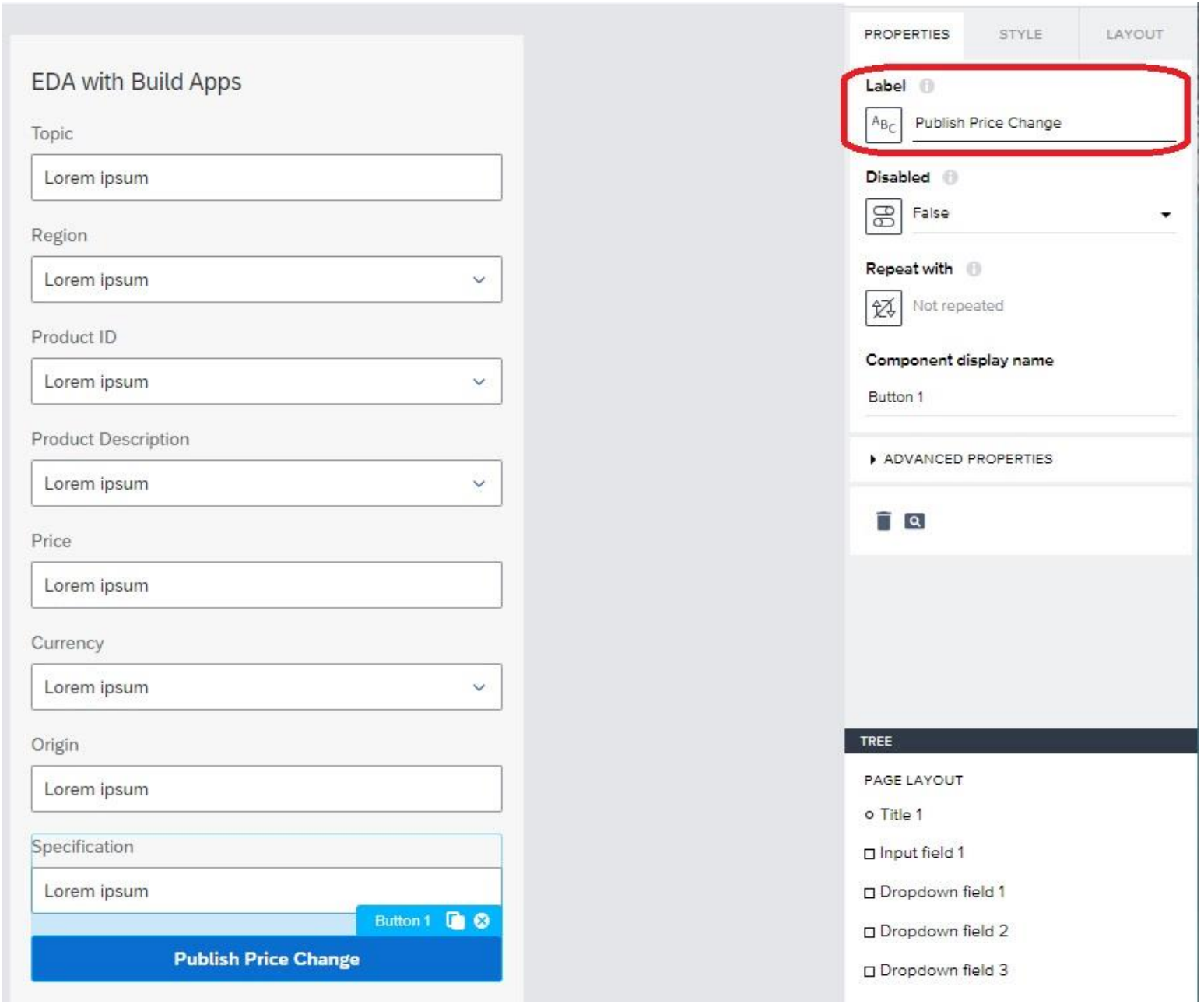
Again, bind all the various fields to your currency page variable as before for region & productid.

And also set “Show value as label” to “True”.

Double check that your configuration looks like this:



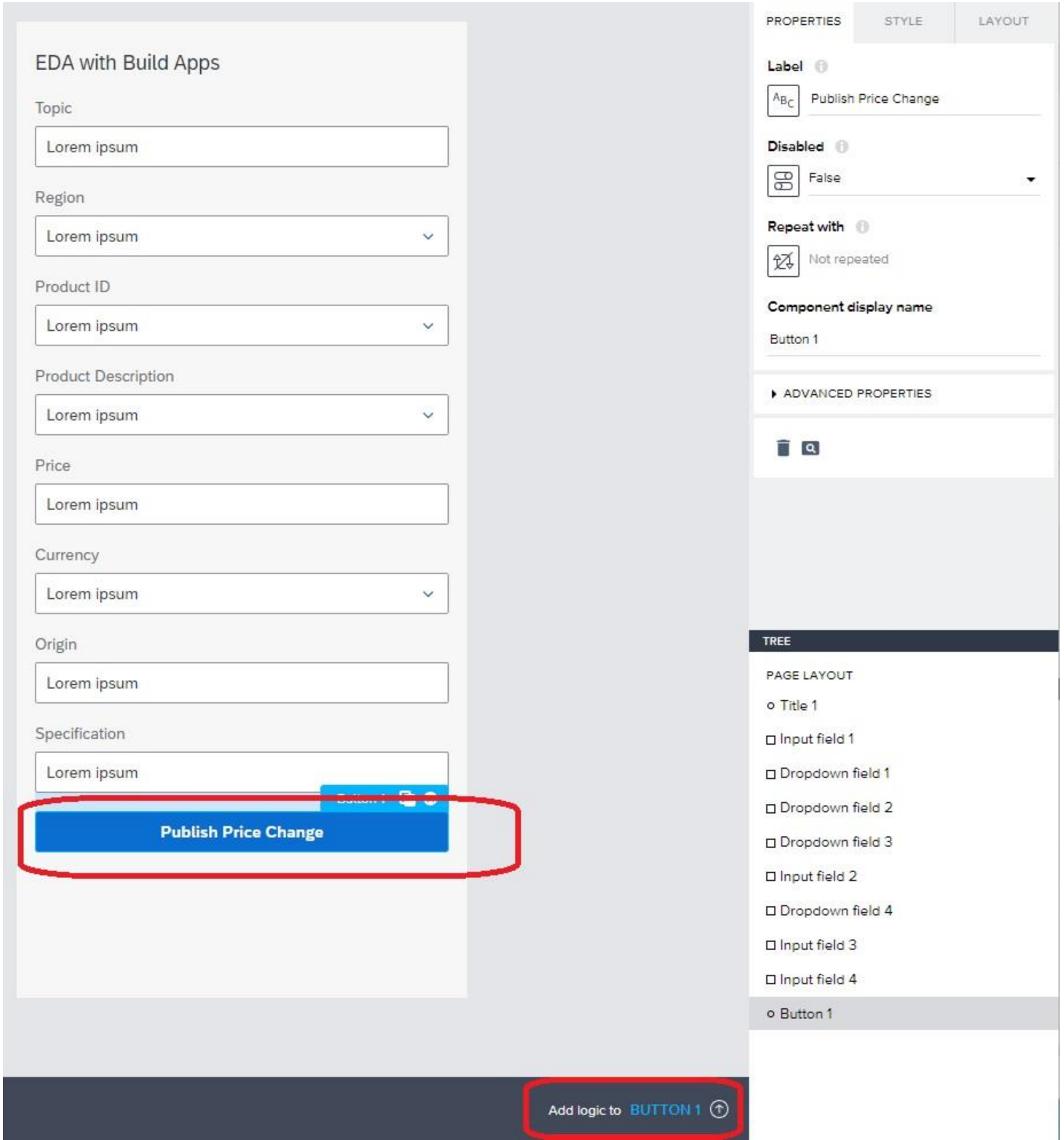
Just one small change left and the form is complete, change the label on the Button to read “Publish Price Change”.

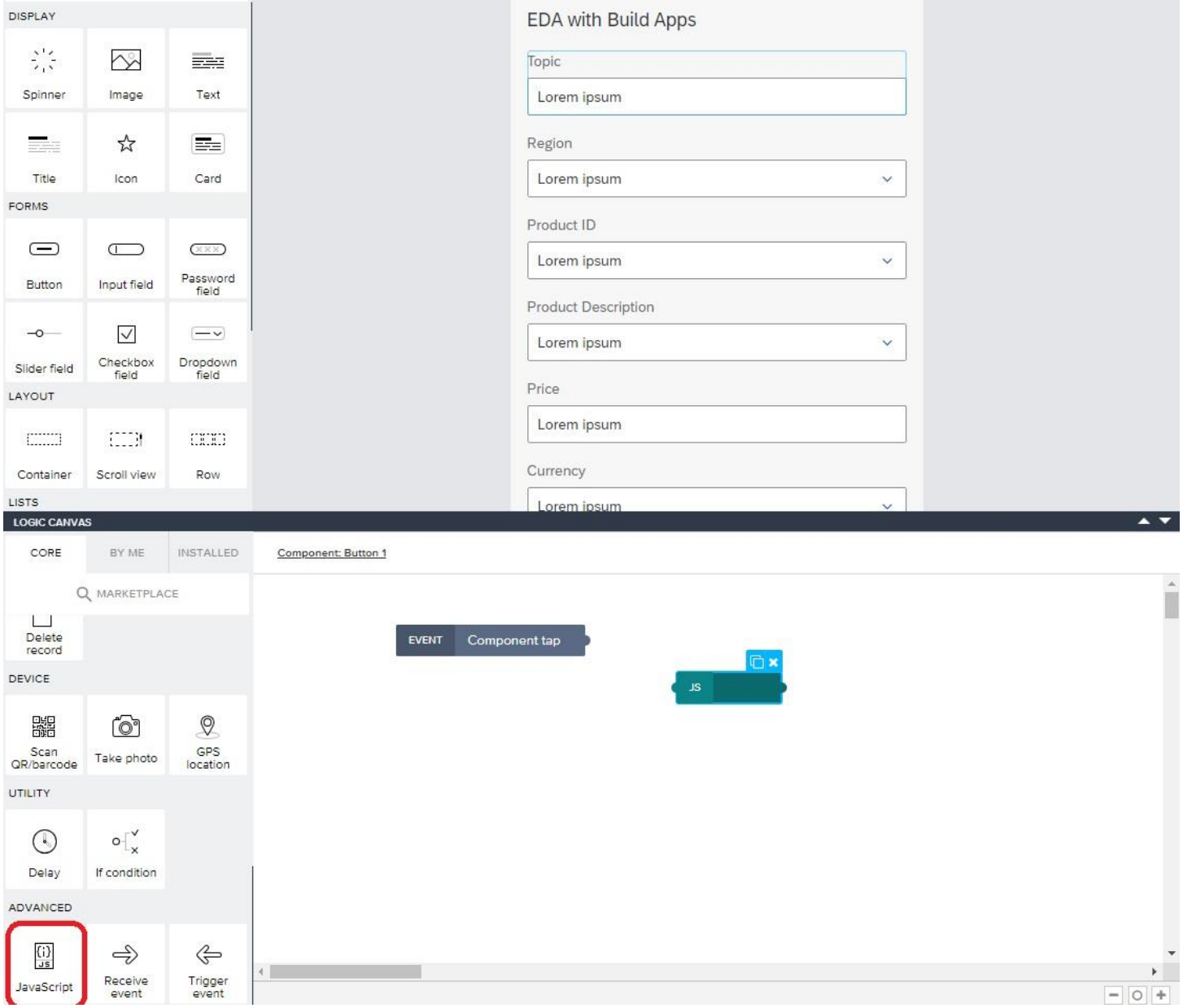


Congratulations, your form is now complete and we must implement the logic behind the “Publish Price Change” button.

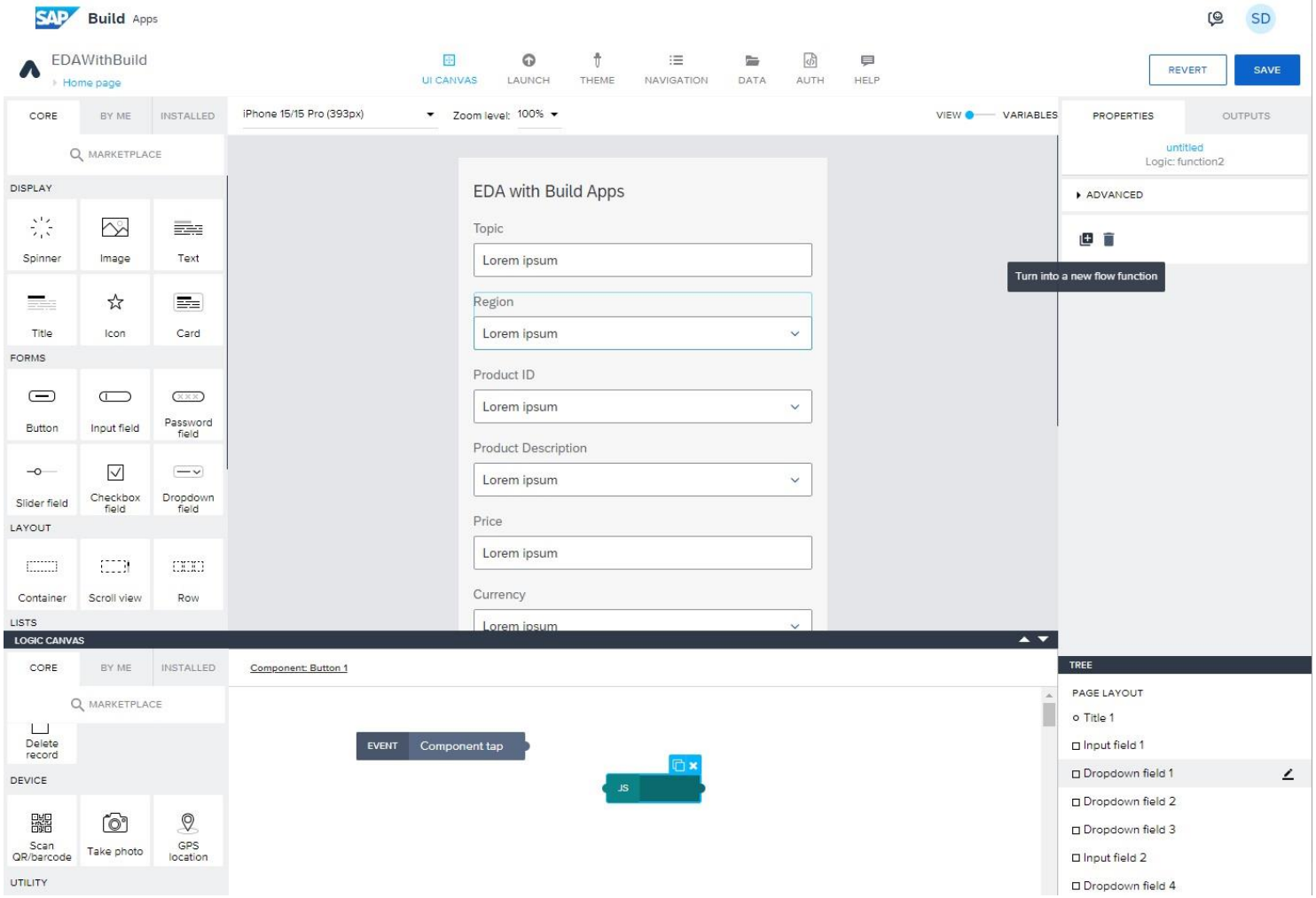
## Step 3 – Adding Logic to our Form to Publish an Event

At the moment, SAP Build Apps does not natively support the publication of Events, however they do support the use of Javascript Libraries. So in this next section, we will be taking advantage of this feature to load the libraries necessary to publish an event to the Advanced Event Mesh. When you click on the button, you will see text at the bottom of the screen “Add Logic to Button1”, click this section to reveal another panel.

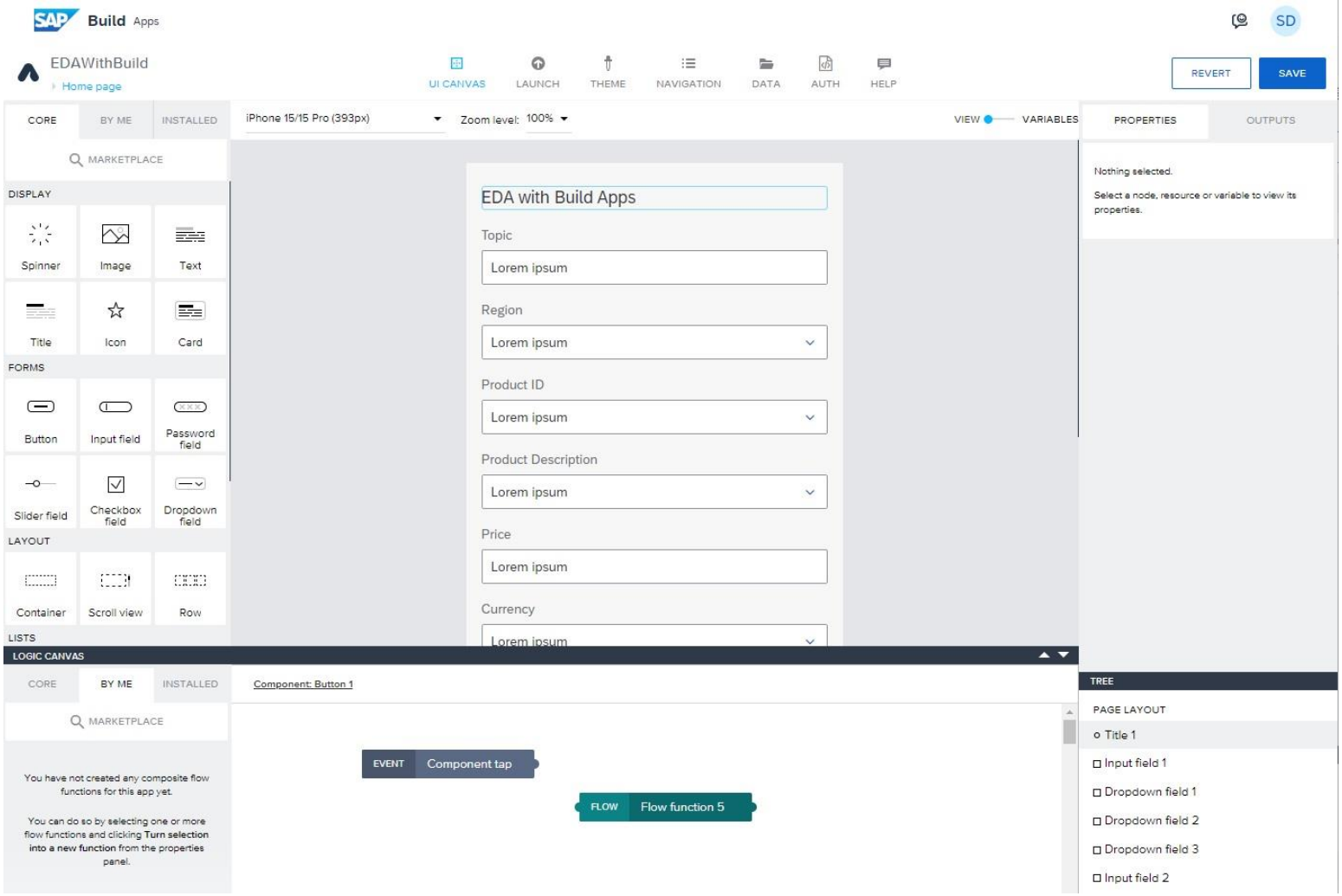





On the left hand side, you will scroll down to the “Advanced” group of widgets and you will drag and drop the Javascript widget onto the pallet as in the screenshot.




Now we will convert that JS component into a Flow function. With the Javascript widget selected, you should see on the upper right side a little “+” symbol. Click the Plus symbol to add a flow function that we will now configure. You will notice that your JS object has magically changed as per the next screenshot.




Voila, you have a Flow Function to work with. Now “double click” on the Flow Function.

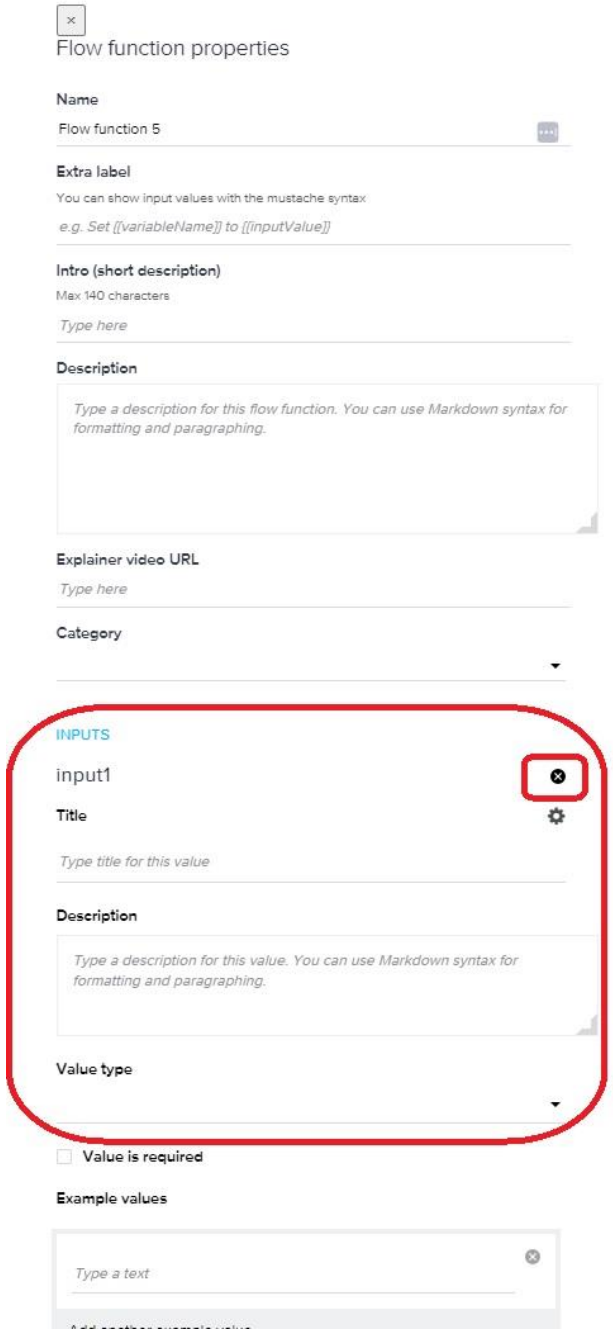
Double Click on the “Input” Box (the flow function graphic) to reveal a screen where you will specify the input parameters.

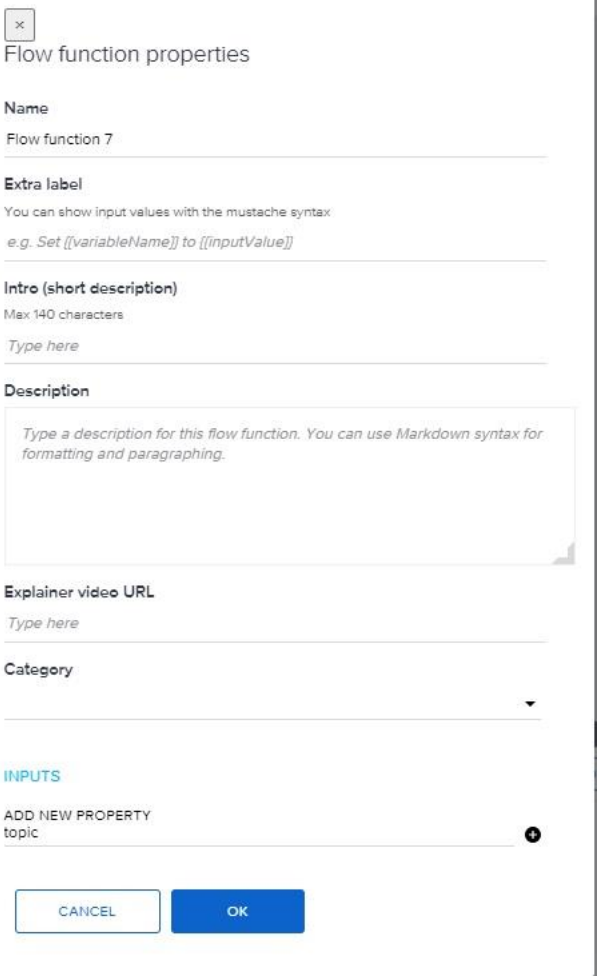

From this next screen, you will be defining the same properties that we defined on the form and the same properties we defined for page variables.

- topic Value Type “Text”
- region Value Type “Text”
- origin Value Type “Text”
- specification Value Type “Text”
- production_description Value Type “Text”
- product_id Value Type “Text”
- pricing Value Type “Text”
- currency Value Type “Text”

Click on “Edit Properties” to start.

Then click the X next to “input1” to remove the default property that is created.



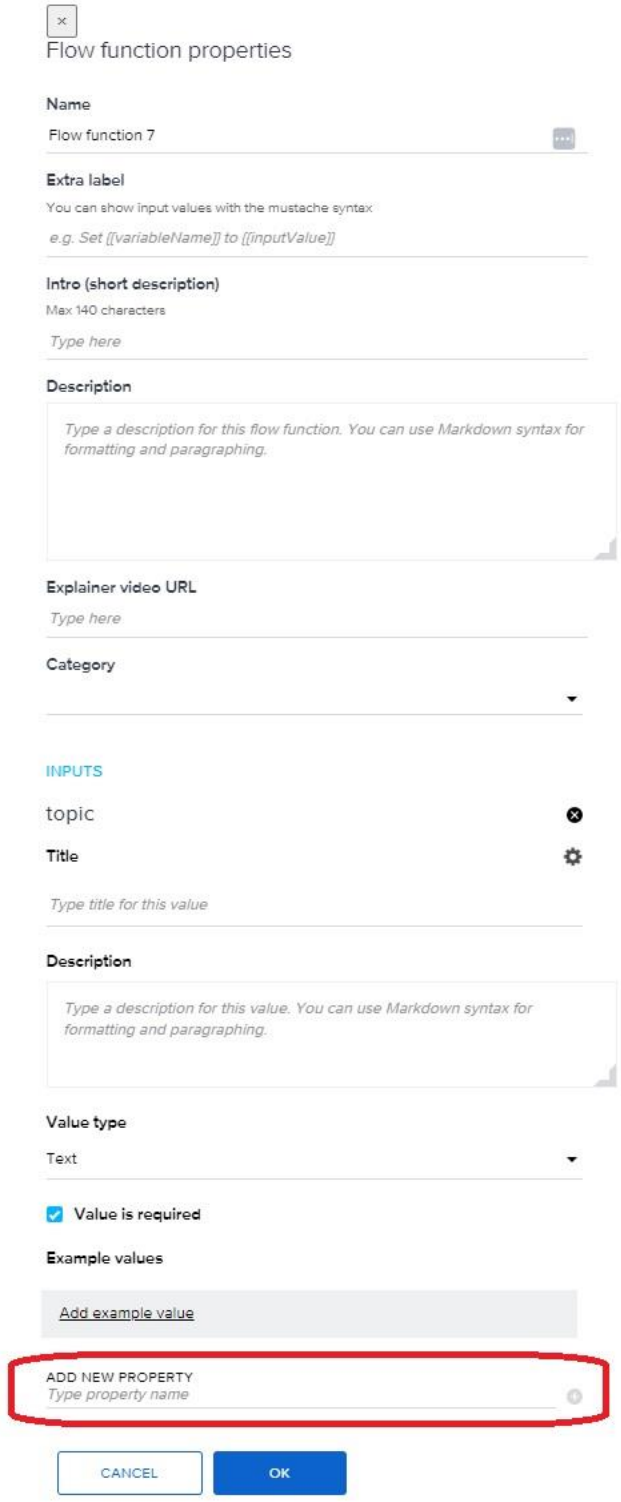
After you remove the default property, you will then enter “topic” as your first property to be added.

***DO NOT CLICK OK**** instead click the small plus sign to the right of the property input field to add the next field.

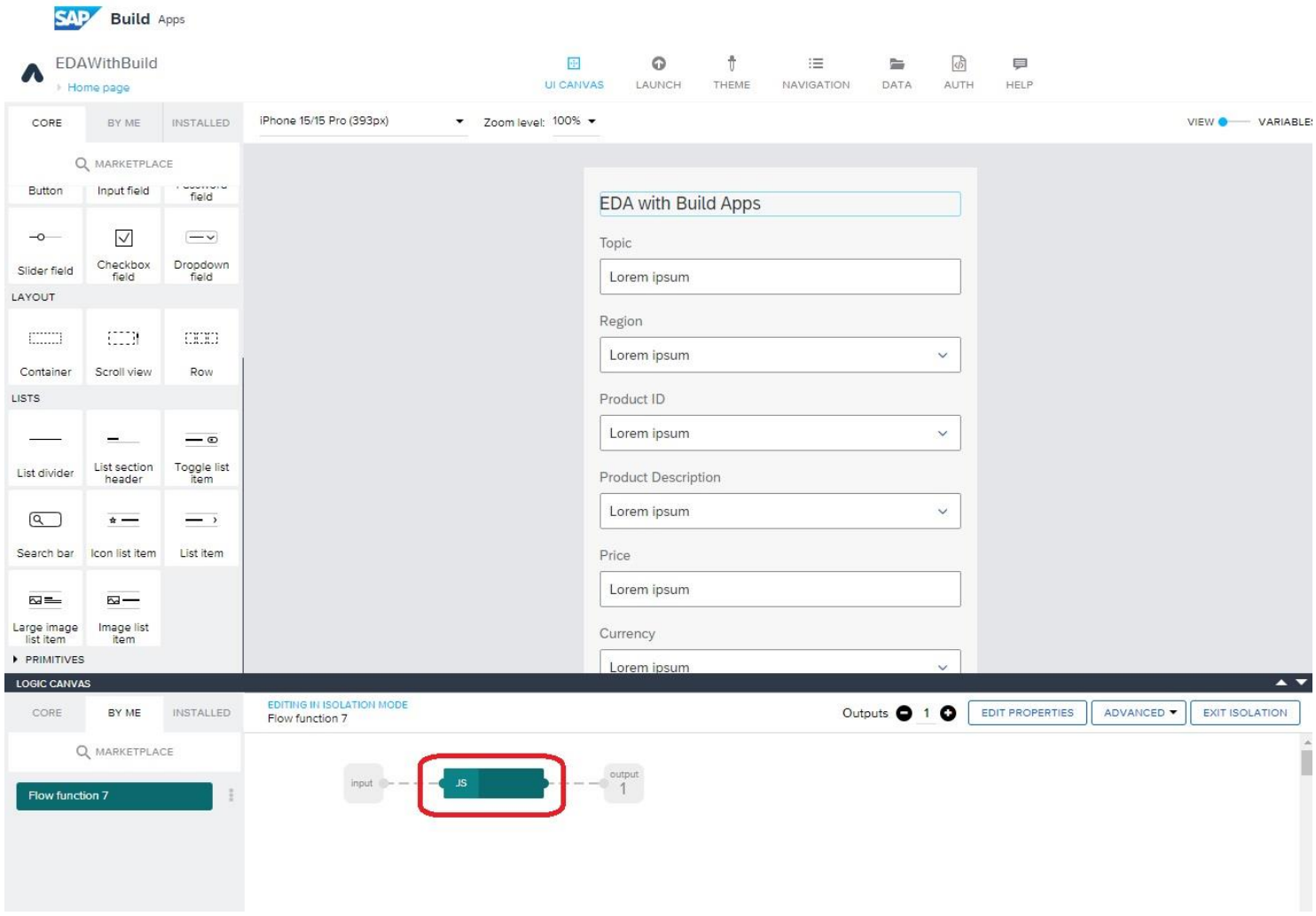
Once you click the “+” button, the property “topic” has been added and you can repeat the process for all of the other fields. Here’s the full list again:

- topic Value Type “Text”
- region Value Type “Text”
- origin Value Type “Text”
- specification Value Type “Text”
- production_description Value Type “Text”
- product_id Value Type “Text”
- pricing Value Type “Text”
- currency Value Type “Text”

Only click the OK, once all the properties have been added.



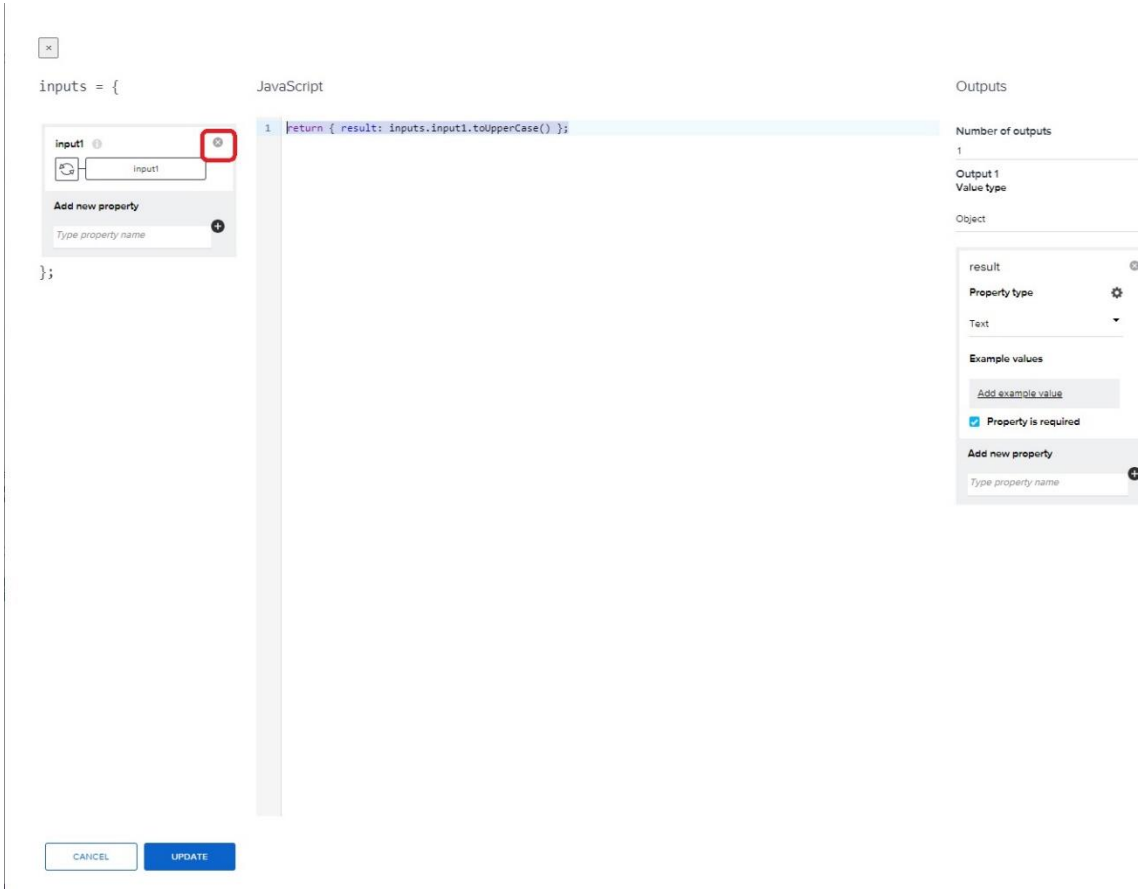
Once you click “OK” you will be back to this screen. From here you will double click on the JS Box to enter the Javascript code that will be executed when the button is pressed and also define the variables that the Javascript will use.



Now that you have the JS window open, we need to do 2 things:

1. Recreate all the same variables as above. Prior to creating the variables, **REMOVE** the default input parameter by clicking the little “x” in red.
    - topic Value Type “Text”
    - region Value Type “Text”
    - origin Value Type “Text”
    - specification Value Type “Text”
    - product_description Value Type “Text”
    - product_id Value Type “Text”
    - pricing Value Type “Text”
    - currency Value Type “Text”

    You will do the same process as before, type the variable name into the field and hit the little “+” sign.



2. Paste the JavaScript Code Below into the JavaScript window and replace what is there.

**** JAVASCRIPT BEGINS****

```
const { topic, pricing, region, product_id, product_description, origin, specification, currency } = inputs;
const { session } = self.solace;
const destination = createDestination(topic, 'topic');
publishMessage(destination, pricing, region, product_id, product_description, origin, specification, currency);
function createDestination(name, type) {
 if (type === 'queue') {
   return solace.SolclientFactory
     .createDurableQueueDestination(name);
 }
 if (type === 'topic') {
   return solace.SolclientFactory
     .createTopicDestination(name);
 }
}
function publishMessage(publishDestination, pricing, region, product_id, product_description, origin, specification, currency) {
 const message = solace.SolclientFactory.createMessage();
 const payload = {
   pricing: pricing,
   region: region,
   product_id: product_id,
   product_description: product_description,
   origin: origin,
```

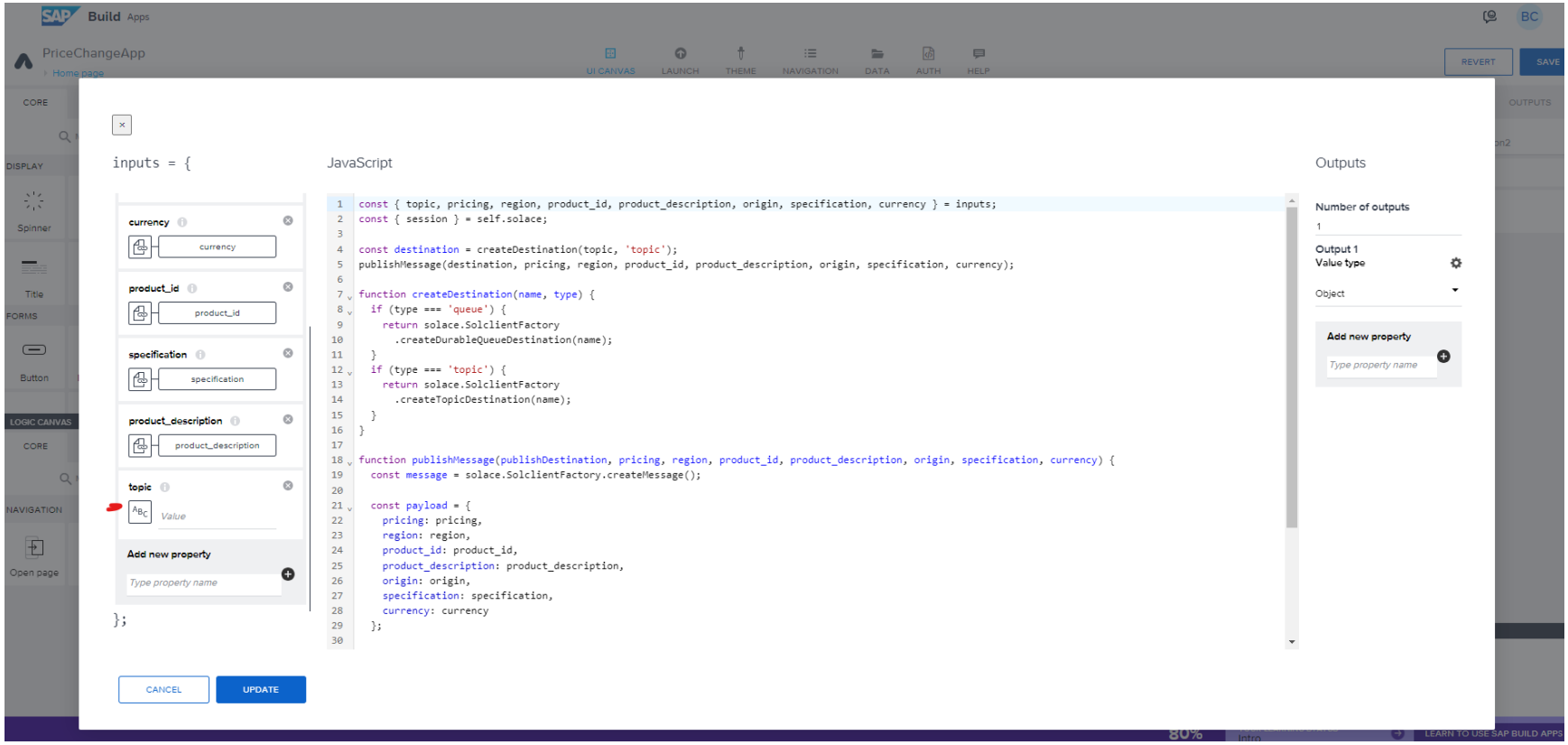
```
    specification: specification,
    currency: currency
  };
  message.setDestination(publishDestination);
  message.setBinaryAttachment(JSON.stringify(payload));
  session.send(message);
}
```

****JAVASCRIPT ENDS*****

Once the variables are created and the Javascript is pasted in, you should have a screen that looks like this.

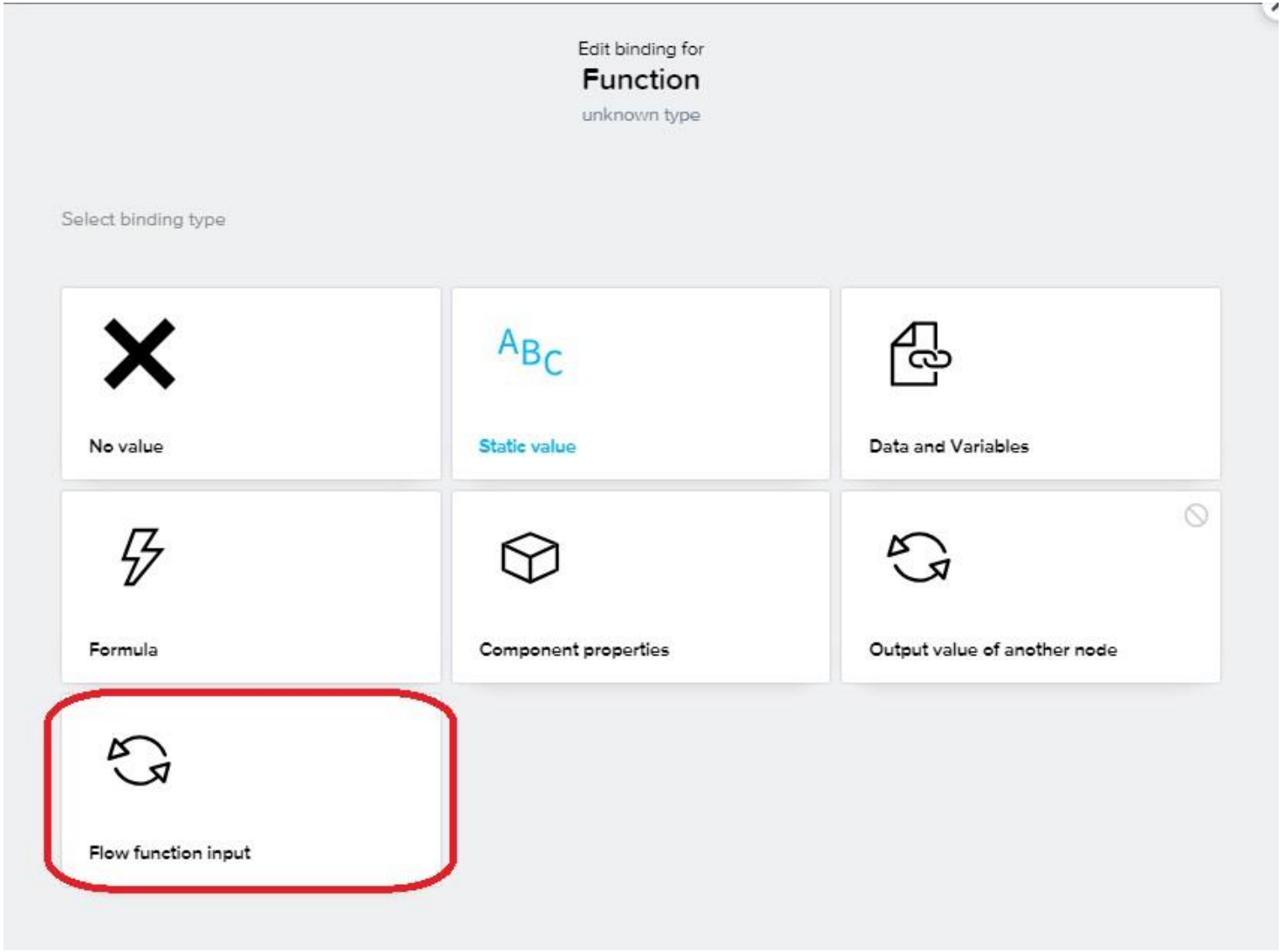
You should have 8 variables defined on the left as input to the function.

BUT, if you notice there is nothing bound to the function at this point. Now we will select each of the variables and bind them to the flow function variables you defined in the previous step. You will start by clicking on the ABC that is highlighted. Yours will be unbinded to start.

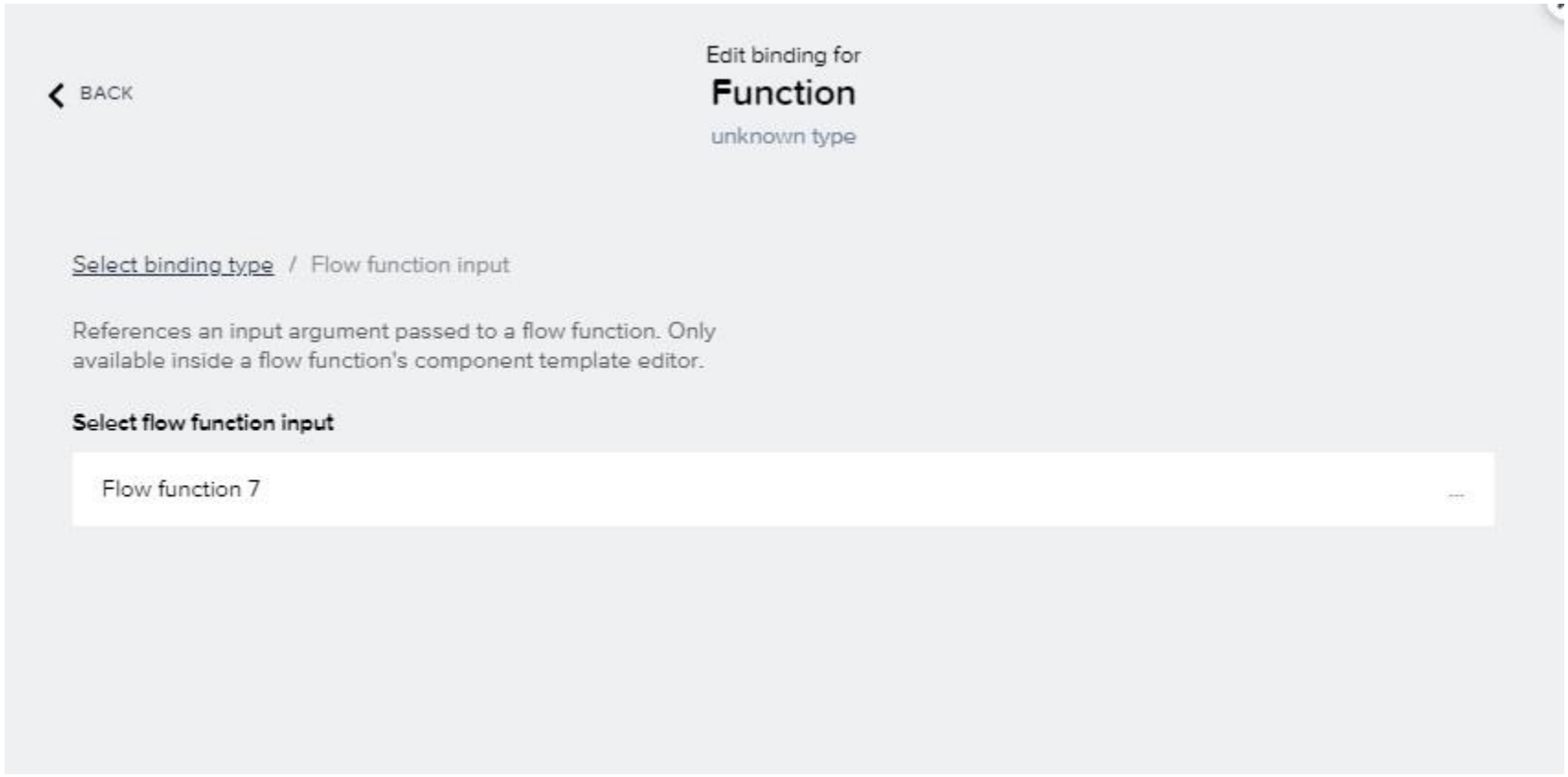


Select “Flow Function Input”

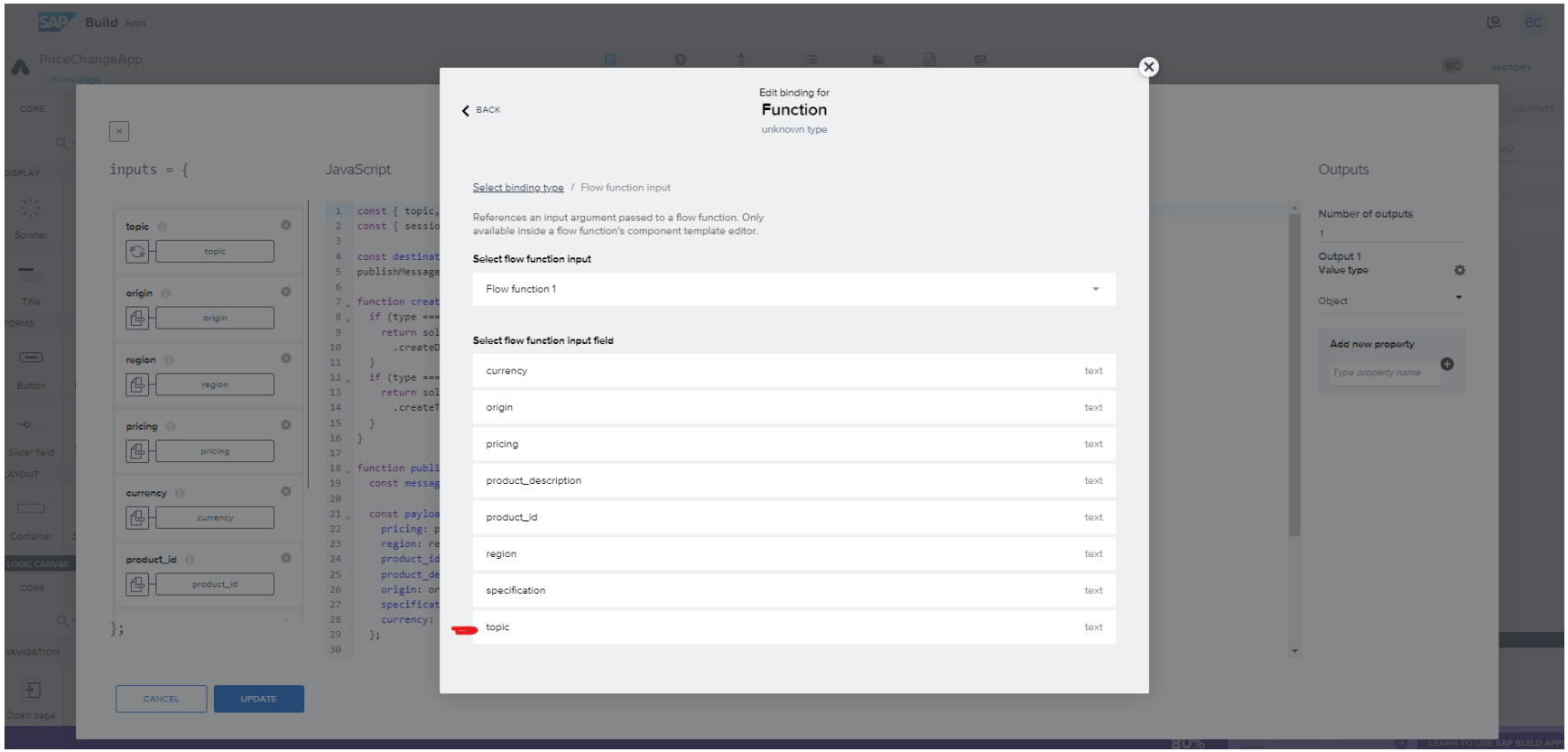
Your flow function will likely be labelled Flow Function 1 (***I’ve done 7 iterations 😊 ***).



Click the Flow Function to reveal the variables available.



From the list, select the input field being mapped, in this case “topic”.




You have a chance to validate here before pressing “SAVE”

BACK

Edit binding for

## Function

unknown type

Select binding type / Flow function input

References an input argument passed to a flow function. Only available inside a flow function's component template editor.

**Select flow function input**

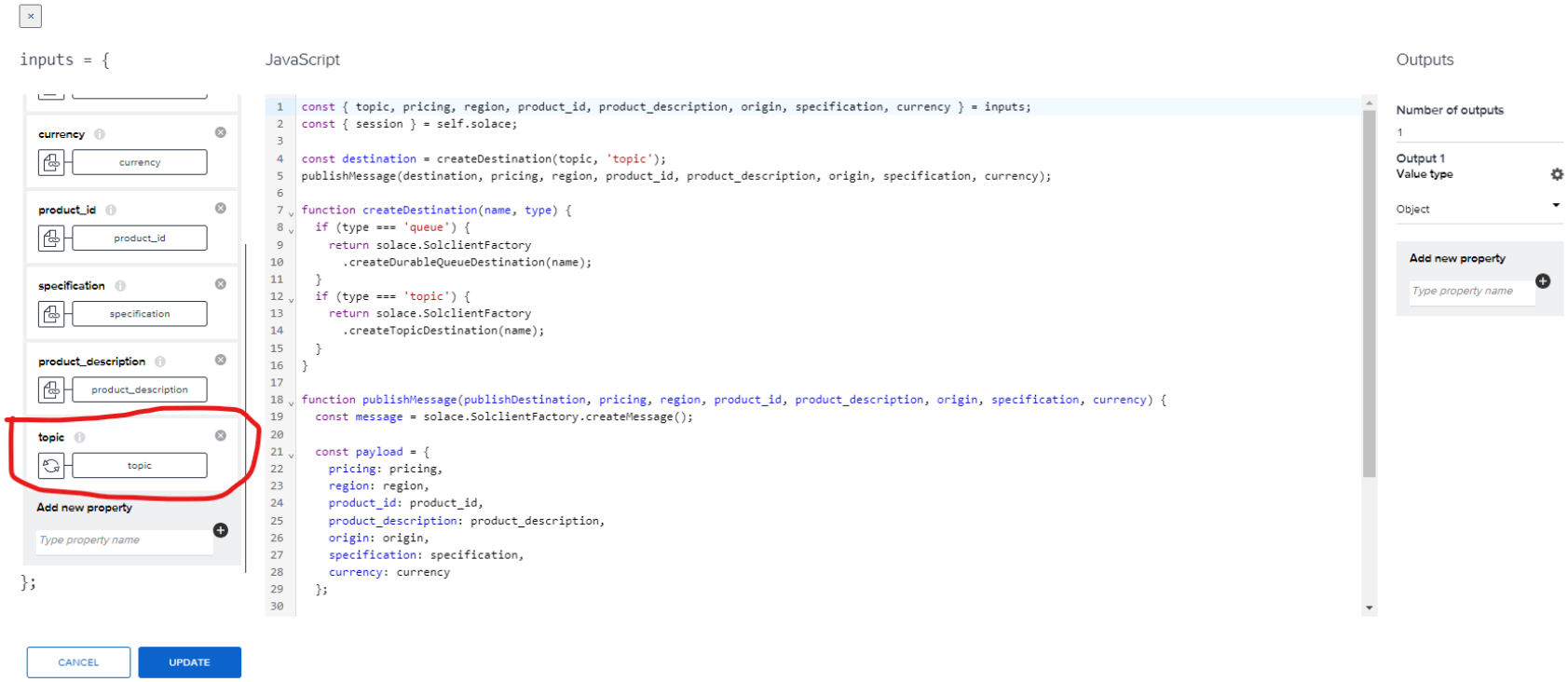
Flow function 7

**Select flow function input field**

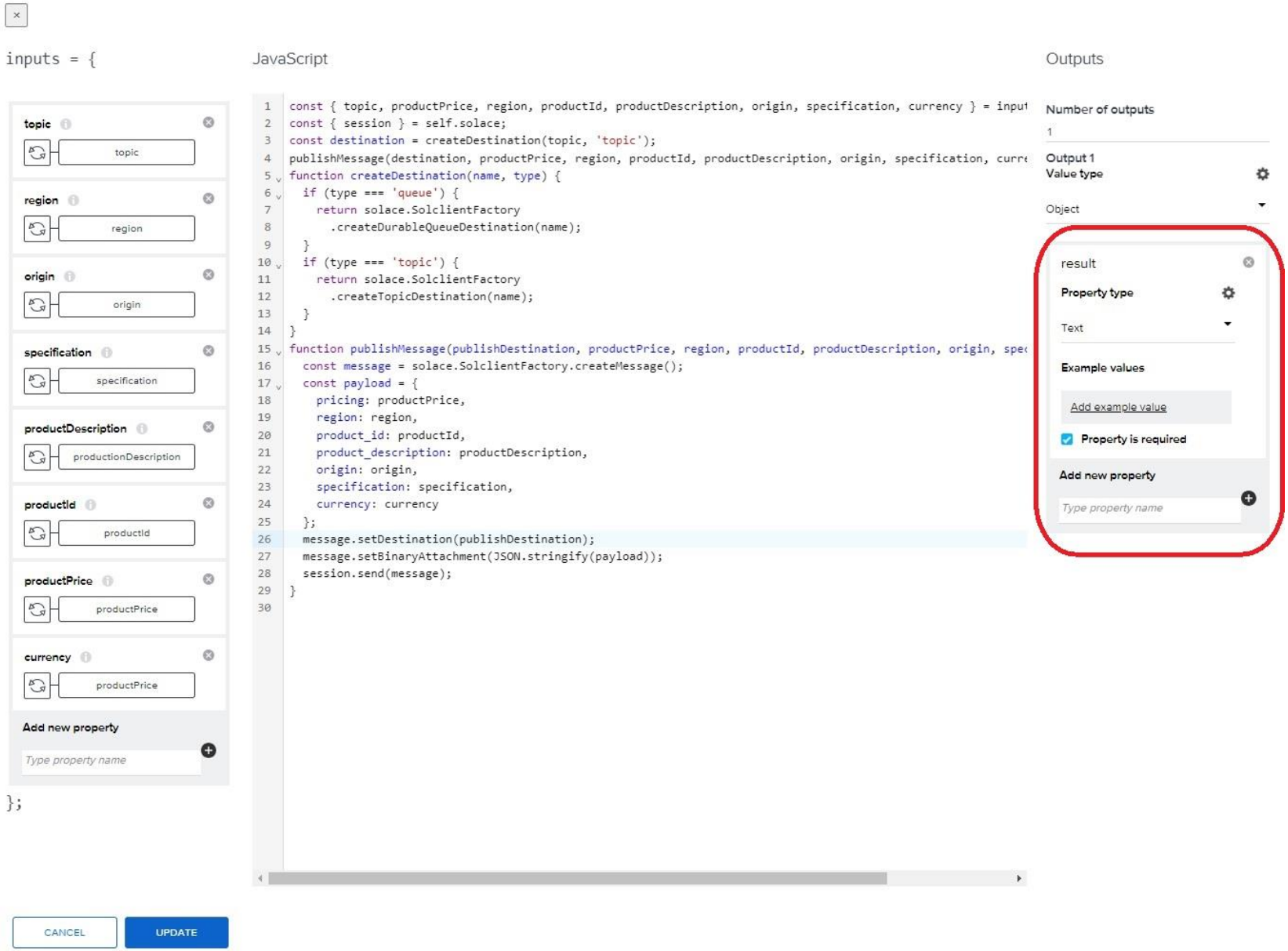
topic

SAVE

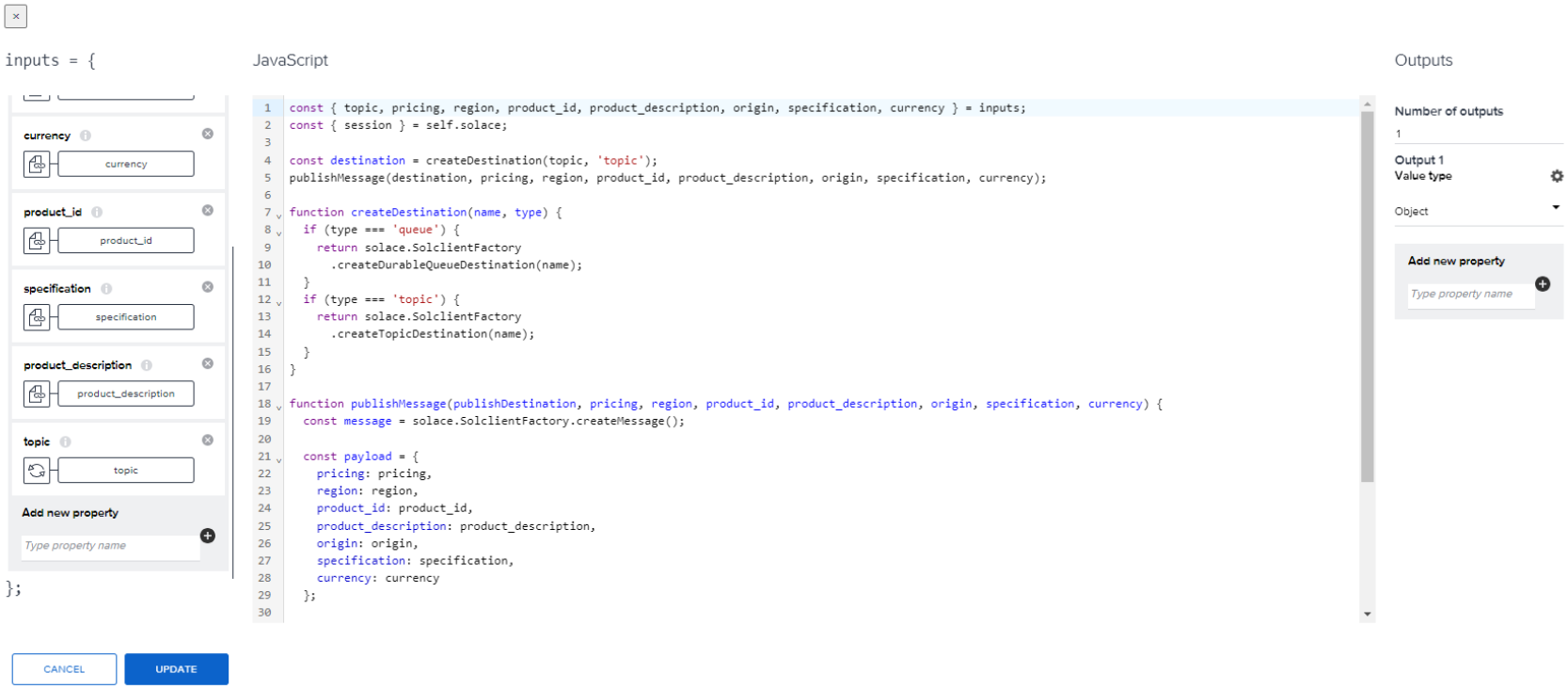
After saving the first one, it should look like this and if you hover over the topic you should see that it is mapped to the flow function input.



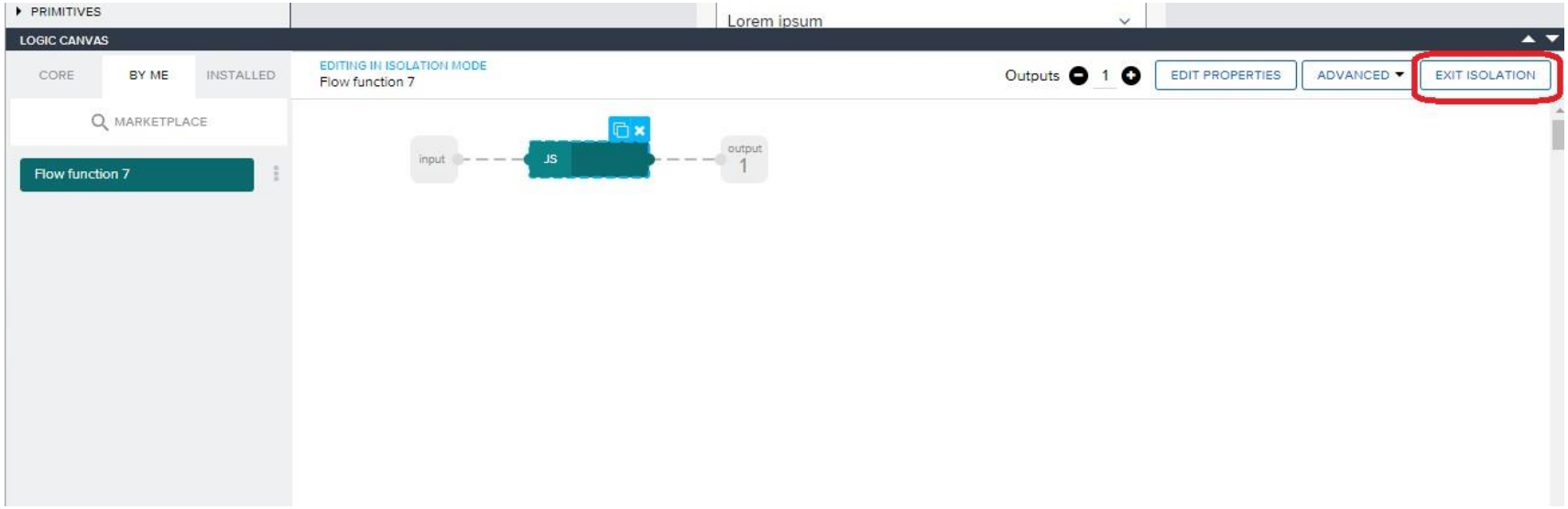
Once completed, it should look like this but one final update. Each time you add a JS widget, it adds a default Output Parameter which you can see highlighted in RED. You will need to remove this.



This is the final result that you should see.

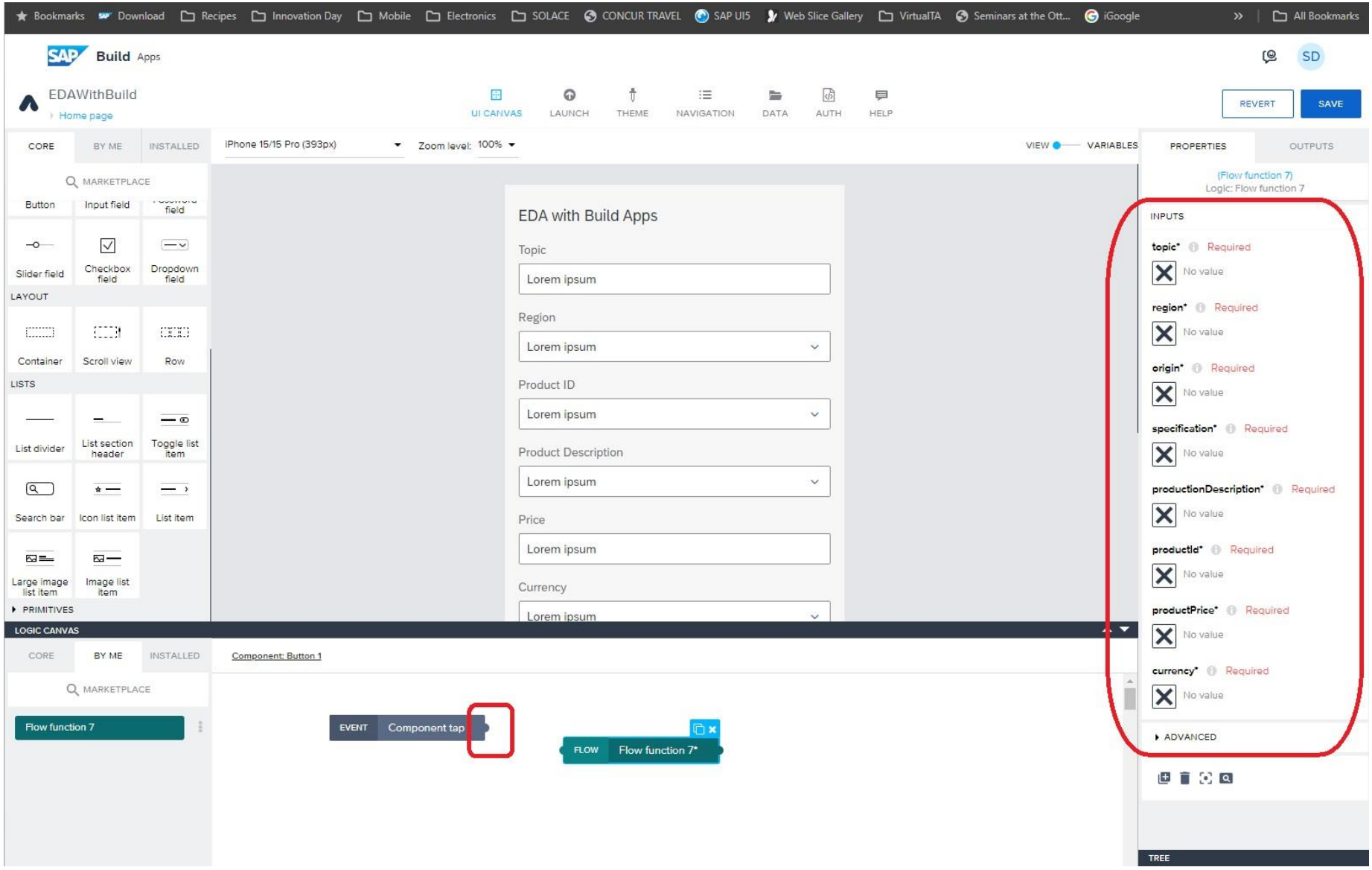


After pressing “UPDATE”, you will return to the JS function. Press the “Exit ISOLATION” button.

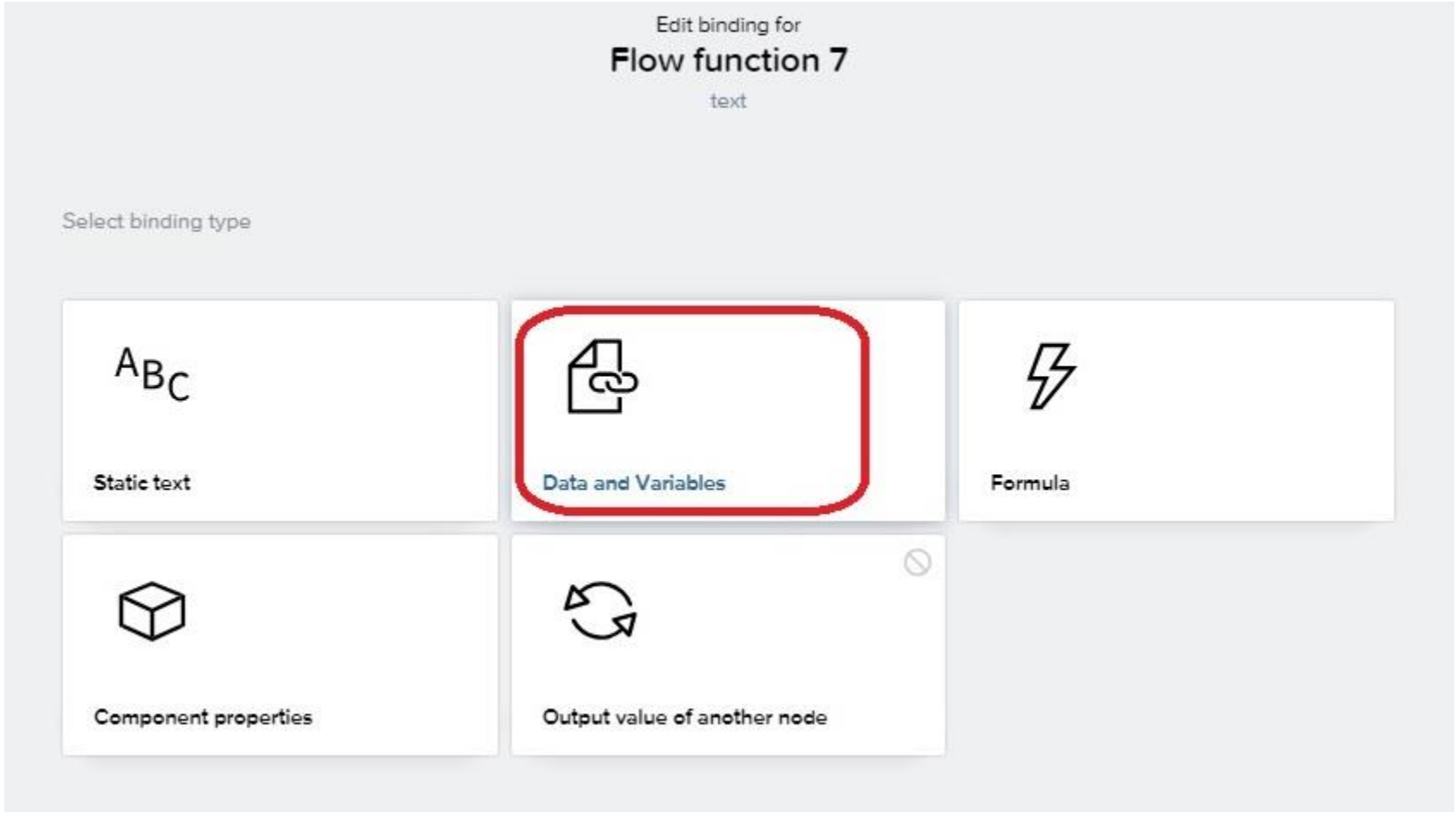


From here, we have 2 things to do:

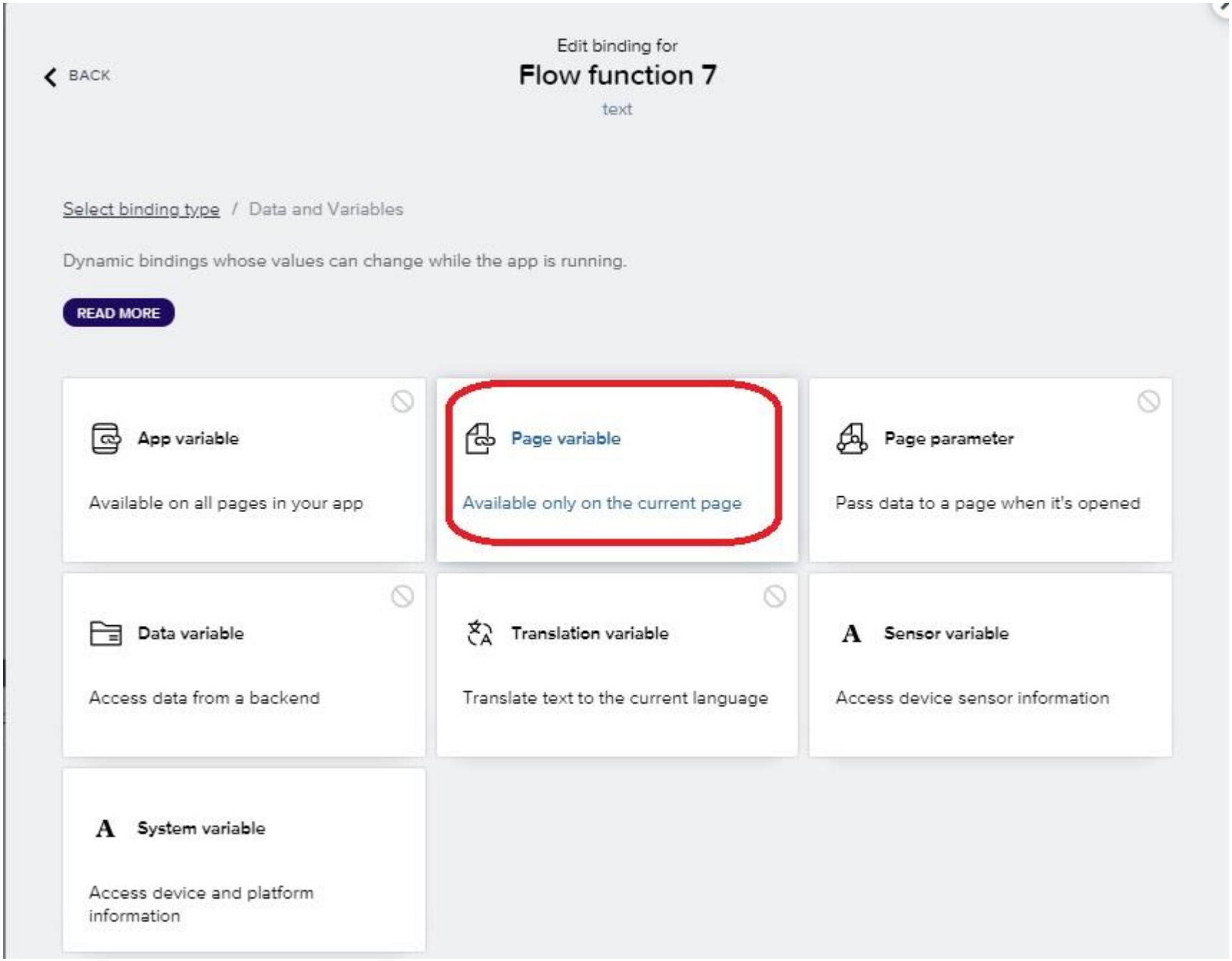
1) Bind all of the Flow Function Inputs to our page variables (click on the flow function to reveal this properties pane).
2) Connect our “Component Tap” event to the Flow Function by simply draging a line between the dot that is highlighted by the red square to the dot on the flow function. Connecting these two components will trigger the flow function when the Component Tap event is pressed.



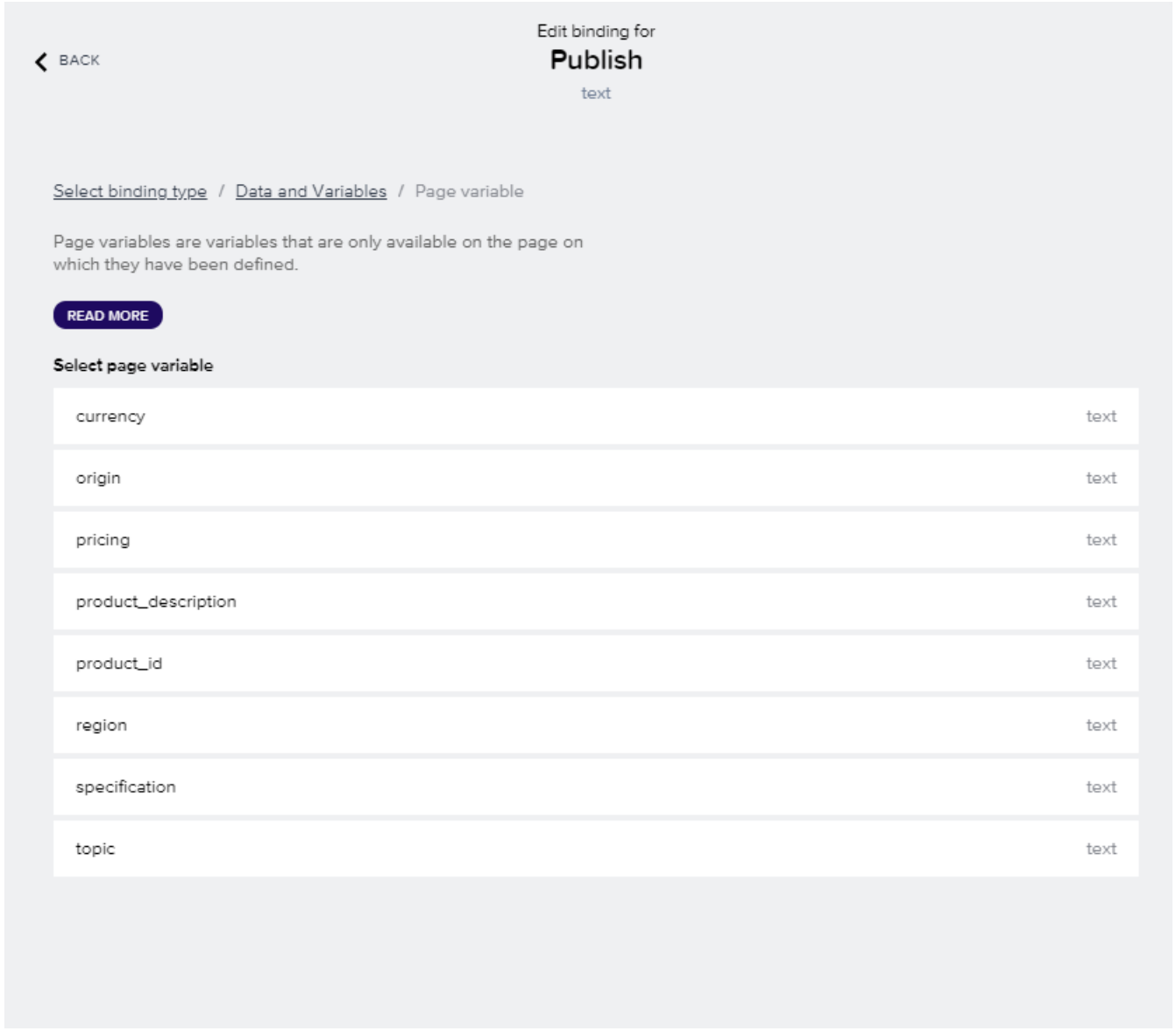
Let’s start by mapping the first variable. Click the “X” just underneath the topic input. Select Data and Variables.



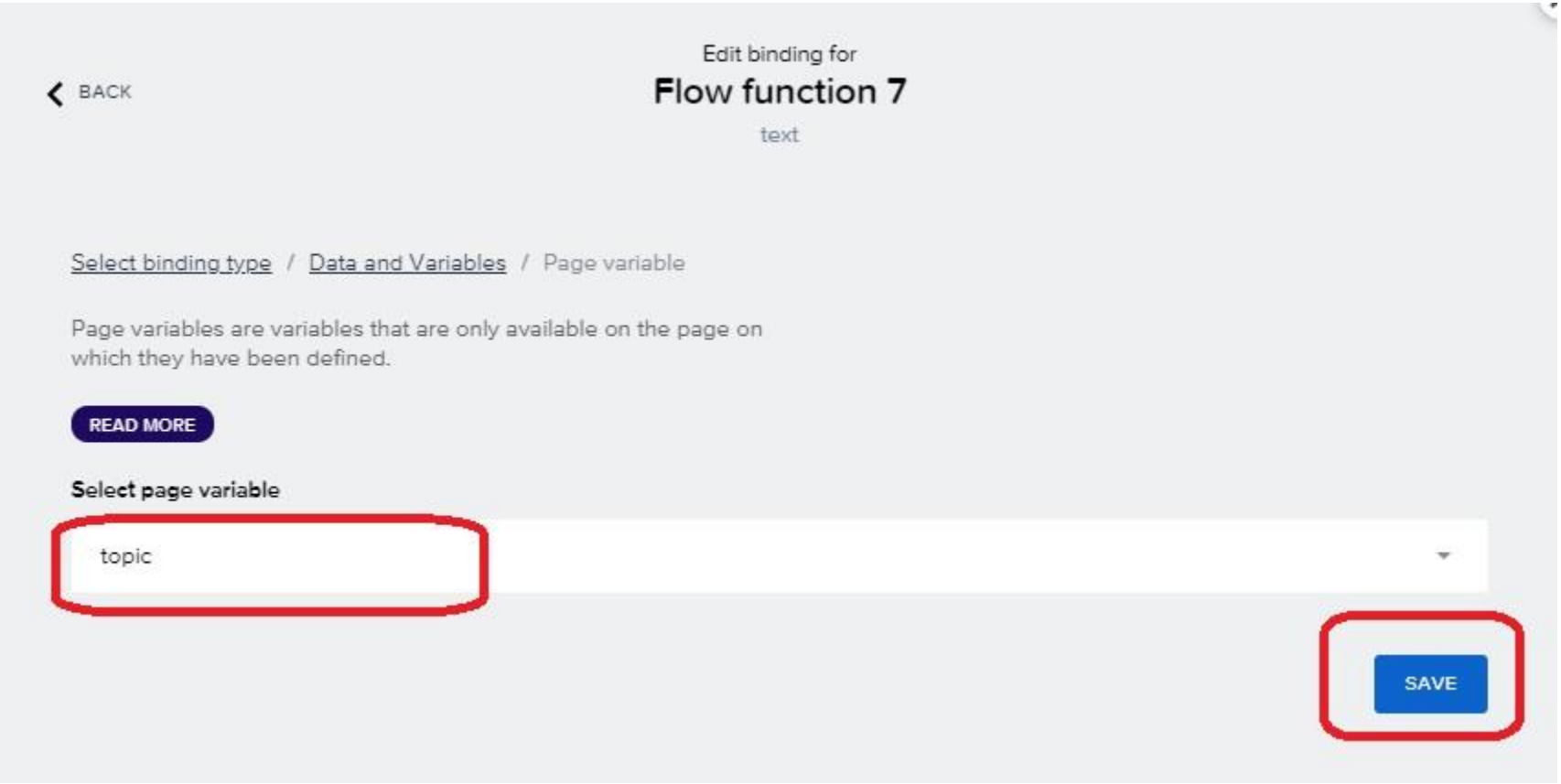
Select “Page variable”



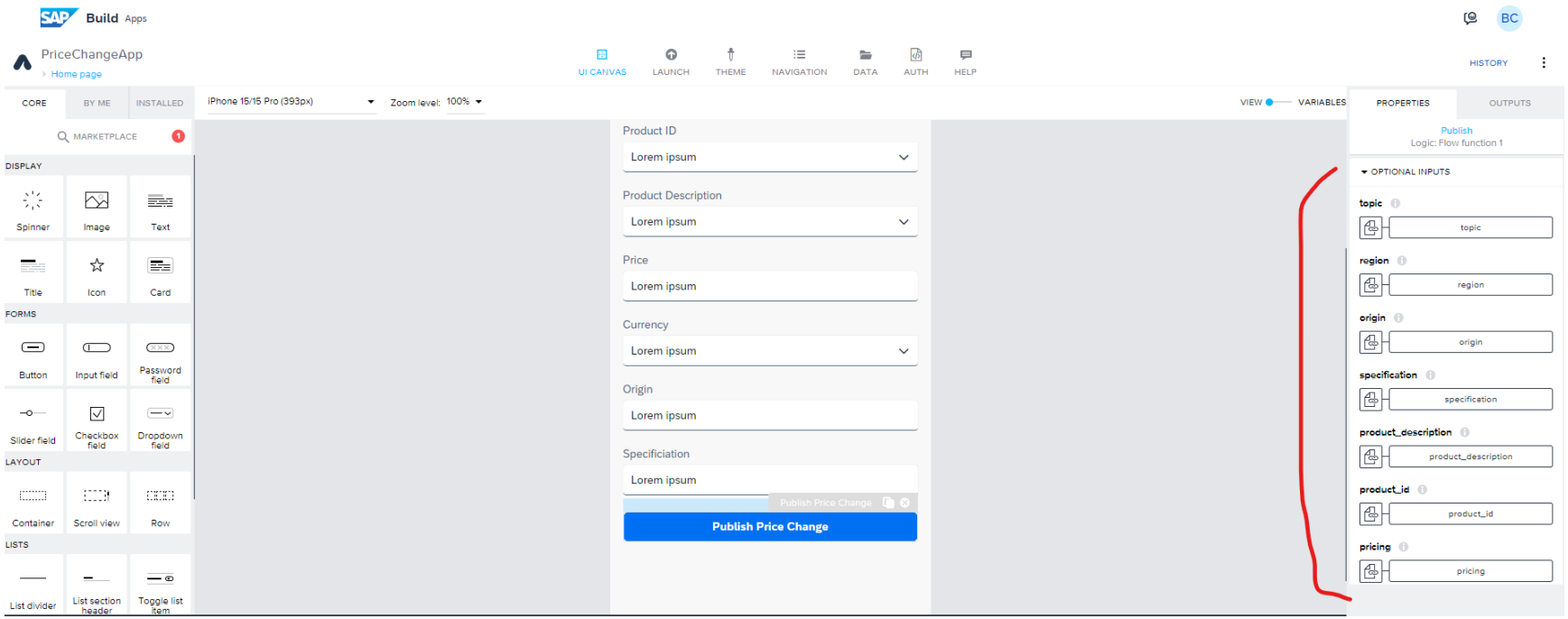
Select the page variable for “topic”



The system will prompt you before you click “SAVE”

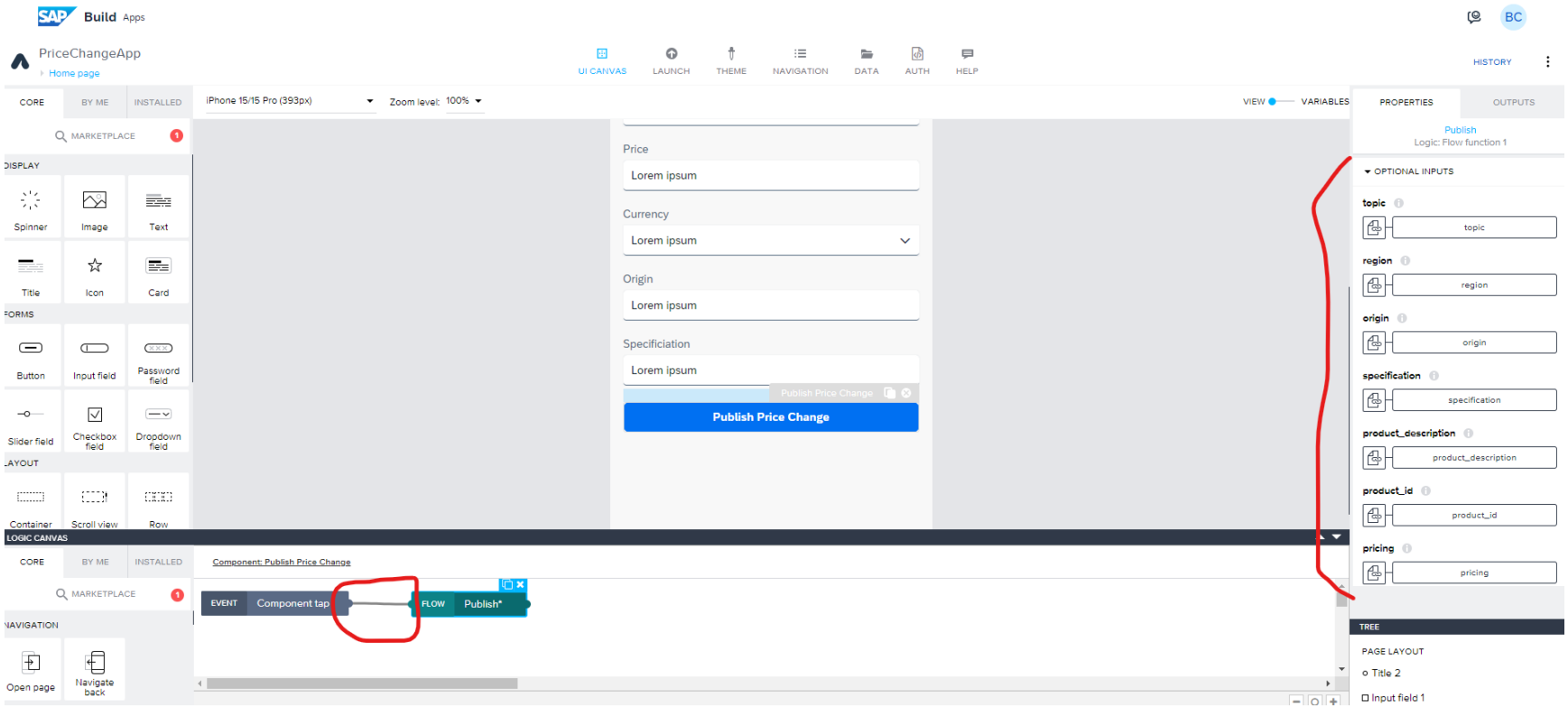


Now if you hover over the topic, SAP Build will display which page variable is currently bound to that input




Repeat the process for all of the remaining variables.

You should now have a screen that looks like this. Remember to connect the Component Tap Event to the flow function.

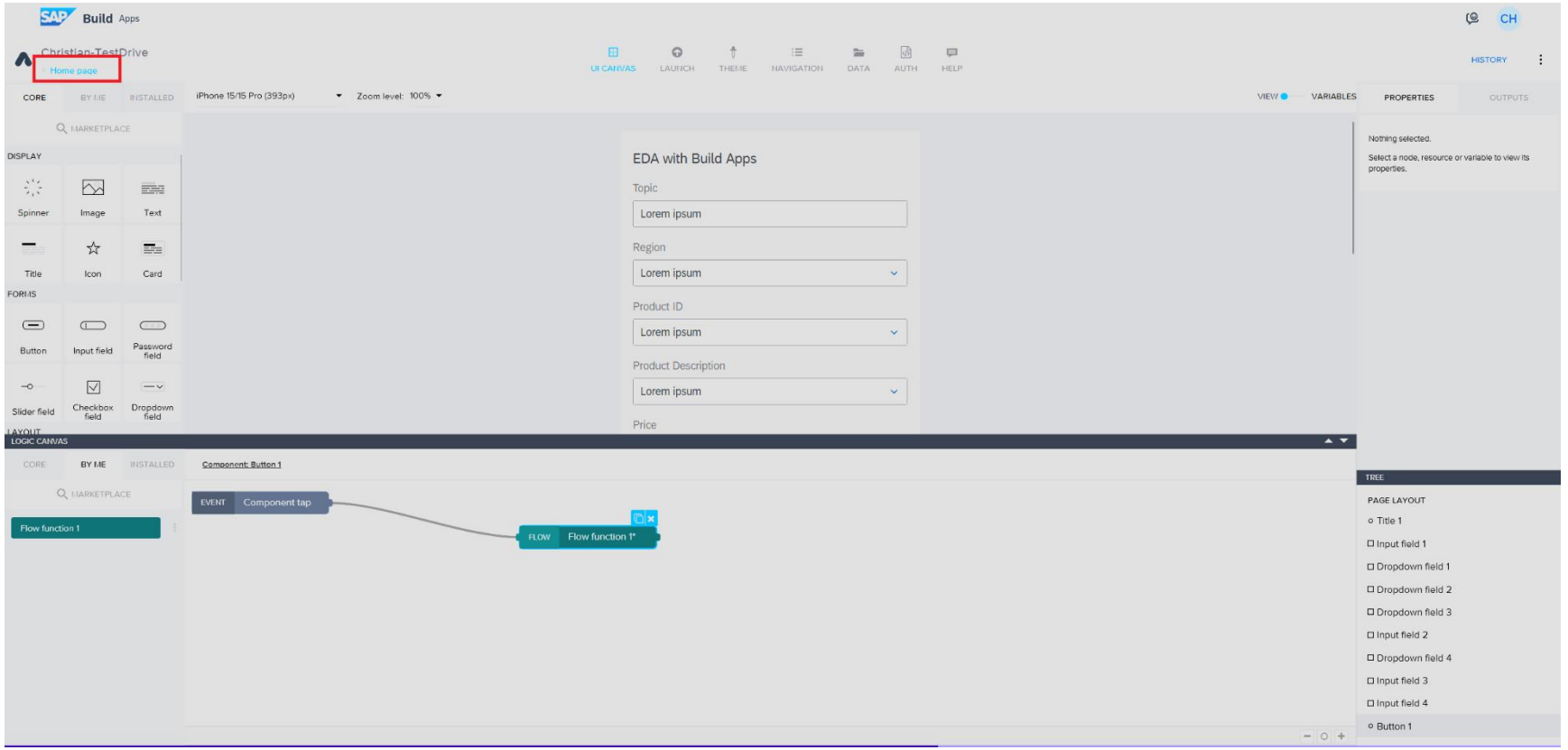



Hit Save at the top right when you are done.

# Step 4 – Loading the Javascript Libraries

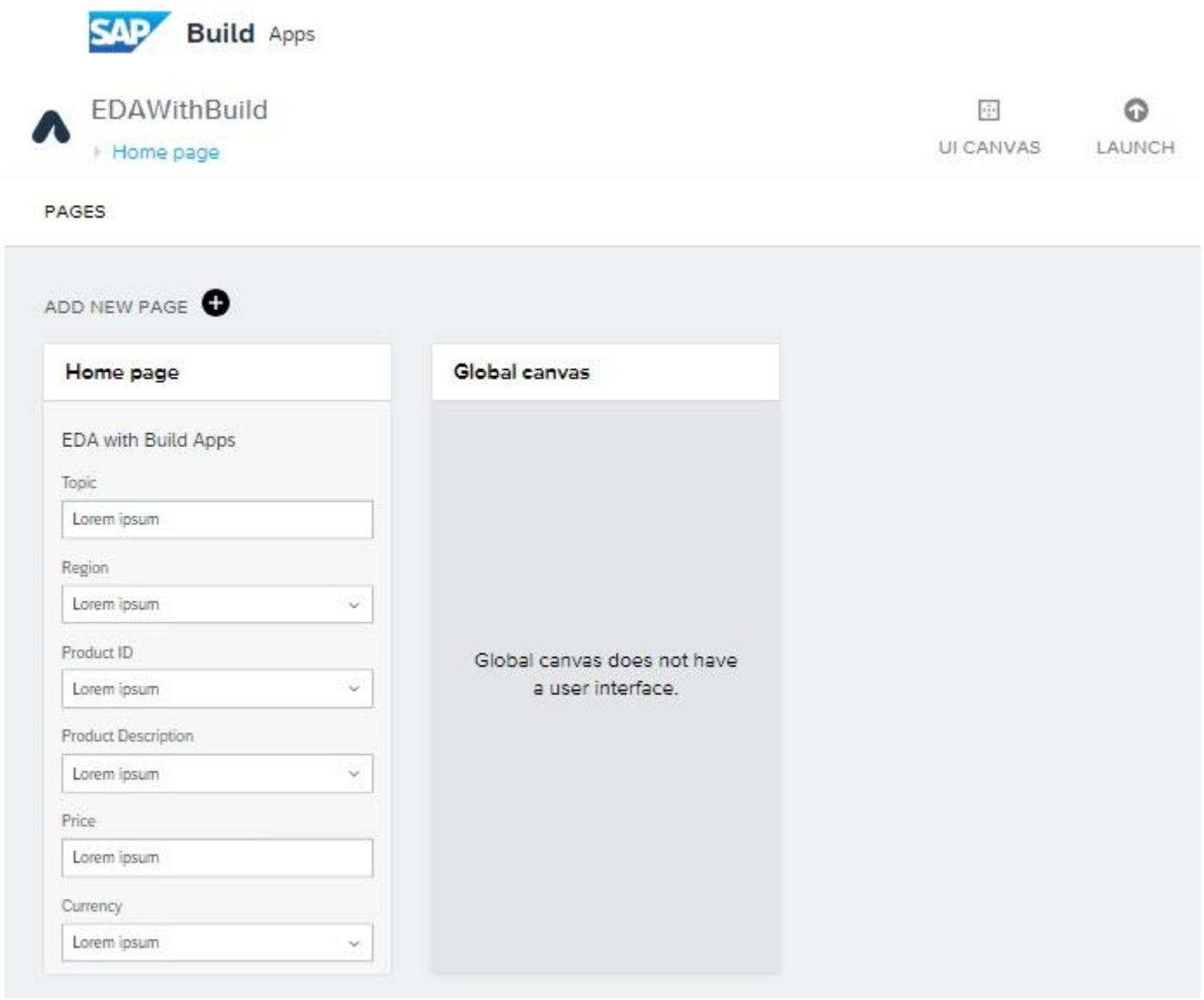
In order to be able to publish the events as per the Javascript code, we need to load the library that is used by AEM.

We will do this when the application starts. At the top left, you will see the HomePage is highlighted.

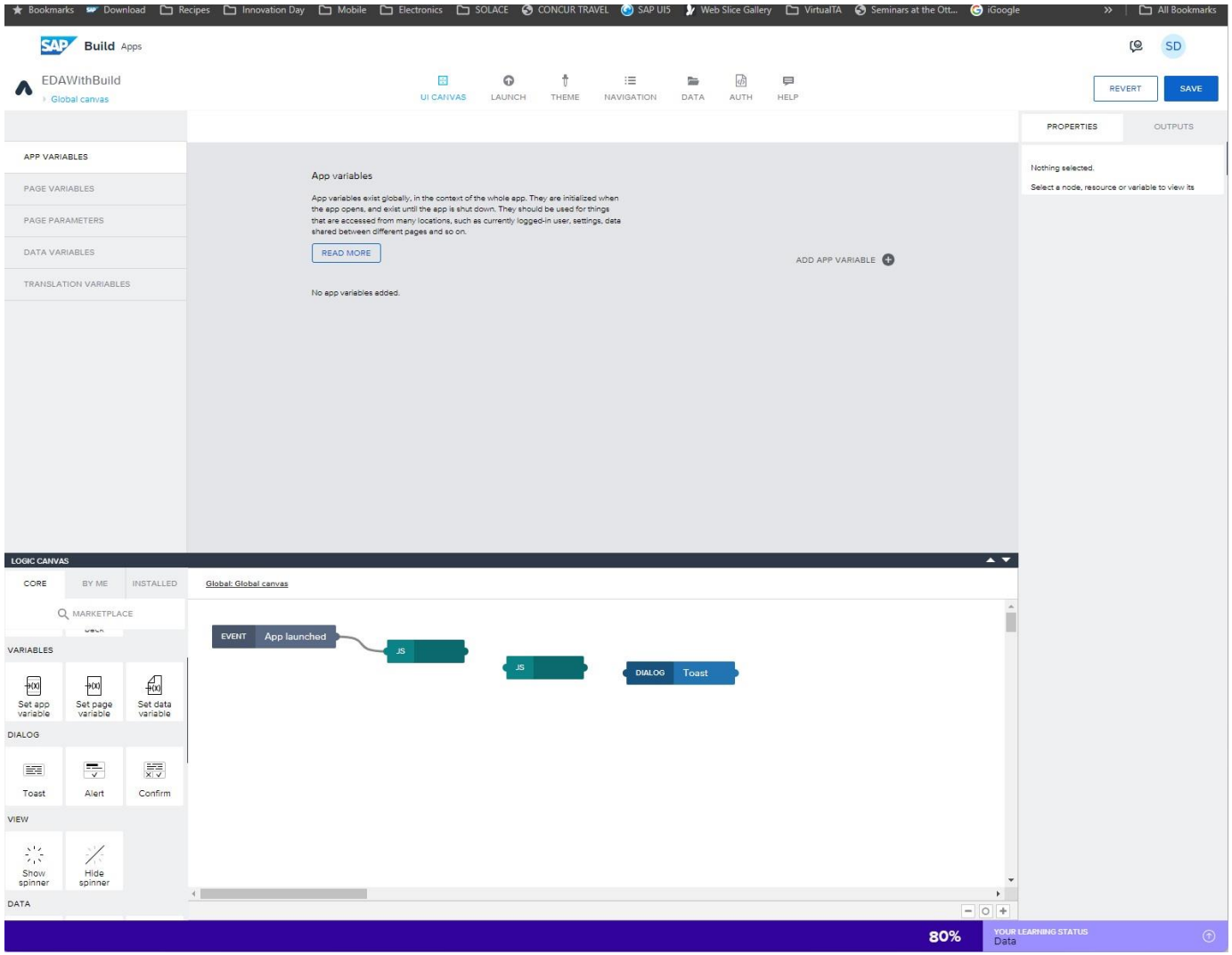



Click On Homepage to reveal the screen for all pages that are available in the application.

Then click on the Global Canvas.

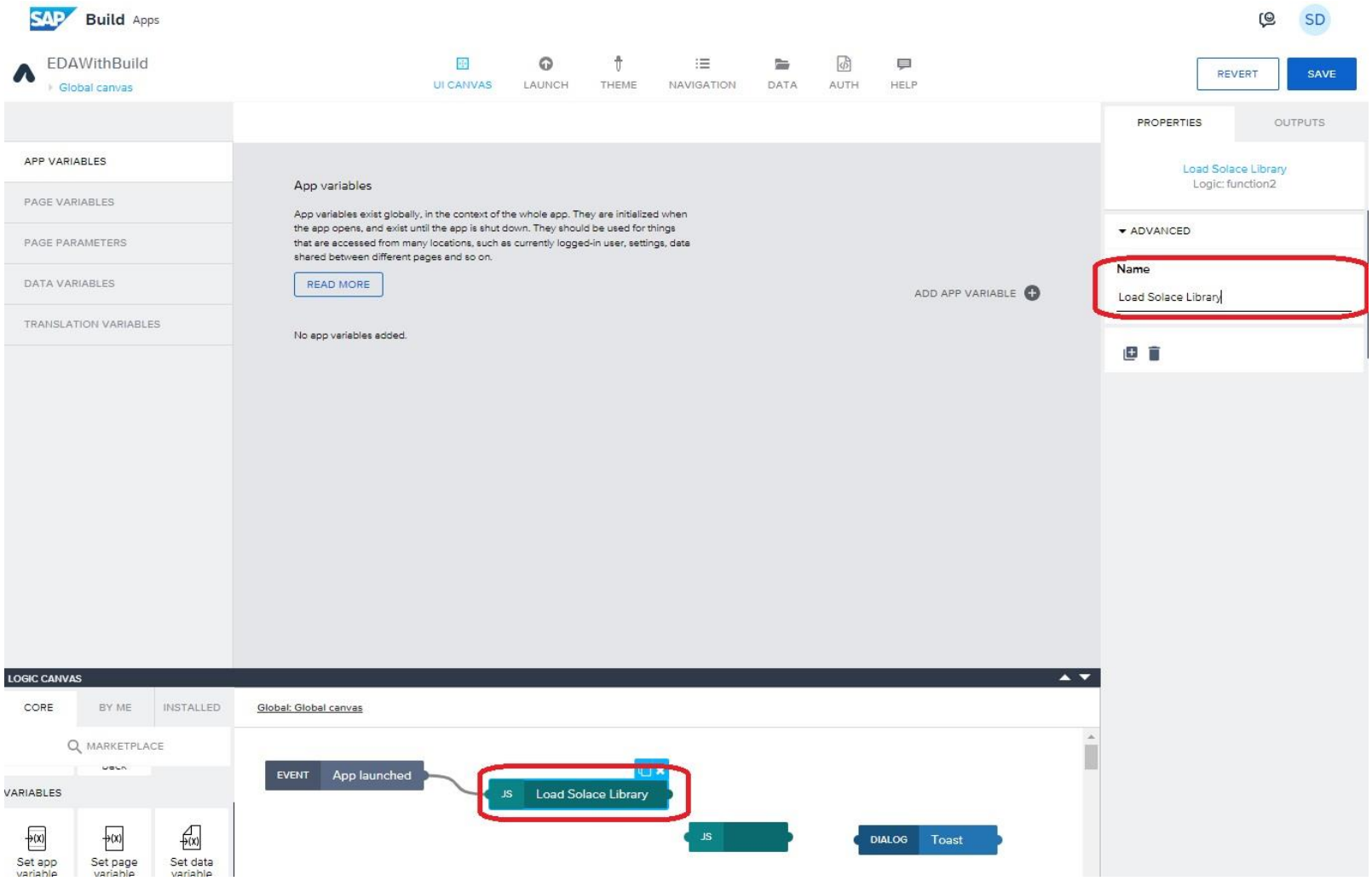


From the component palette in the lower left, click on “Core”, then let’s start by dragging 2 JavaScript widgets and 1 Toast widget  onto the canvas as show.



From here,first make sure your App Launched event is connected by dragging a line from App Launchpad Event to the first JavaScript box.

Then single click the first Javascript box to reveal the property editor on the right. Enter “Load Solace Library” for the Name (under Advanced).



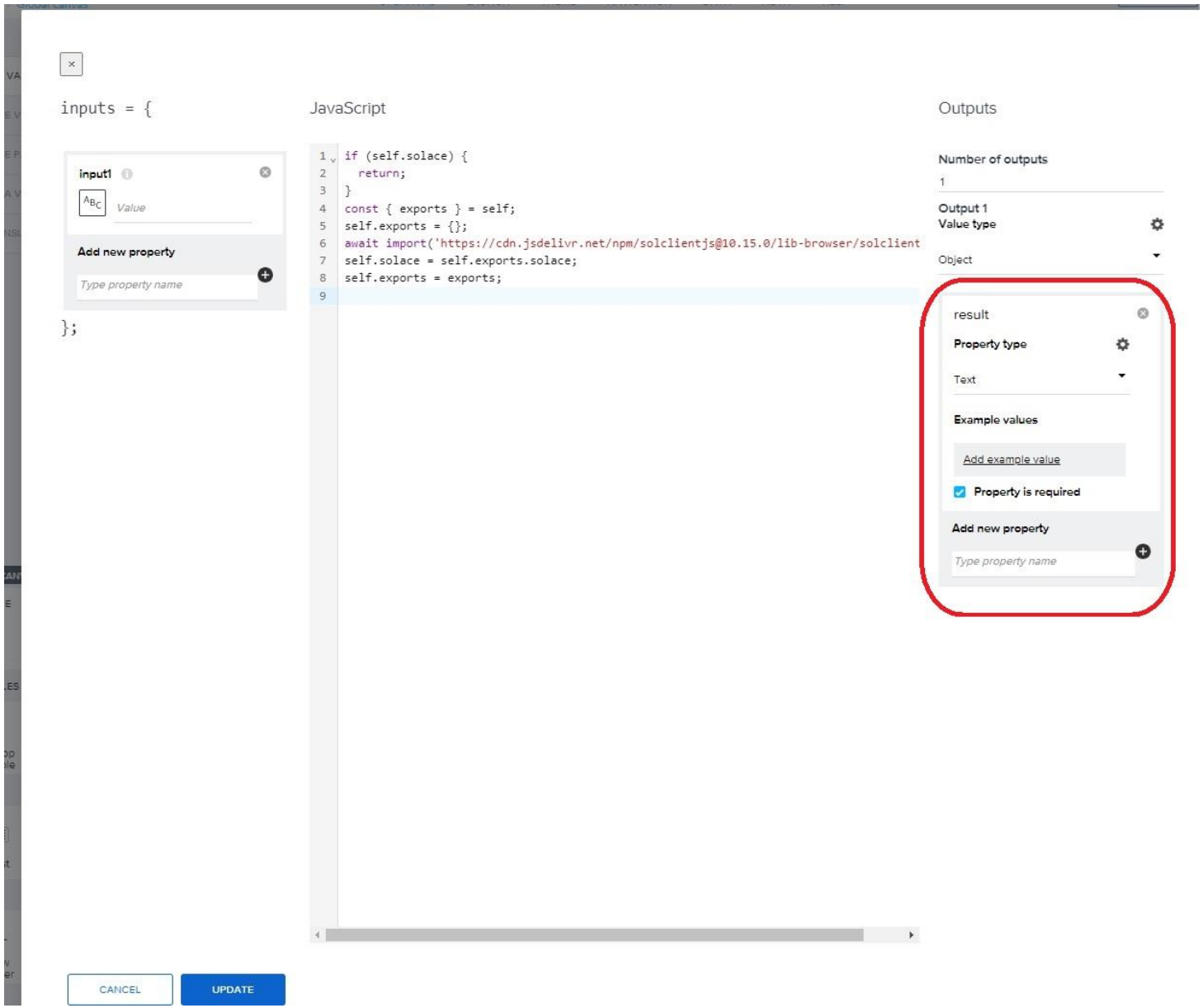

Now double click on the Load Solace Library so we can copy paste the following code.

****JAVASCRIPT BEGINS****

```
if (self.solace) {
  return;
}
const { exports } = self;
self.exports = {};
await import('https://cdn.jsdelivr.net/npm/solclientjs@10.15.0/lib-browser/solclient.js');
self.solace = self.exports.solace;
self.exports = exports;
```

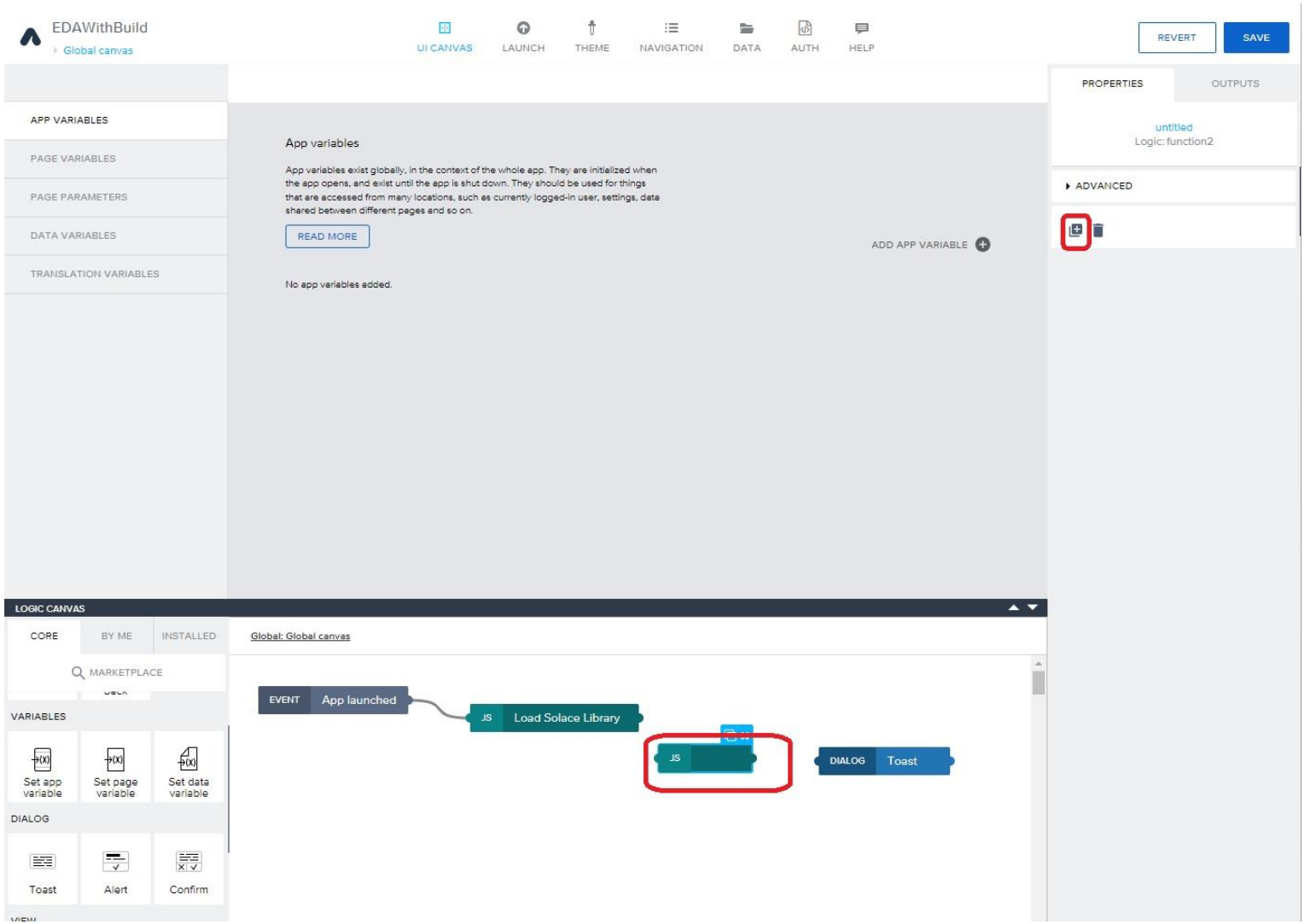
****JAVASCRIPT ENDS****

There are no inputs for this JavaScript but we do have to make one small change. Notice that an output structure has been created. You need to remove this.

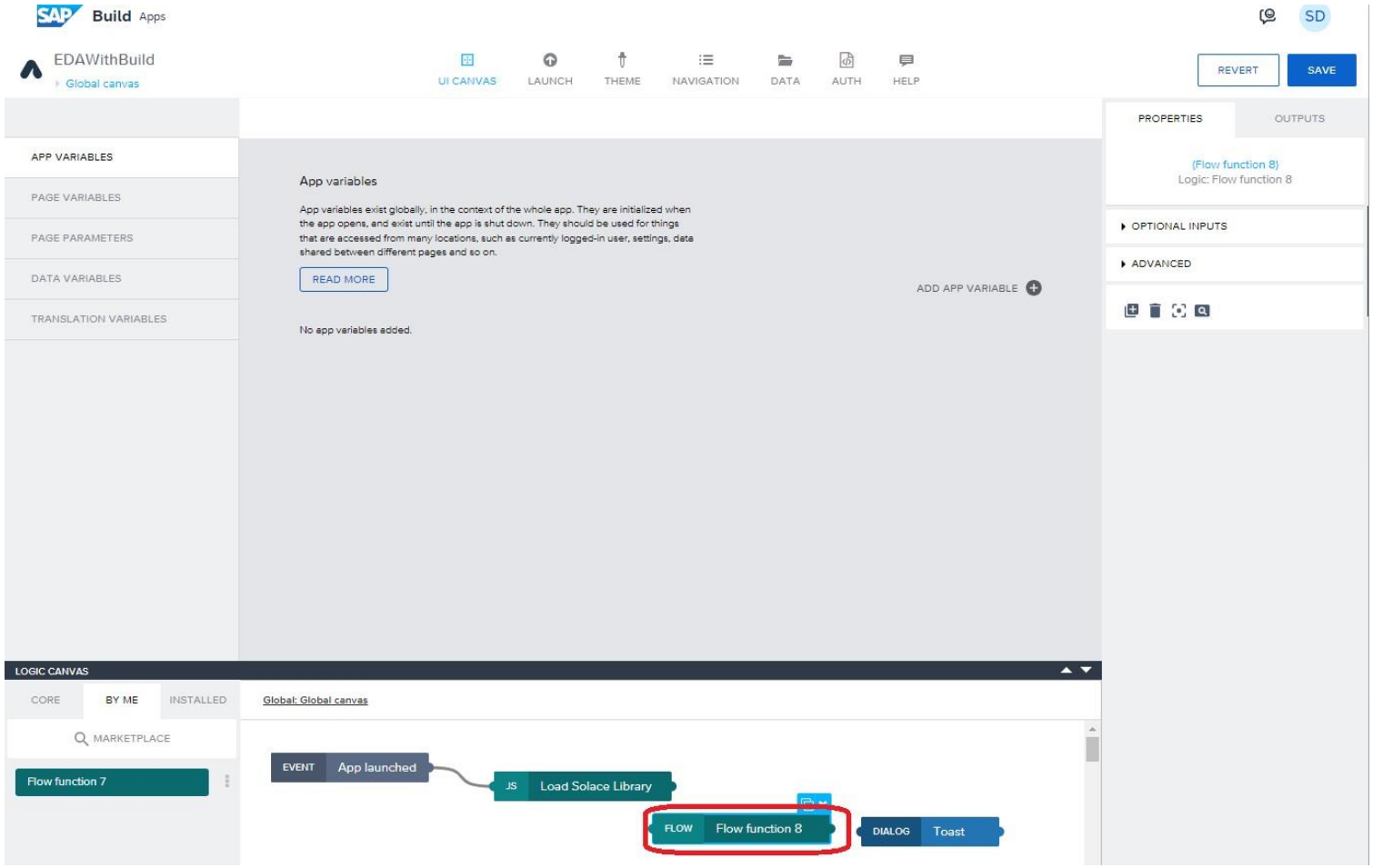

inputs = {
input1
Value
Add new property
Type property name
};
JavaScript
1 if (self.solace) {
2 return;
3 }
4 const { exports } = self;
5 self.exports = {};
6 await import('https://cdn.jsdelivr.net/npm/solclientjs@10.15.0/lib-browser/solclient
7 self.solace = self.exports.solace;
8 self.exports = exports;
9
Outputs
Number of outputs
1
Output 1
Value type
Object
result
Property type
Text
Example values
Add example value
Property is required
Add new property
Type property name
CANCEL
UPDATE

Your finished JavaScript should look like this.

Now we need to convert the remaining the JavaScript widget into a flow function. If you recall from earlier, we single click the JS widget and then on the upper right, we hit the small “+” sign. Here, we will be creating the 4 parameters we need to specify the connection to the Advanced Event Mesh. YES…we know they should not be hard coded but for this purpose it will be fine 😊

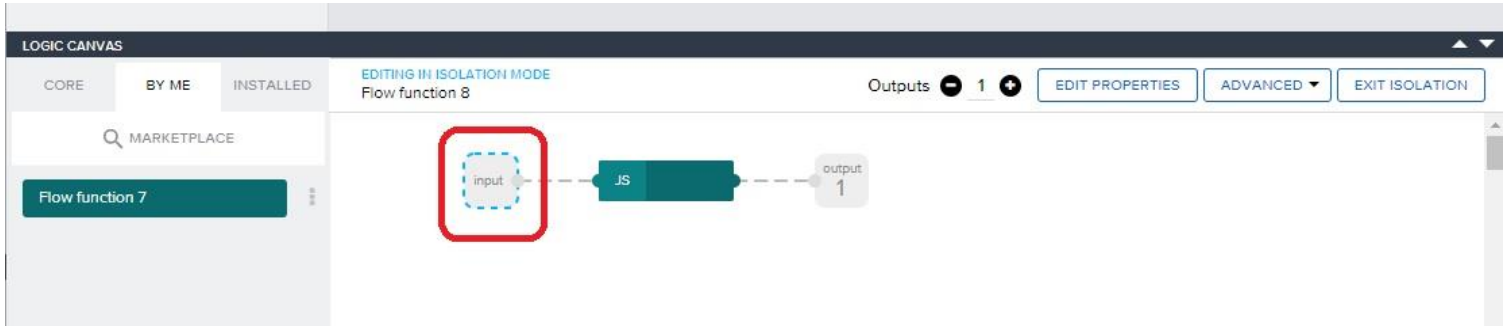


Now we will have a new Flow Function and you will double click it.

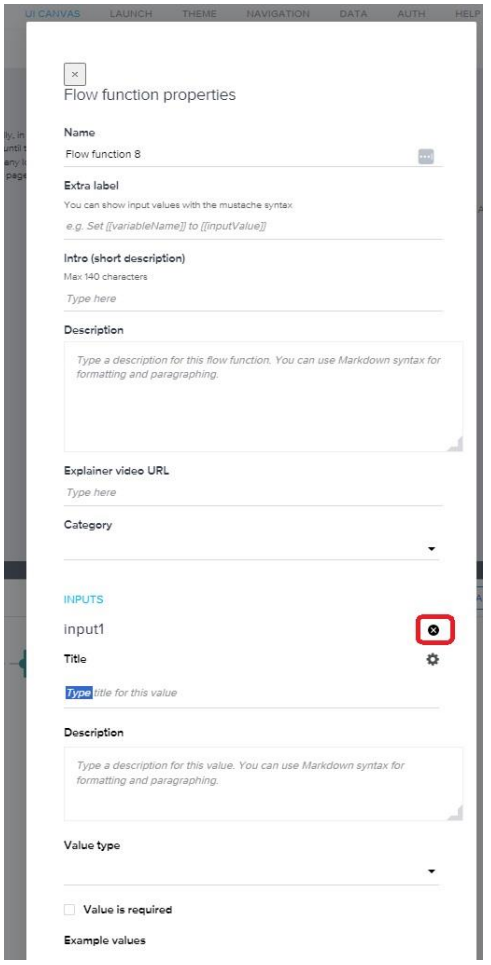



Double click on the Flow Function

Then double click on the “input”.




As before, we will start by clicking the X to remove the default parameter.



Now we will add the following parameters:

url

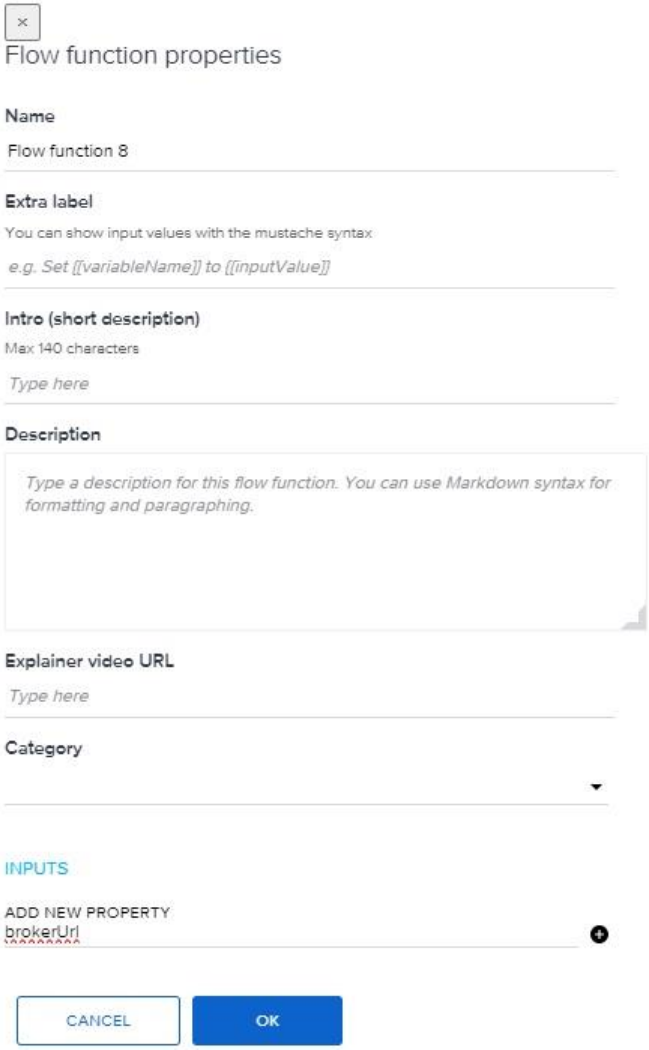
vpnName

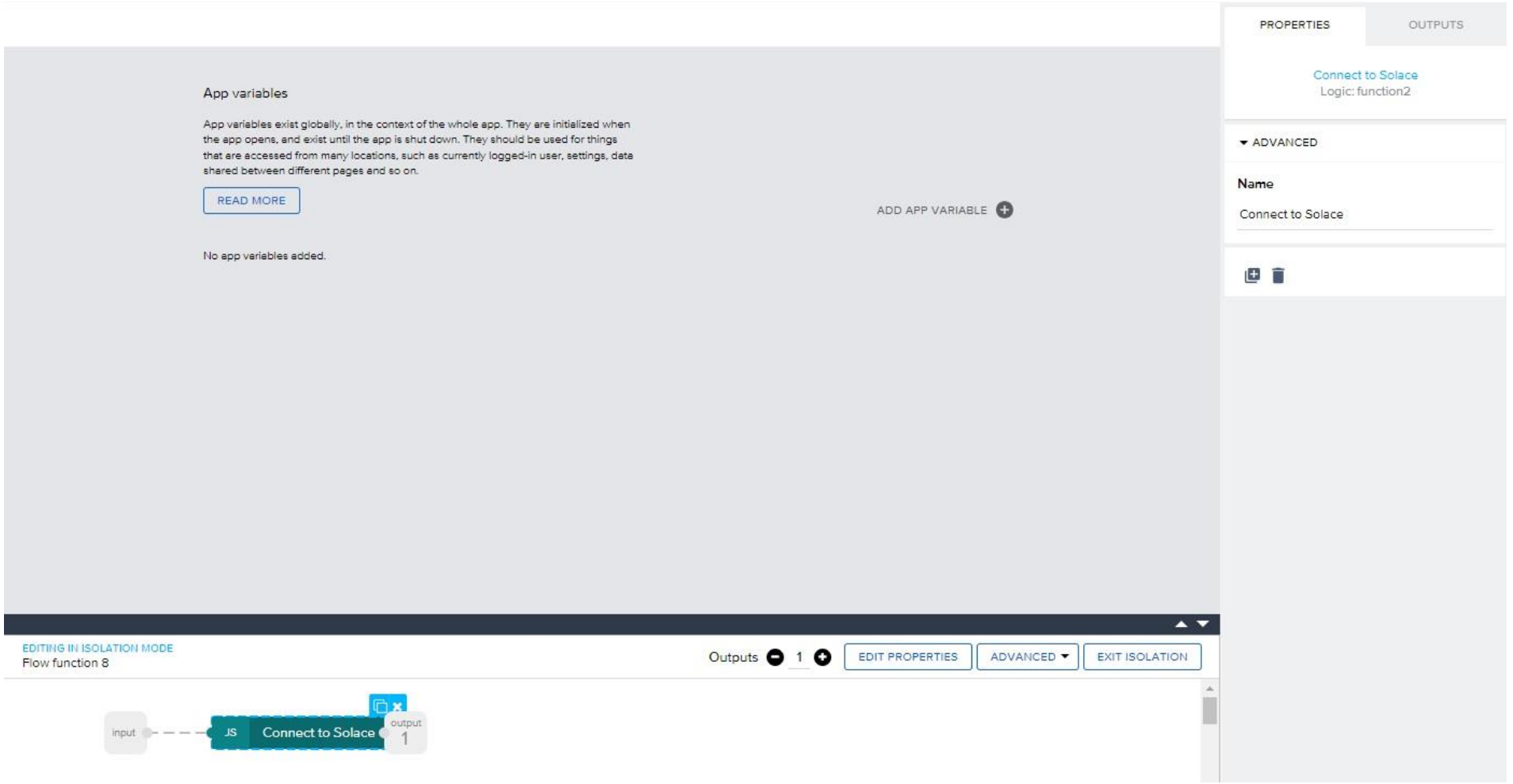
username

password

***Remember, add the parameters one at a time and each time hit the “+”, not the OK button until the end.

After all the parameters are entered and you Hit OK, you will be returned to the JS function. Single Click it to access the properties and name it “Connect to Solace”.






Now we will double click on the JavaScript Widget so we can enter the connection code.

Paste the following code into Javascript area

****JAVASCRIPT BEGINS****

```
debugger;


if (self.solace === undefined) {
  return;
}
async function createSolaceSession() {
  const factoryProps = new solace.SolclientFactoryProperties();
  factoryProps.profile = solace.SolclientFactoryProfiles.version10_5;
  solace.SolclientFactory.init(factoryProps);
  const session = solace.SolclientFactory.createSession(inputs);
  const sessionUp = new Promise(rs =>
    session.once(solace.SessionEventCode.UP_NOTICE, rs));
  //session.on(solace.SessionEventCode.MESSAGE, onMessage);
  session.connect();
  self.solace.session = session;
  await sessionUp;
  debugger;
  return {};
}
await createSolaceSession();
```

****JAVASCRIPT CODE ENDS****

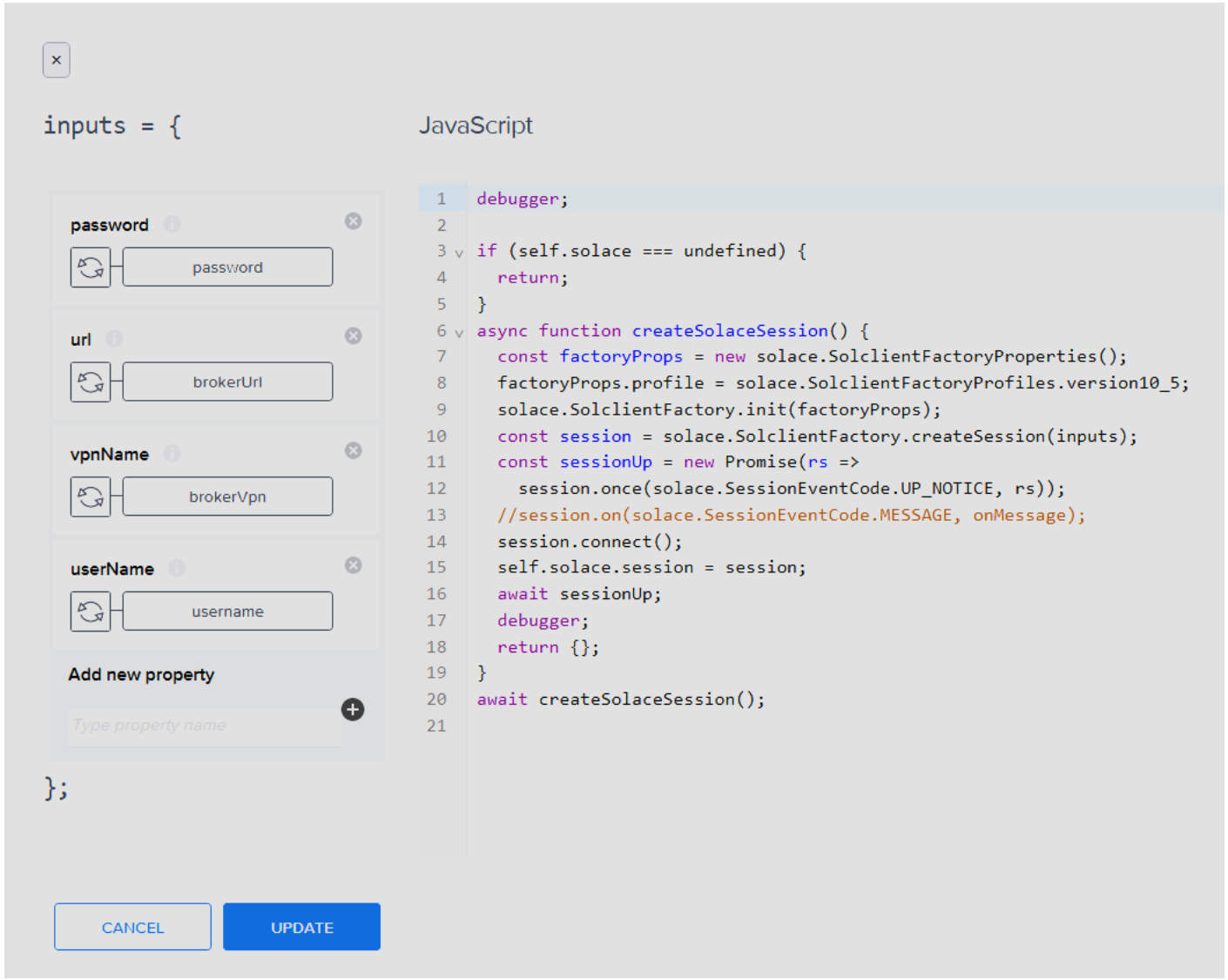

Before moving on to create the inputs for the function, you need to remove the output for “result” highlighted in red.

Now under the inputs, we will create the four properties that we created earlier for (match the case):

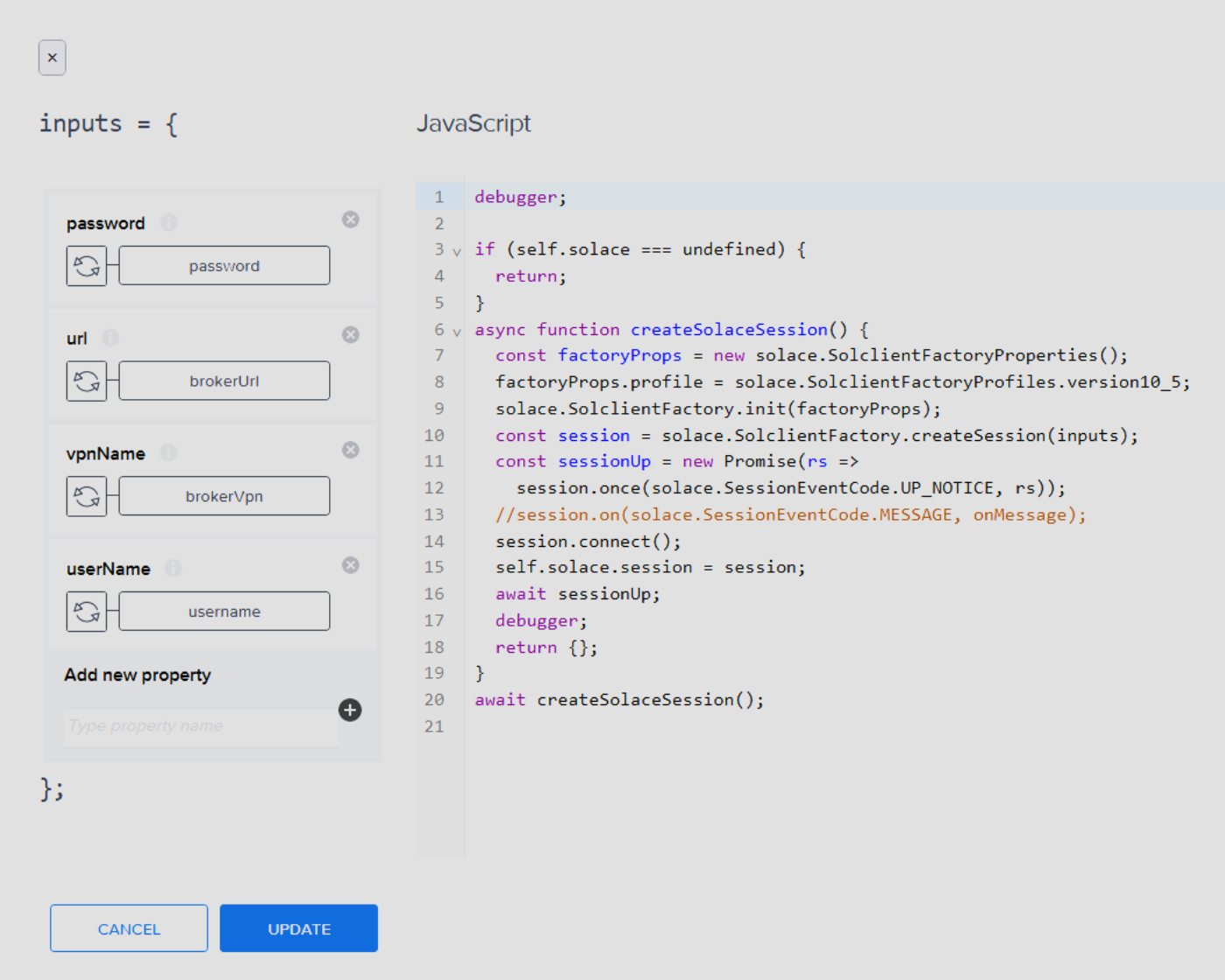
- url
- vpnName
- userName
- password

****DO NOT FORGET TO REMOVE THE DEFAULT INPUT, YOU SHOULD HAVE 4 WHEN YOU ARE DONE****

Your flow function should look like this.



But, one thing missing….your inputs are not bound to anything. You need to bind each of those to the correspond ***Flow Function Variables***….do you remember how to do it so it looks like this?



So, now that you have it bound…where are the values specified for those 4 parameters?

Remember to hit “Update” when you are done.

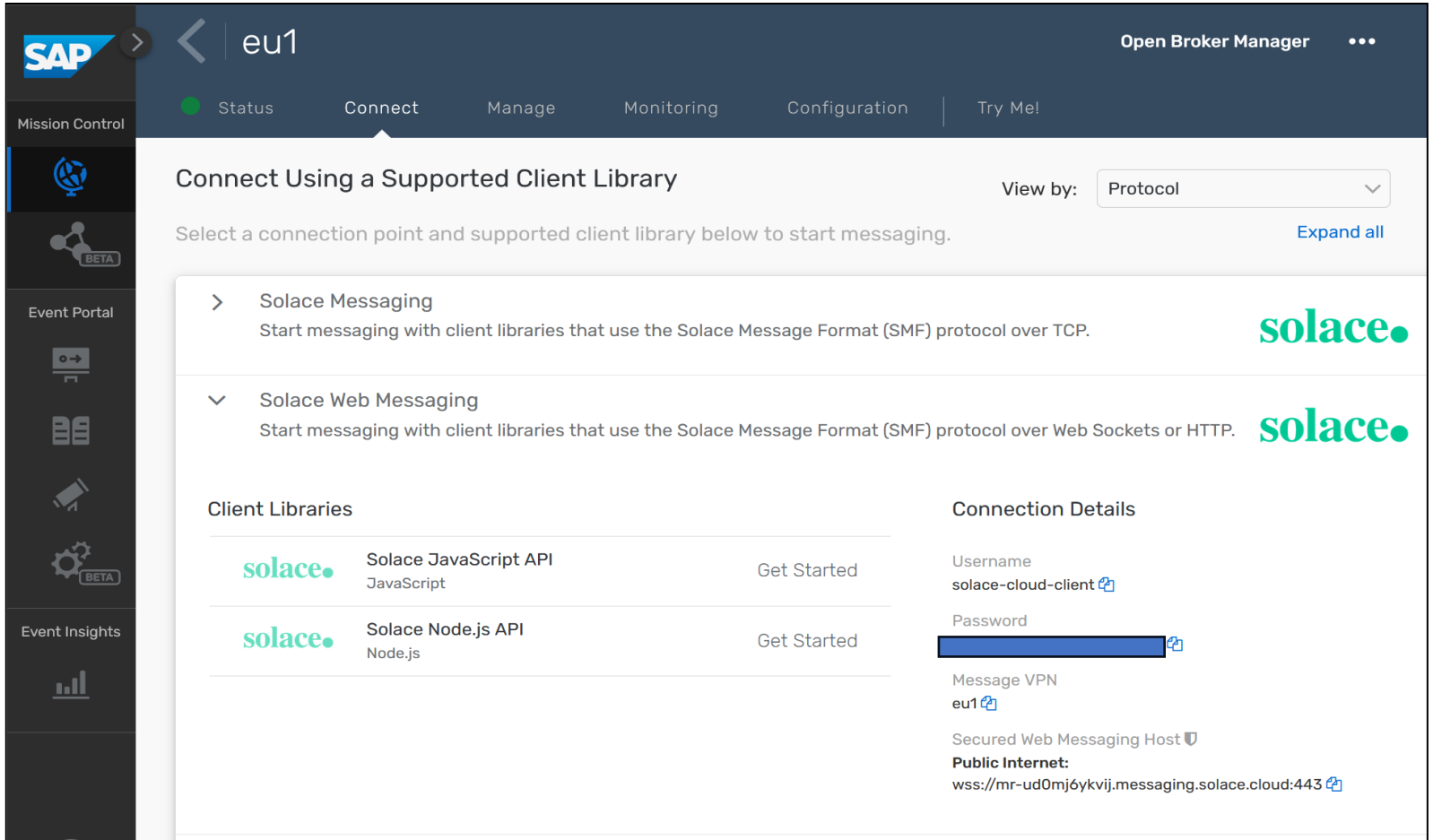
Return back to the global canvas by clicking “Exit Isolation”.

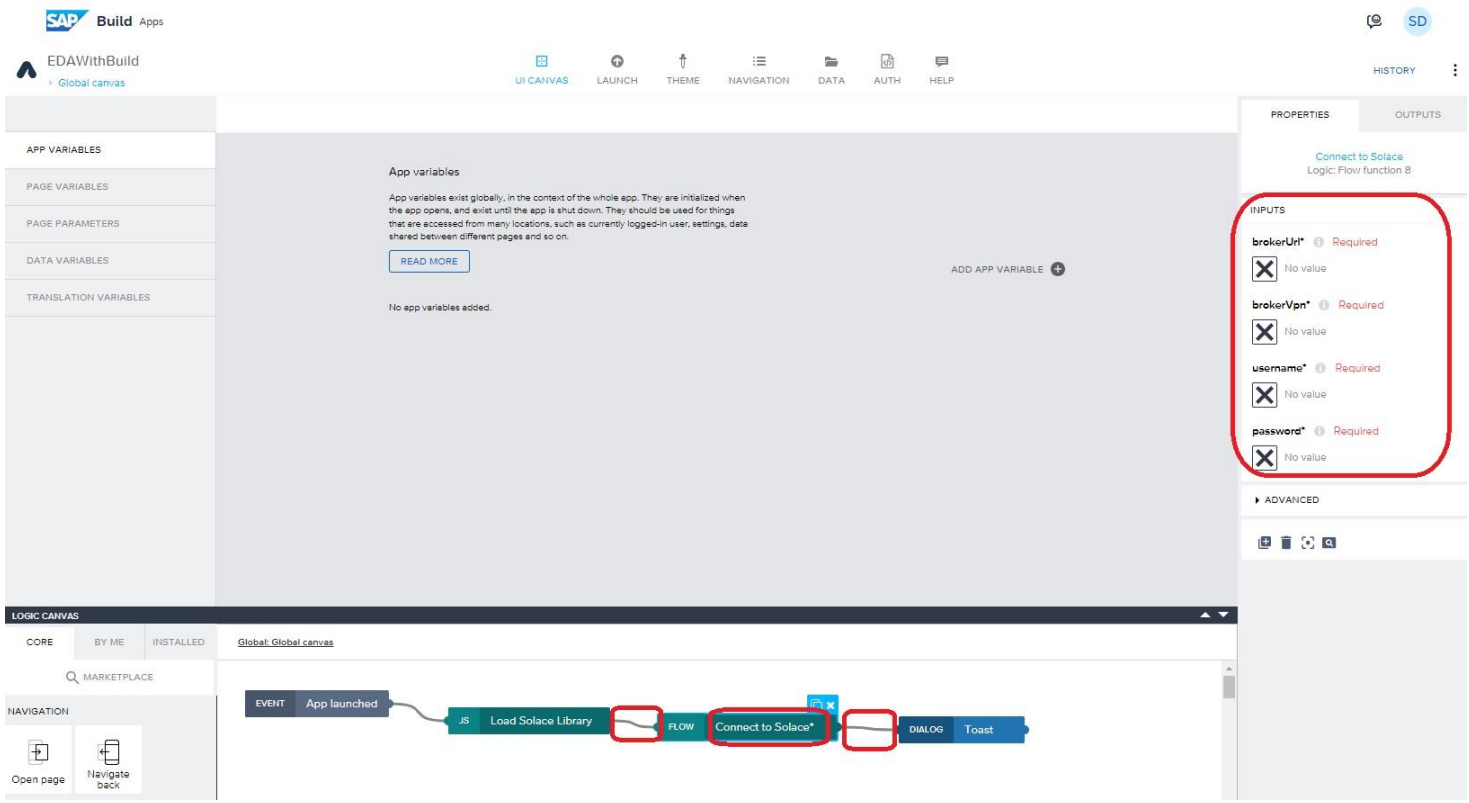
If you single click the Flow Function, the 4 parameters will appear and this is where you will enter the address of your broker. Once you enter each of those parameters, we just have a few things left to do.

First let’s make sure that the various widgets are connected like in the flow below. You will also notice that we have changed the name on the Flow Function to “Connect to Solace”.

Second, make to populate the 4 properties in the red box or you won’t connect 😊

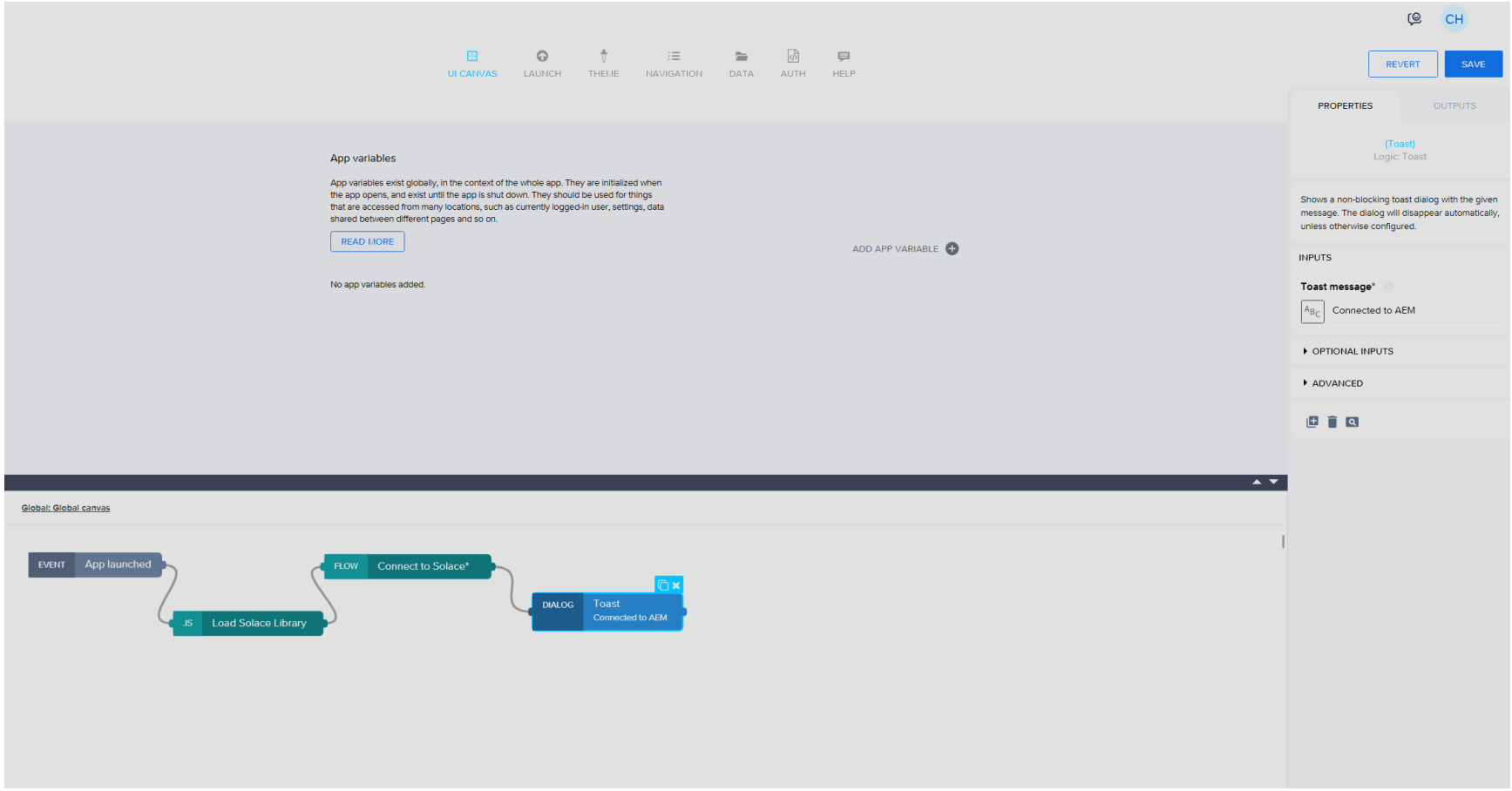
Copy these values from your “Connect” tab from your AEM broker service.




Build Apps
SD
EDAWithBuild
Global canvas
UI CANVAS
LAUNCH
THEME
NAVIGATION
DATA
AUTH
HELP
HISTORY
APP VARIABLES
PAGE VARIABLES
PAGE PARAMETERS
DATA VARIABLES
TRANSLATION VARIABLES
App variables
App variables exist globally, in the context of the whole app. They are initialized when the app opens, and exist until the app is shut down. They should be used for things that are accessed from many locations, such as currently logged-in user, settings, data shared between different pages and so on.
READ MORE
ADD APP VARIABLE
No app variables added.
PROPERTIES
OUTPUTS
Connect to Solace
Logic: Flow function 8
INPUTS
brokerUrl* Required
No value
brokerVpn* Required
No value
username* Required
No value
password* Required
No value
ADVANCED
LOGIC CANVAS
CORE
BY ME
INSTALLED
Global: Global canvas
MARKETPLACE
NAVIGATION
Open page
Navigate back
EVENT App launched
JS Load Solace Library
FLOW Connect to Solace*
DIALOG Toast

Last step, let's configure the Message Toast. Click on the Dialog Box.

On the right of the screen, enter your connection message "Connected"

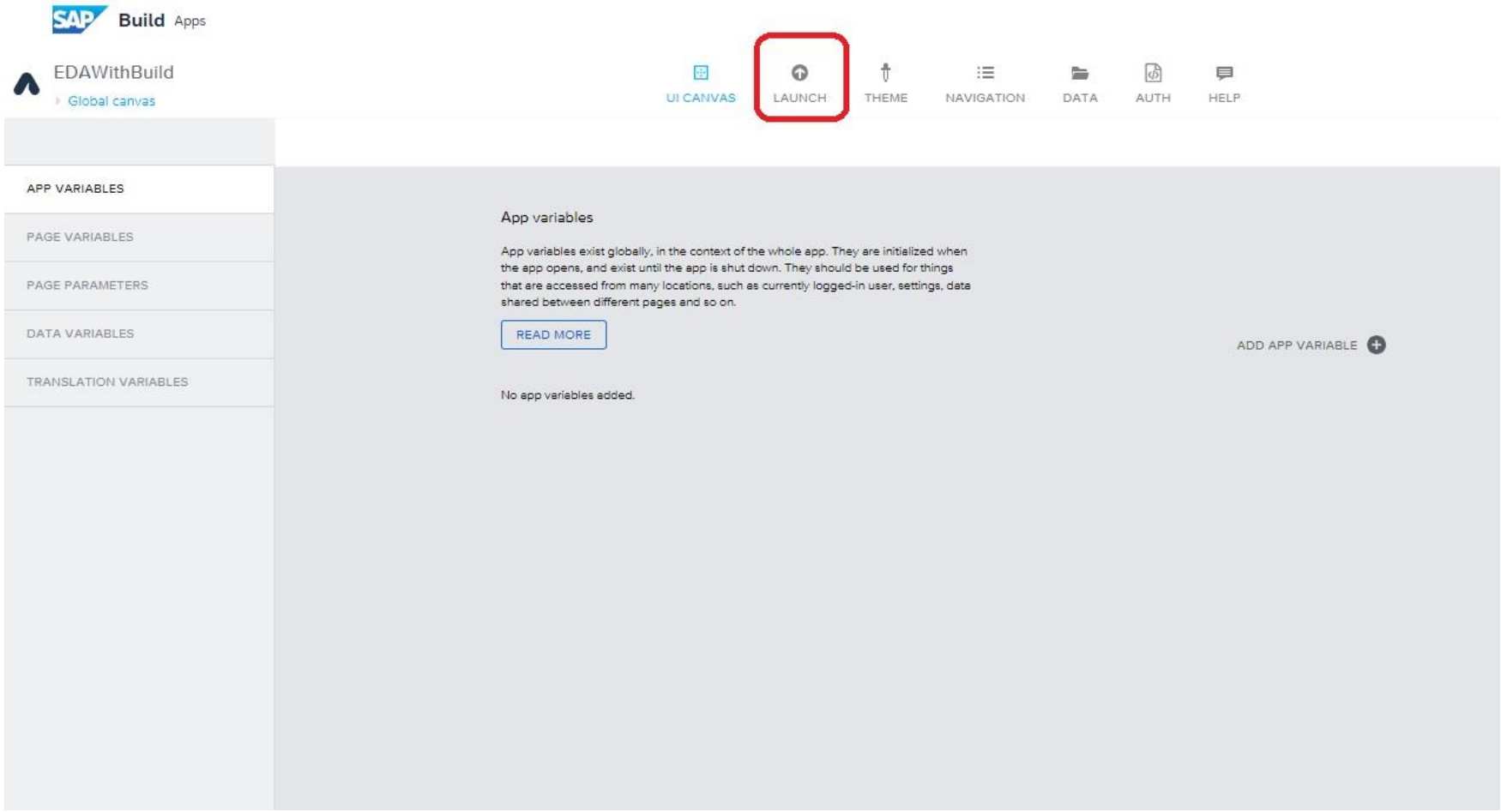



Hit Save.

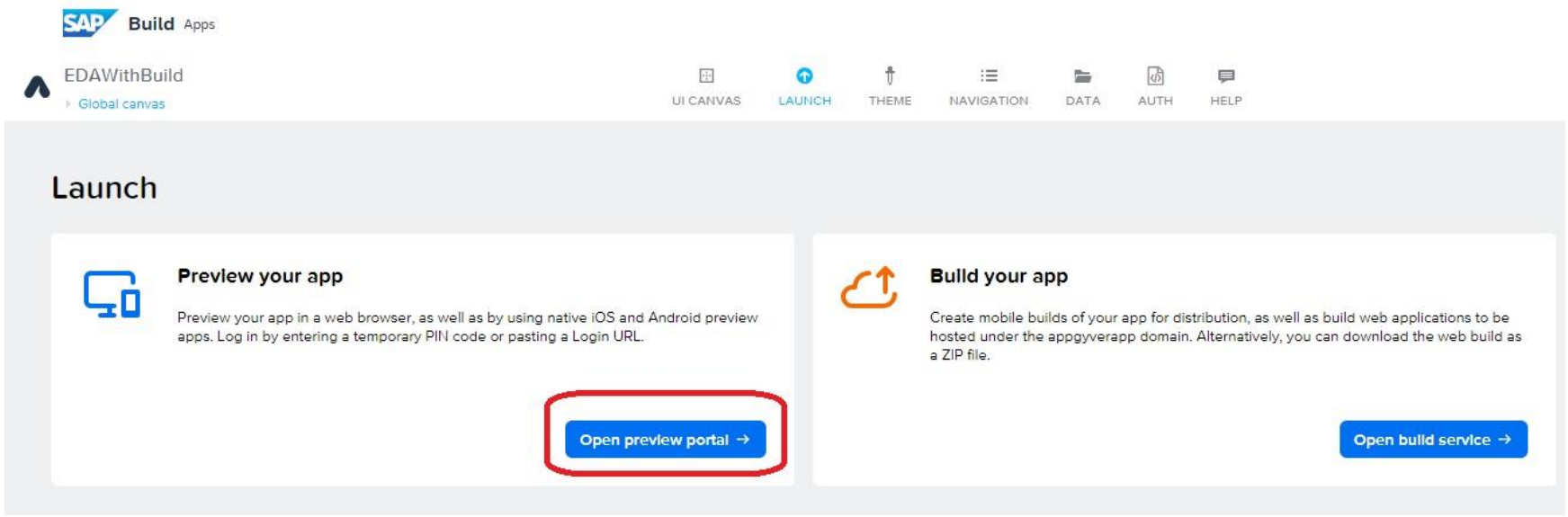
WOOHOO, you should now have an application that is ready to start sending EVENTS.

## Step 5 – Previewing the Application

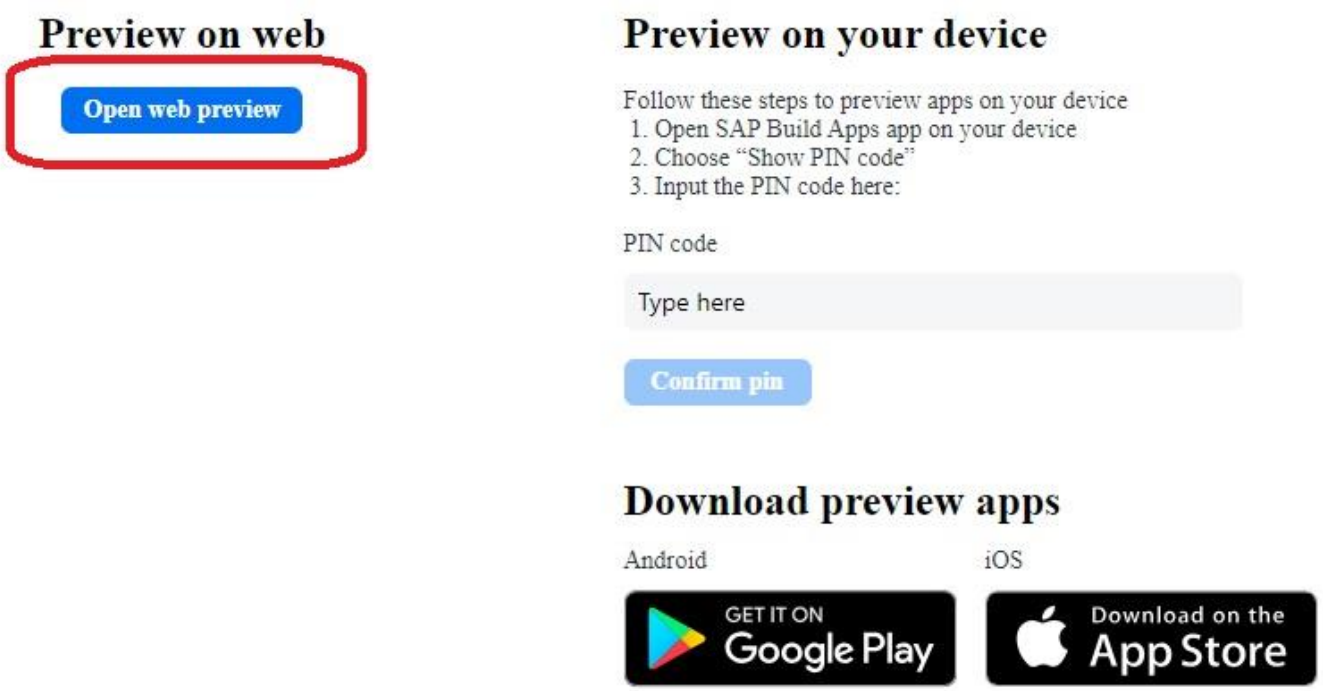
In order to pre-view our application, we need to click the “Launch” application icon at the top of the page.



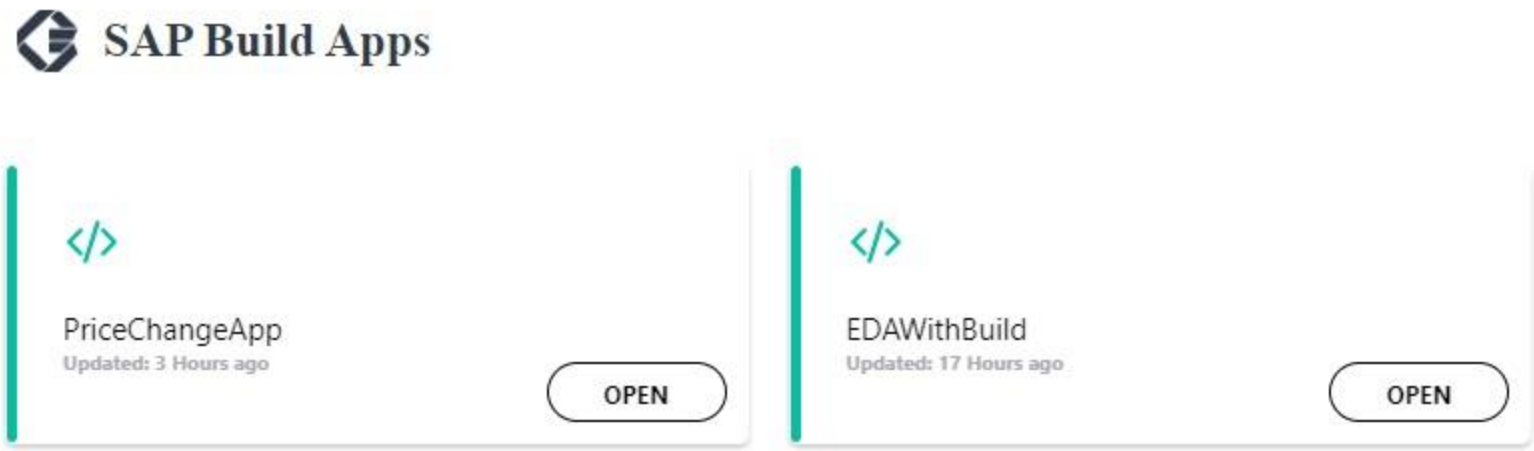

We won’t be building the application today, only Previewing the Web Version, Click on the “Open preview portal” button.




We will be using the Web Preview.




Now you will just click on the app that we have built "EDAWithBuild" to see it running.




If you've done all of your clicking properly 😊, you should see a message toast pop-up that says you are connected to Solace. This is great news, the library has been loaded.

Now you can fill out the fields and publish your first message from SAP Build Apps….nice work 😊

Home page

# Home page

EDA with Build Apps

Topic

Type here...

Region

Product ID

Product Description

Price

Type here...

Currency

Origin

Type here...

Specification

Type here...

**Publish Price Change**

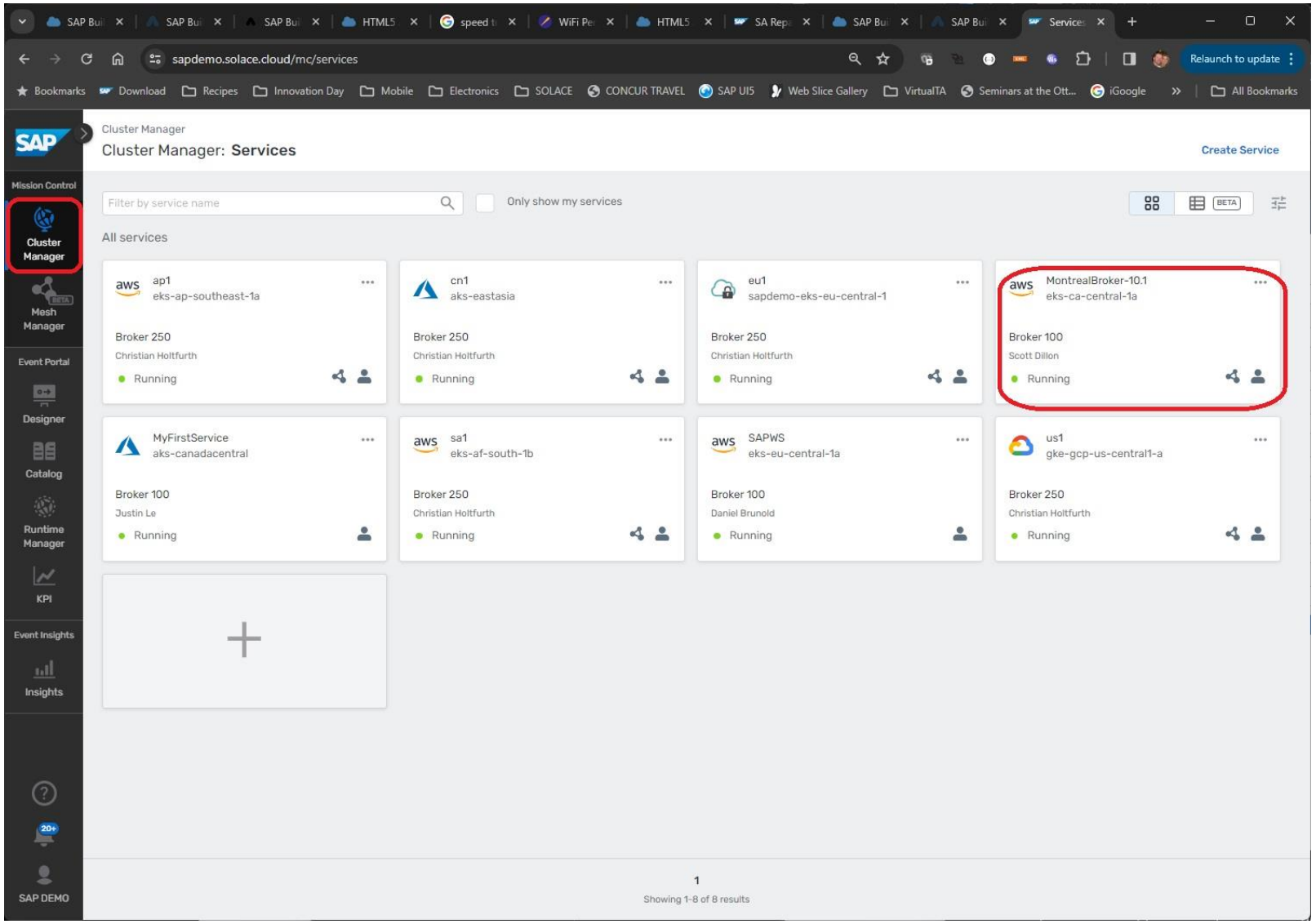
Connected to Solace

## Step 6 – Testing the Application

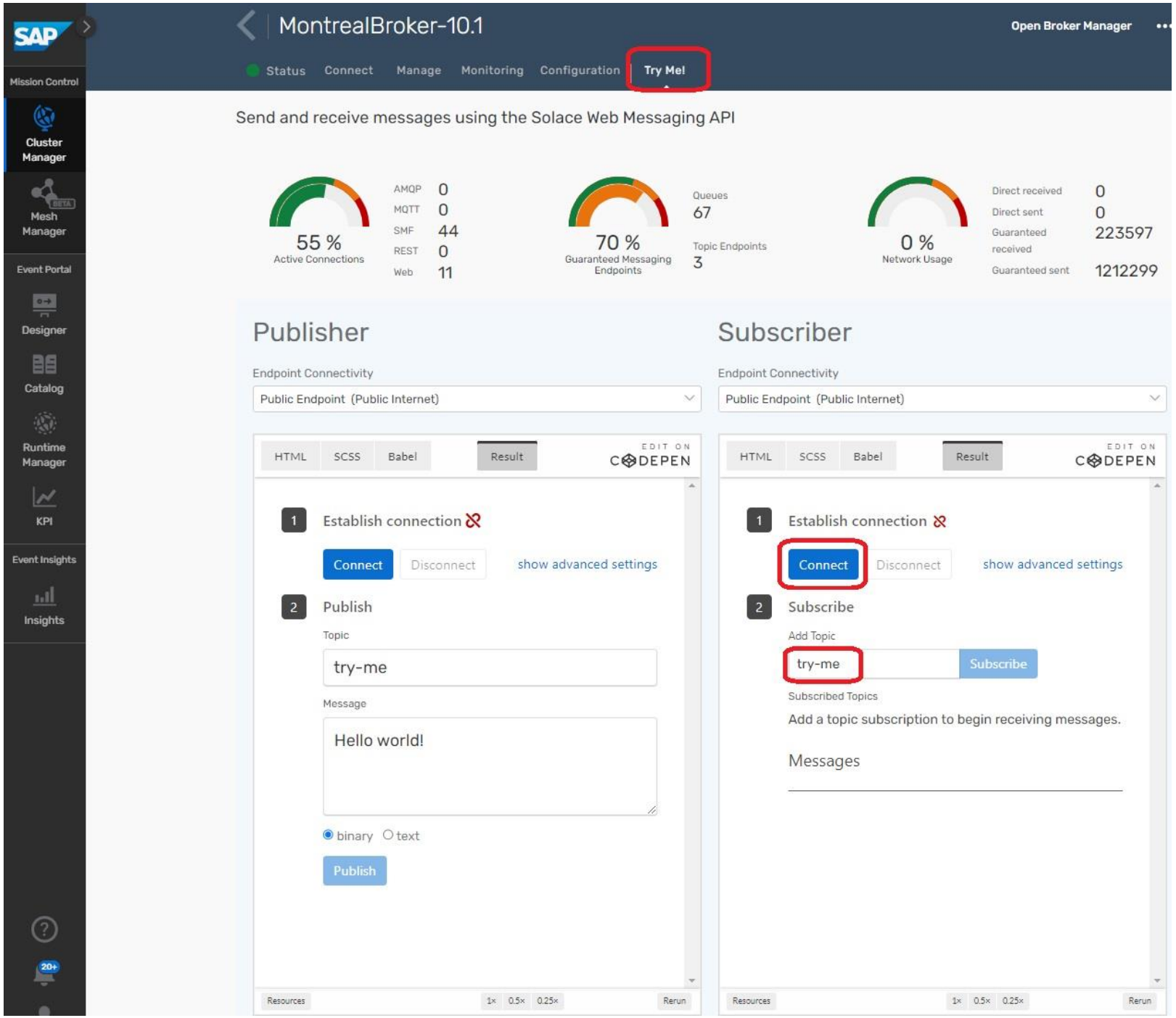
Now that you have the application up and running, we want to make sure that when you Publish the Price Change, it's being received on the Advanced Event Mesh.

From the console of the Advanced Event Mesh, you will be navigating to the "Try Me" tab.

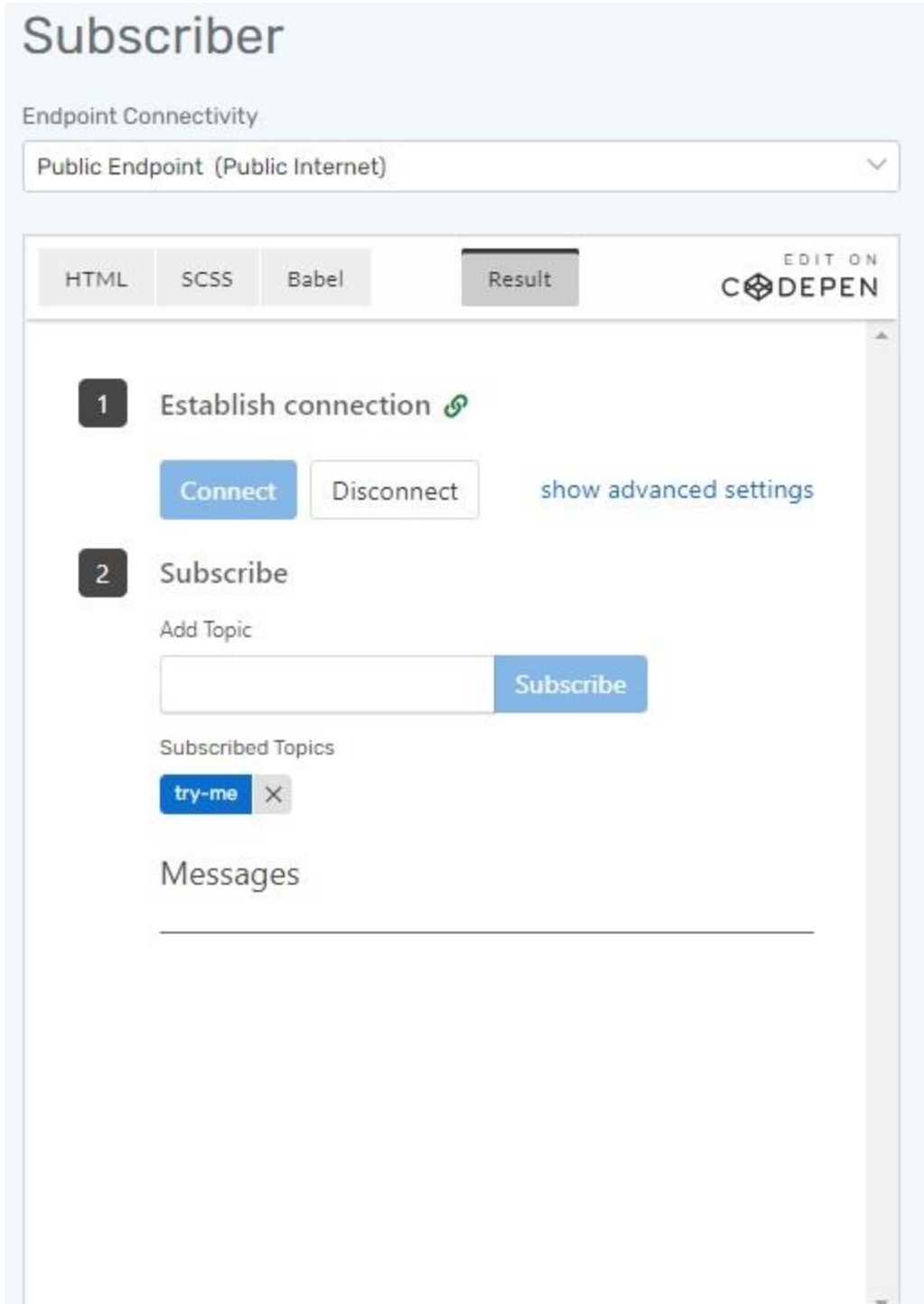
Once you open the console, click the "Cluster Manager" on the left side of the screen and then select the Broker you have created during the session. I will be selecting the Montreal broker.



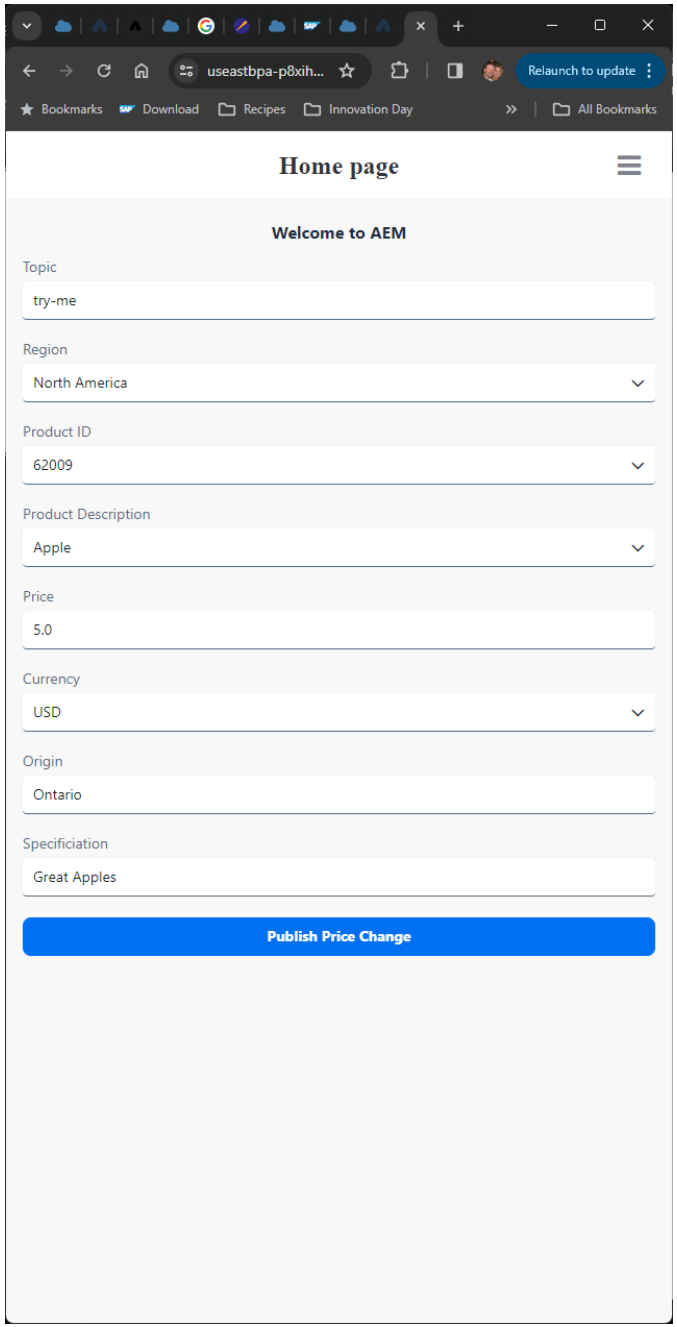

Now you will select the far right option for “Try Me” and press the “Connect” button which has been pre-populated with the credentials for your broker. We will also use the default topic, just hit the “Subscribe” button.


MontrealBroker-10.1
Open Broker Manager
Status
Connect
Manage
Monitoring
Configuration
Try Me!
Mission Control
Cluster Manager
Mesh Manager
Event Portal
Designer
Catalog
Runtime Manager
KPI
Event Insights
Insights
Send and receive messages using the Solace Web Messaging API
55 %
Active Connections
AMQP 0
MQTT 0
SMF 44
REST 0
Web 11
70 %
Guaranteed Messaging Endpoints
Queues
67
Topic Endpoints
3
0 %
Network Usage
Direct received 0
Direct sent 0
Guaranteed received 223597
Guaranteed sent 1212299
Publisher
Endpoint Connectivity
Public Endpoint (Public Internet)
HTML
SCSS
Babel
Result
EDIT ON CODEPEN
1 Establish connection
Connect
Disconnect
show advanced settings
2 Publish
Topic
try-me
Message
Hello world!
binary
text
Publish
Resources
1×
0.5×
0.25×
Rerun
Subscriber
Endpoint Connectivity
Public Endpoint (Public Internet)
HTML
SCSS
Babel
Result
EDIT ON CODEPEN
1 Establish connection
Connect
Disconnect
show advanced settings
2 Subscribe
Add Topic
try-me
Subscribe
Subscribed Topics
Add a topic subscription to begin receiving messages.
Messages
Resources
1×
0.5×
0.25×
Rerun

Once you have successfully connected, you will see that both the Connect Button and the Subscribe Button are greyed out and we have 1 Topic listed for the “Subscribed Topics”. Now this web based client is listening for Events that have the “try-me” topic.


Subscriber
Endpoint Connectivity
Public Endpoint (Public Internet)
HTML
SCSS
Babel
Result
EDIT ON
CODEPEN
1
Establish connection
Connect
Disconnect
show advanced settings
2
Subscribe
Add Topic
Subscribe
Subscribed Topics
try-me
Messages

So, let’s try it with the app. You can see we have defaulted the Topic to “try-me”. For this first test, just leave the default topic in place and use the drop down fields to populate where necessary like below and hit the “Publish Price Change”.


useastbpa-p8xih...
Relaunch to update
Bookmarks
Download
Recipes
Innovation Day
All Bookmarks
Home page
Welcome to AEM
Topic
try-me
Region
North America
Product ID
62009
Product Description
Apple
Price
5.0
Currency
USD
Origin
Ontario
Specificiation
Great Apples
Publish Price Change

If your application is configured properly, you have have just successfully published to the AEM service and also subscribed from it. Congratulations.