Calculated Prices App By Holest Engineering

1. Introduction

“Calculated Prices” is Shopify app that lets you:

- *Add additional product price modifying parameters.*
  You can make these additional parameters conditional and define their display order.
- *You can define price modifying formulas that are product specific or shop-wide.*
  (inside formula you can use product or variant properties, additional parameters values, customer tags, client country, date and time as conditions).
- *You can add conditional file uploads for product orders.*
- *You can add conditional text inputs for product orders.*

“Calculated Prices” APP automatically embeds additional product elements and modifies prices as you define. APP will manage its separate draft order that has specified modified prices depending on user input.

2. General Configuration

When you enter app from admin you will see few parameters you need to set. Here is an explanation of each of them:

App-enabled in front-end: Enables you to deactivate/activate app for front-end shop on need.

The 'Add to Cart' action loads page on my current theme (check this if statement is true): rarely on some themes, the *add to cart using ajax* is not used. We need to know exactly if this is the case to handle things properly. If you change theme and this statement changes for your case you need to update this setting.

Max. a number of price modifying params per product: Here you need to enter the maximal number of price parameters you can possibly have for some product. Shop environment will define values as Parm1, Parm2, Parm3 ...

+ **Apply formulas on parameter values:** this is to set whether you want global and product formula to effect price of price parameter or not

Max. a number of free text inputs per product: Here you need to enter a maximal number of text inputs you could possibly have for some product. Shop environment will define values as Input1, Input2, Input3 ...

Max. a number of file uploads per product: Here you need to enter a maximal number of uploads you could possibly have for some product. Shop environment will define values as Upload1, Upload2, Upload3 ...

This parameter will tell EL SBPM app how many parameter/input/upload columns it needs to show for you.
Note that no value is removed until you explicitly do so. For example, if you have 5 parameters and set value for some product in column 5 and then change this parameter to 3 value for column 5 will stay - it will just not be shown or used.

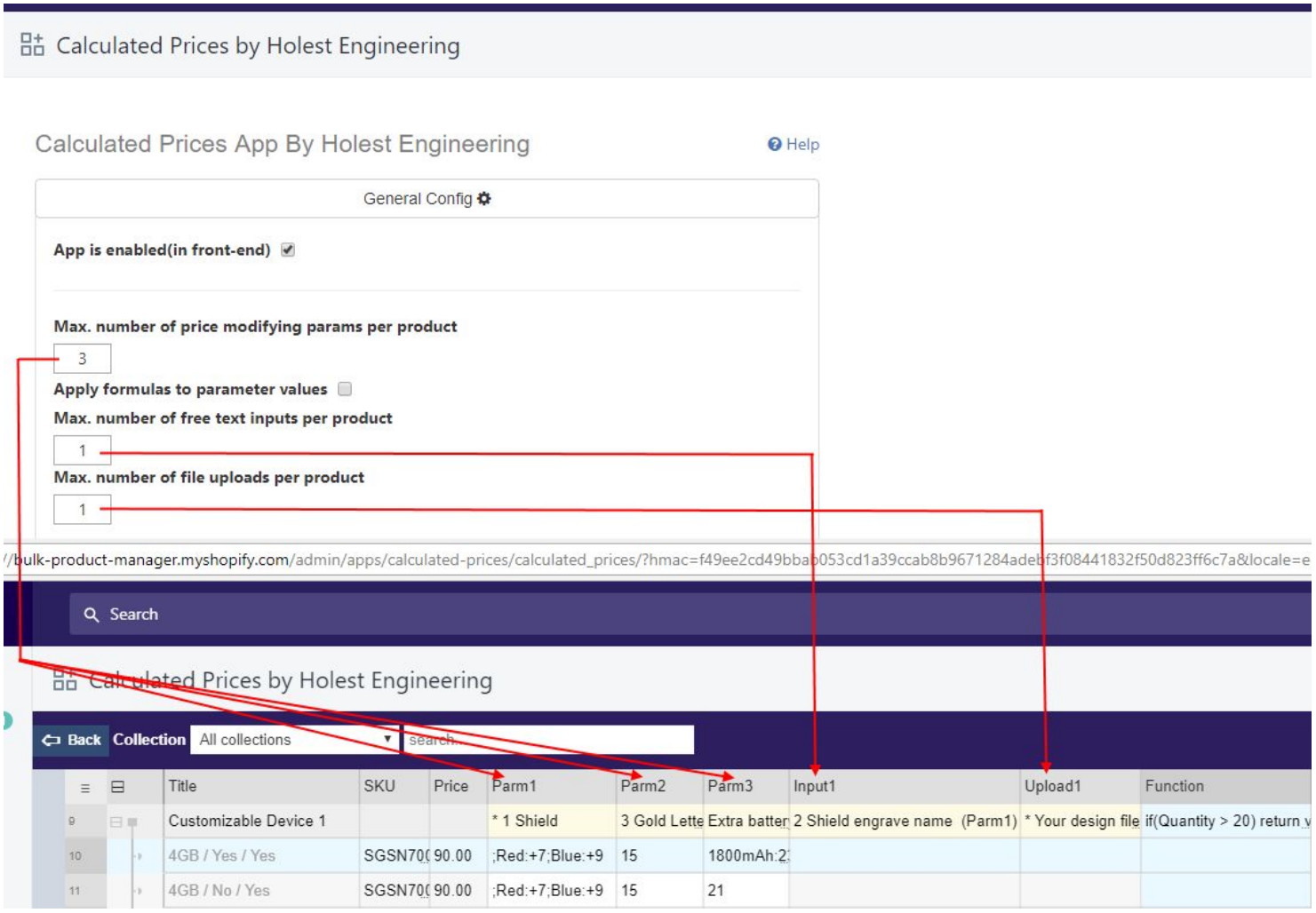


IMAGE1

Google drive settings: this app uses Google drive for file uploads. You need to configure this if you use uploads. Details about settings this up can be found in separate instructions you can see when you click on *"How do I configure this?"* button.

Shop-wide(global) formula: This is the place where you enter the shop-wide global formula. The global formula is related to the whole shop and does not consider particular products. We will explain it details under section (4.)

INSTALLATION NOTE: After you save the configuration for the first time go to your frontend and refresh page few times in 10s periods to allow the app to embed into your shop properly.

3. Managing Calculated Prices

3.1 Additional parameter fields

103997	Customizable Device 1	1 Engraving	Gold Letters (Parm1)	Wooden box
449511	14GB / No / Yes	+13.00	12	13
449511	16GB / Yes / ?	+14.00	11	12
449511	5GB / Yes / Yes	+15.00	10	11
449511	y / A76 / ?	+16.00	0	-10
449511	X / a / ?	+17.00		9
449511	4GB / No / Yes	Red:+6;Blue:+8	6	7
449511	4GB / Yes / Yes	+18.00	7	8

IMAGE2

DEFINITION SYNTAX:
[*] [order num] [label text] [(condition)] [@html_attribute1 = “value” @html_attribute2 = “value” … @html_attributeN = “value”]

or

[order num] [*] [label text] [(condition)] [@html_attribute1 = “value” @html_attribute2 = “value” … @html_attributeN = “value”]

*	will make field required
order num	number that defines display placement order of the field
label text	field label text
(condition)	condition to satisfy in order of showing field
@html_attributeX = “value”	Attributes that should be added to input element (prefixed with @) like @style=”max-width:80px;color:green;” will add style attribute to input element. If you want to use common input instead of textarea you can use @type=”text”. @type=”number”, @type=”password”, @type=”date” are also valid. If you want to change caption of field on product page (to be different than filed name) you can use caption attribute. For checkbox parameters you can also specify place of price in Caption with wildcard $price like: @caption=“Checking this checkbox will add $price per product”

3.1.1 Price modifying parameters

Price modifying parameters are price modifications that apply when a value is defined for particular variant and when conditions for this parameter to take effect are met. They only exist on the product page (as they are a property of particular product).

Price modifying parameters are defined like this:

- In product cell enter you text label from parameter (how you want it to appear), example “Engraving”
- in a variant, cells enter modification values

Values can be positive and negative. Value 0 means that parameter exists but that costs nothing. An empty value means that for that variant parameter does not exist.

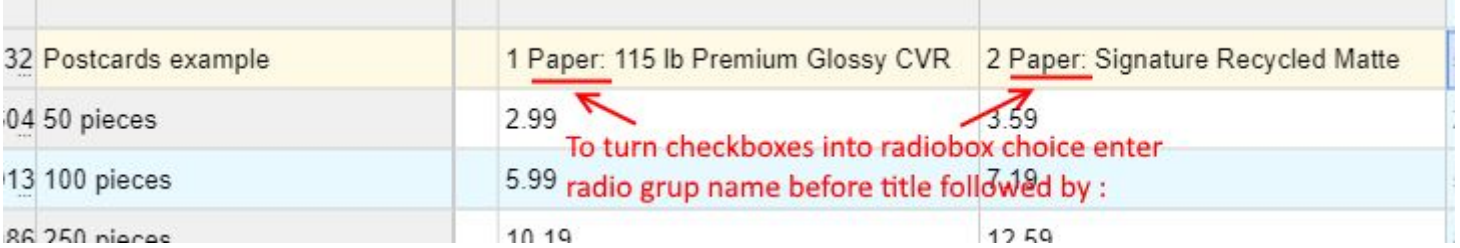
There are two types of parameters single-option and multi-option ones.
Single option parameters turn into checkboxes on a product page. Logic is simple. If they apply value you enter in its cell is added to product price. Example if defined as "+18.00", 18 is added to product value.

Multi-option parameters have choices that may have different values. They turn into select dropdowns on a product page. They are defined as semi-colon separated name:value list: name1:add_price1;name2:add_price2….

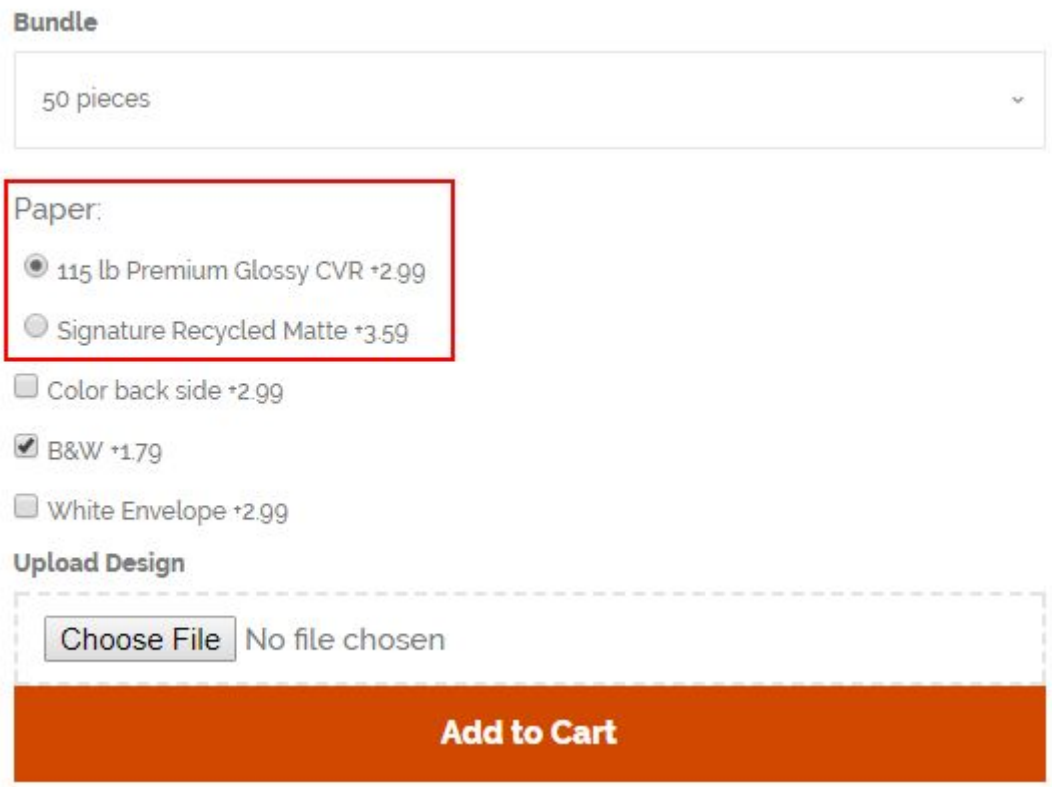
You can see them on above picture for *4GB / No / Yes* variant for Engraving parameter: *Red:+6; Blue:+8*

*Turning checkboxes into radio box choice

If you want to turn multiple fields that render into checkboxes then enter radio group name before label title followed by: See following images:



This will result in this:



3.2 Product order uploads and text inputs

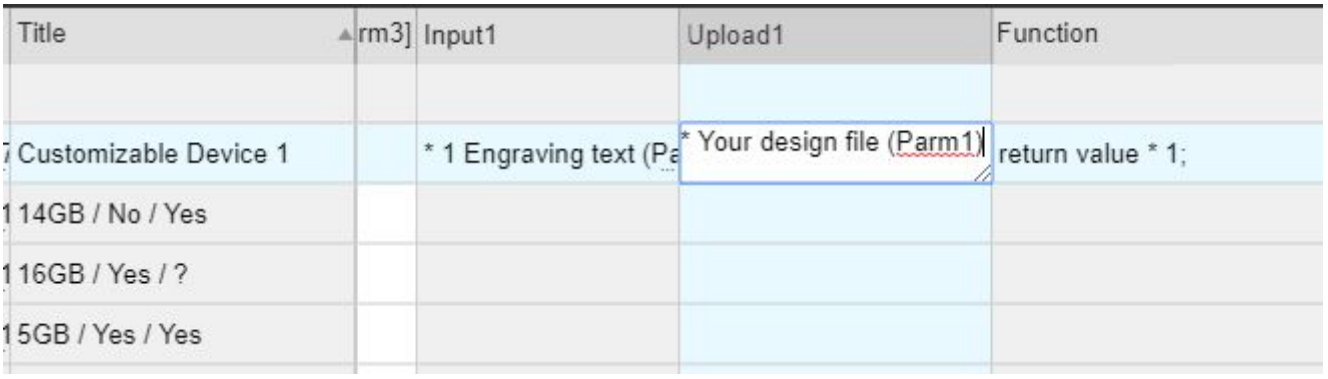


IMAGE3

DEFINITION SYNTAX:
[*] [order num] [label text] [(condition)] [@html_attribute1 = “value” @html_attribute2 = “value” … @html_attributeN = “value”]

or

[order num] [*] [label text] [(condition)] [@html_attribute1 = “value” @html_attribute2 = “value” … @html_attributeN = “value”]

*	will make field required
order num	number that defines display placement order of the field
label text	field label text
(condition)	condition to satisfy in order of showing field

@html_attributeX = “value”	Attributes that should be added to input element (prefixed with @) like @style=”max-width:80px;color:green;” will add style attribute to input element. If you want to use common input instead of textarea you can use @type=”text”. @type=”number”, @type=”password”, @type=”date” are also valid. If you want to change caption of field on product page (to be different than filed name) you can use caption attribute. For checkbox parameters you can also specify place of price in Caption with wildcard $price like: @caption=“Checking this checkbox will add $price per product”

It is common that when you have additional parameters for a product you have some need to have some custom text or file upload. If you want to activate input/file upload for some product then you enter field definition in appropriate product cell - same as for price parameter.

On product page inputs will become variables Input1, Input2…

This values represent HTML elements. If you need text form input 1 in condition or formula you can get is as *Input1.value* . If you need inputted text length then *Input1.value.length*

On product page uploads will become variables Upload1, Upload2…

This values represent HTML elements. If you need to check if Uplaod1 has filed in condition or formula you can use *!!Upload1.files[0]* - will give true if user selected upload file for Upload1 otherwise false.

Uploaded files are stored in a moment of "product add to cart" to your Google drive.

3.3 Field display conditions, required yes or no

If a parameter has defined value for particular variant it is displayed by default. If you want to hide/show it based on other parameters/inputs/uploads you can define the condition.
Condition string is put in brackets (...) after text label of parameter example:
Gold Letters (Parm1) - this means display Gold Letters parameter only is Parm1 (Engraving on a picture) is checked.
The condition is basically javascript code. It is not required for you to know javascript to achieve a lot knowing few operators. Here are common examples for parameter form picture (*IMAGE2*) *Wooden Box*:

Wooden Box (Parm1 && Parm2) - means display Wooden Box only if Engraving and Gold Letters are checked.

Wooden Box (Parm1 && !Parm2) - means display Wooden Box only if Engraving is checked and Gold Letters is not checked.

Wooden Box (CountryCode3 == 'CAN' && Parm1) - means display Wooden Box only if customer is visiting your site from Canada and Parm1 (Engraving) is checked .

Wooden Box (Time.Hour >= 13 && Time.Hour <= 15 && Parm2) - means display Wooden Box only if customer time is between 1PM and 3PM and Gold Letters is checked.

Wooden Box (hecp_variant.price > 100) - means display Wooden Box only if original price of current variant is greater than 100.

Common operators:

&& AND
|| OR
> greater than
>= greater or equal to
< less than
<= less or equal to
== equal to
Input1.value.indexOf('search') > -1 tests if Input1 contains word search

Conditions can be applied to price parameters, inputs, and uploads

full list of environment variables you can use is given in section 6 of this document

You can make parameters required if you put asterisk * in front of label text with one empty space as separator "* Engraving". This is nonsense for single-option parameter because it can be either checked or not but if you multi-option parameter you may or may not require from the user to make a decision.

3.4 Order of fields

A common need is to order fields in some way. To do this is it enough to put integer number in front of field label text separated with space. Example:

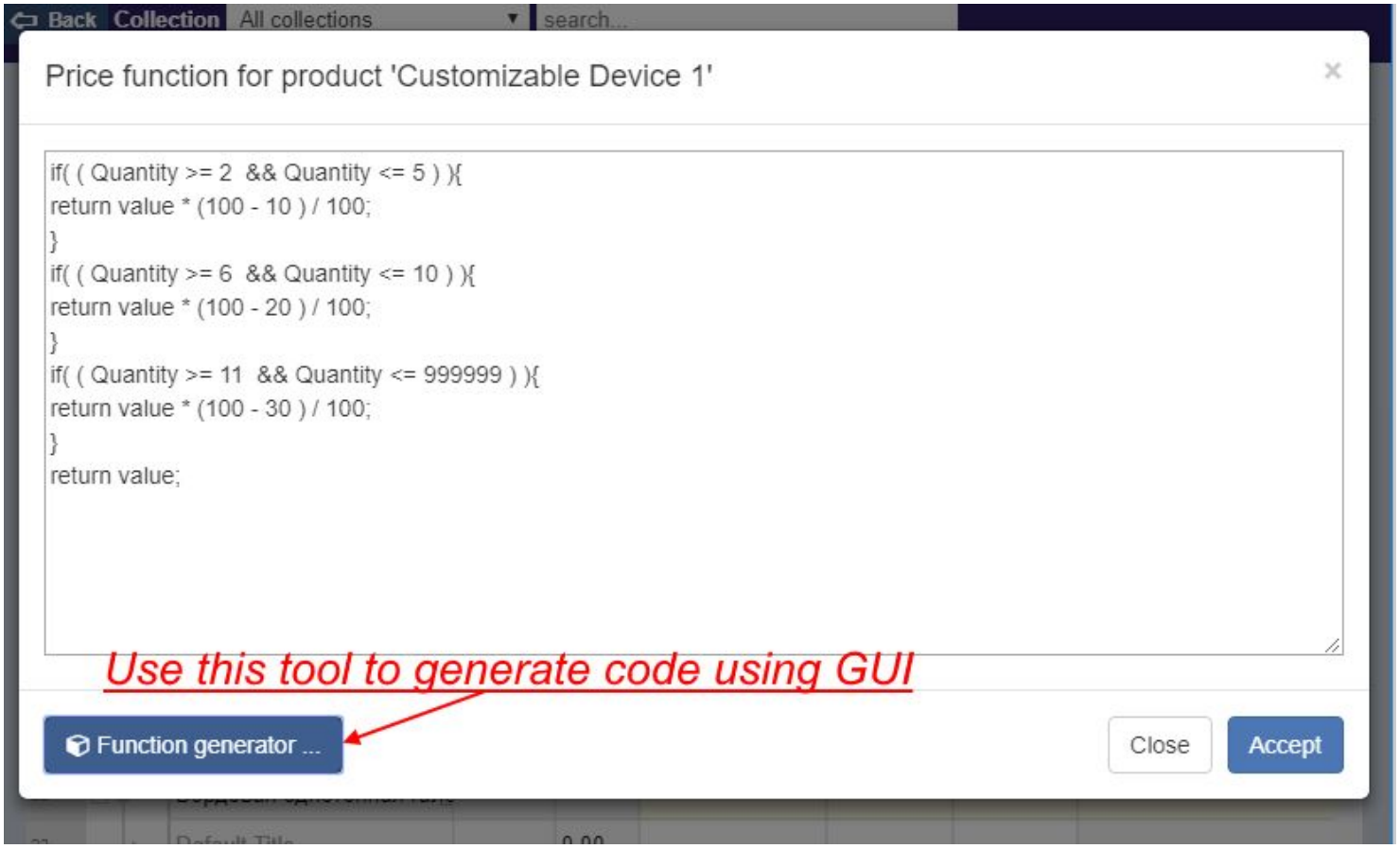
*1 * Engraving*
2 Your design file
*** 3 Gold Letters***

Note that for Engraving first come order number and then requirement Asterix and for Gold Letters, it is the reverse. We did put this in the example so you could see that it is not important.

3.5 Product price formula

Note: Do not add quantity as parameter. You theme already has this fields build-in. You just have to enable it, if it is not there already. Go to 'Theme editor' then find Product Page settings. There should be a checkbox to enable quantity box on product page!

https://help.shopify.com/themes/customization/products/features/add-quantity-selector



(Image shows function that reduces price based on order quantity. Next image shows this code setup in *graphical function generator.*)

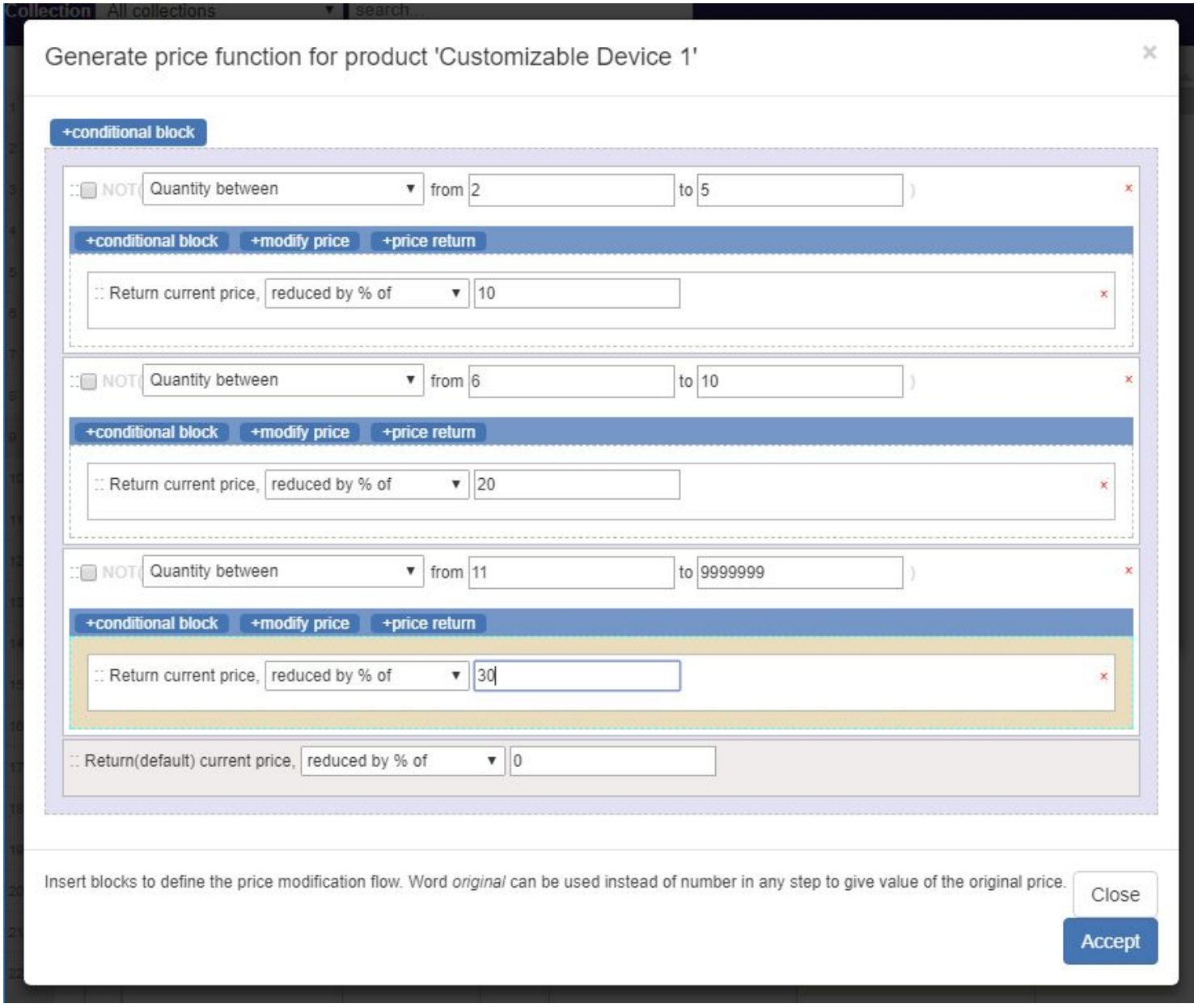


IMAGE5

Product price formulas are used to modify product price based on field and global values. Except for quantity based price need for this formula is rare, only in some specific cases. For most common uses global formula (4.) should do the job for you.
Product price formula applies only on the product page as it is related to a particular product. Use of it is shown under example *5.4 "Engraving price per letter".* In that example, final price depends on input field value which is only available on the product page so the use of it is necessary in this case.
Value of this field can be considered as the body of the javascript function (and that is actually is) looking like this:

```
function (value, parameter ){
//Here comes the value
}
```

value is original price modified by price parameters and global formula. Order of modifications is the global formula, price parameters and at end product formula.

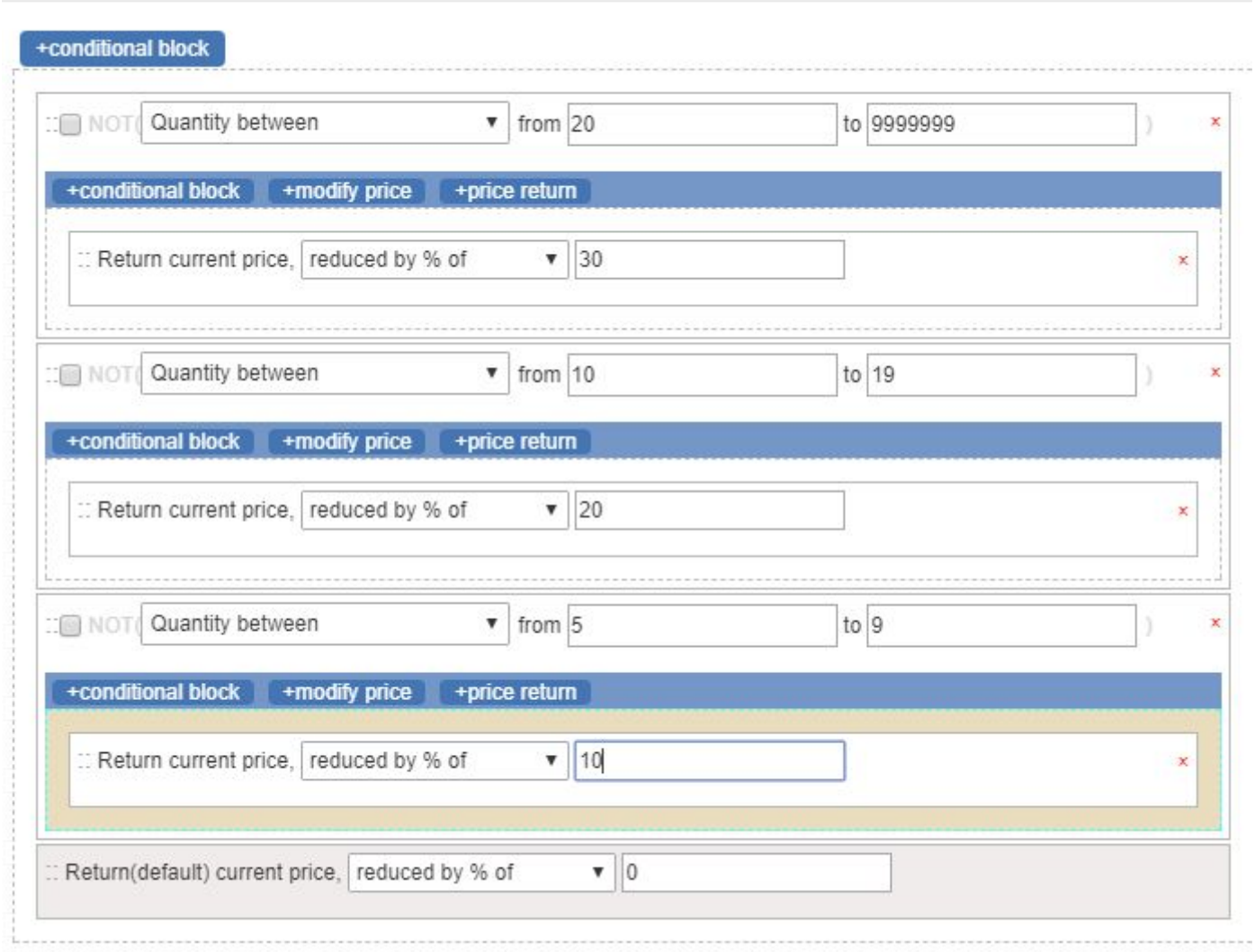
So to simply decrease price using this formula by 30% you would set:

*return value * 0.7;*

For some quantity based calculation it can be:

if(Quantity > 20) *return value * 0.7;*
if(Quantity > 10) *return value * 0.8;*
if(Quantity > 5) *return value * 0.9;*
*return value * 1;*

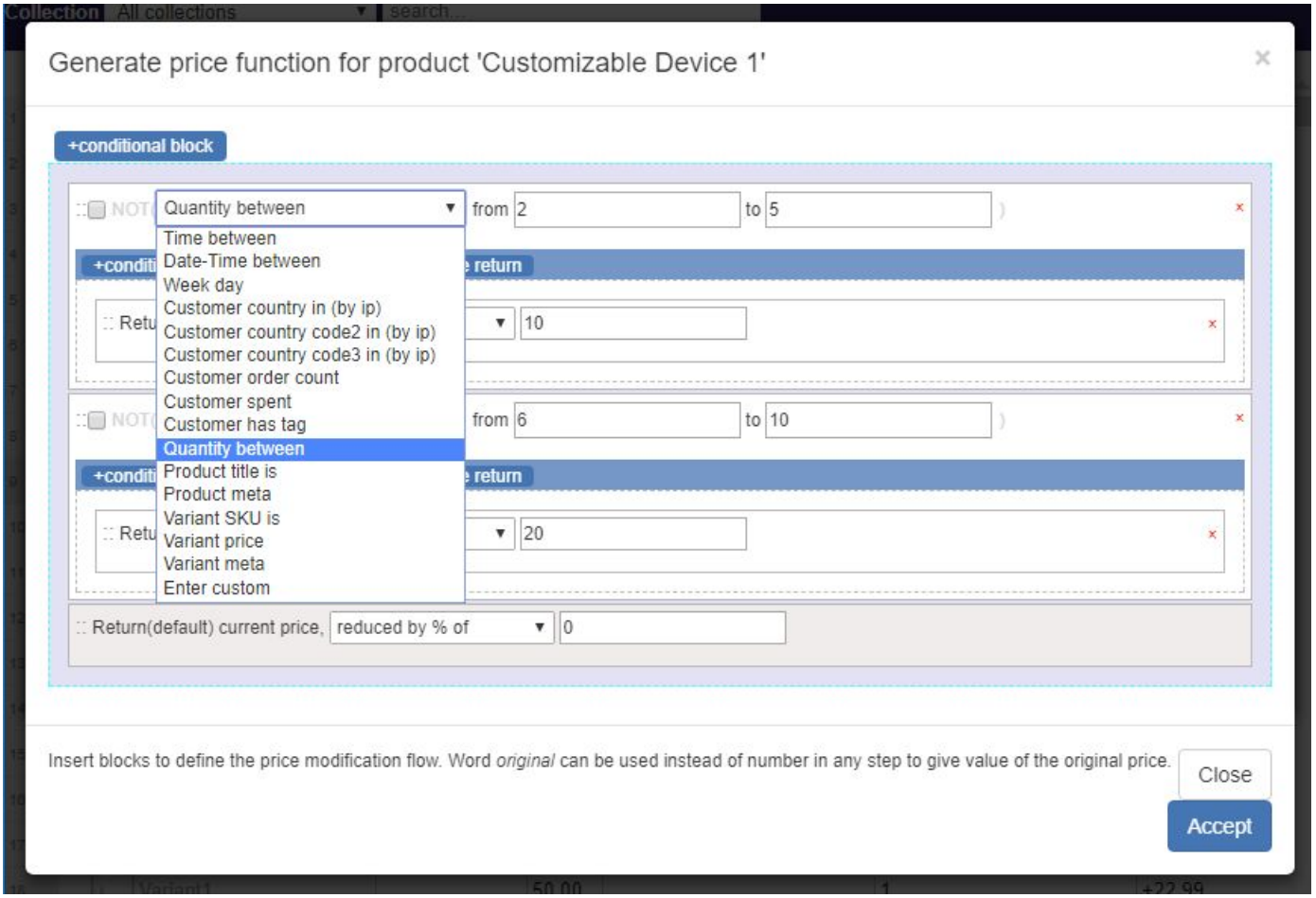
This function would look as on next image in function generator:



If you make syntax error formula will just not be applied. Second argument *parameter* will always be false unless **Apply formulas on parameter values** are checked and function is applying on a parameter. For product price, it will be always false.

Graphical function generator (product specific)

To make function definition easier we provided “Function Generator” option. Pop-up that opens the graphical function generator is located in top right corner. Using this tool you can generate almost every rule needed.



The function generator allows 3 types of elements:

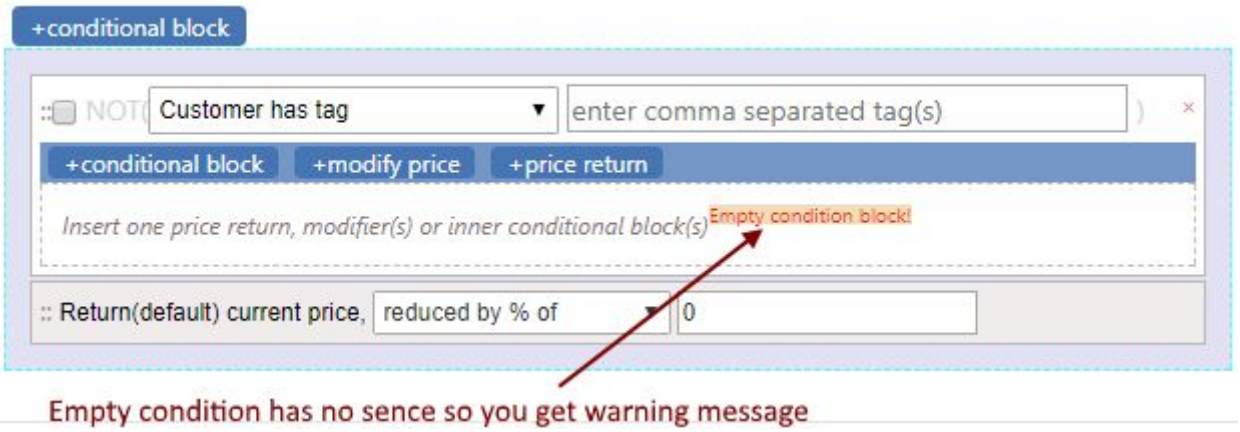
1. Conditional block - creates separated scope of program flow entered if its criteria is met. It can contain any of elements.
2. Return value - immediately leaves price function no matter of nesting level within conditional blocks. It allows final price modification.
3. Price modification - applies some calculation rule on current price value. You can have multiple of this blocks in sequence if needed.

You can insert *conditional blocks* anyware to control calculation flow. Inside each *calculation block* you can insert more nested *conditional blocks* and also modificators of running value or immediate returns from formula.

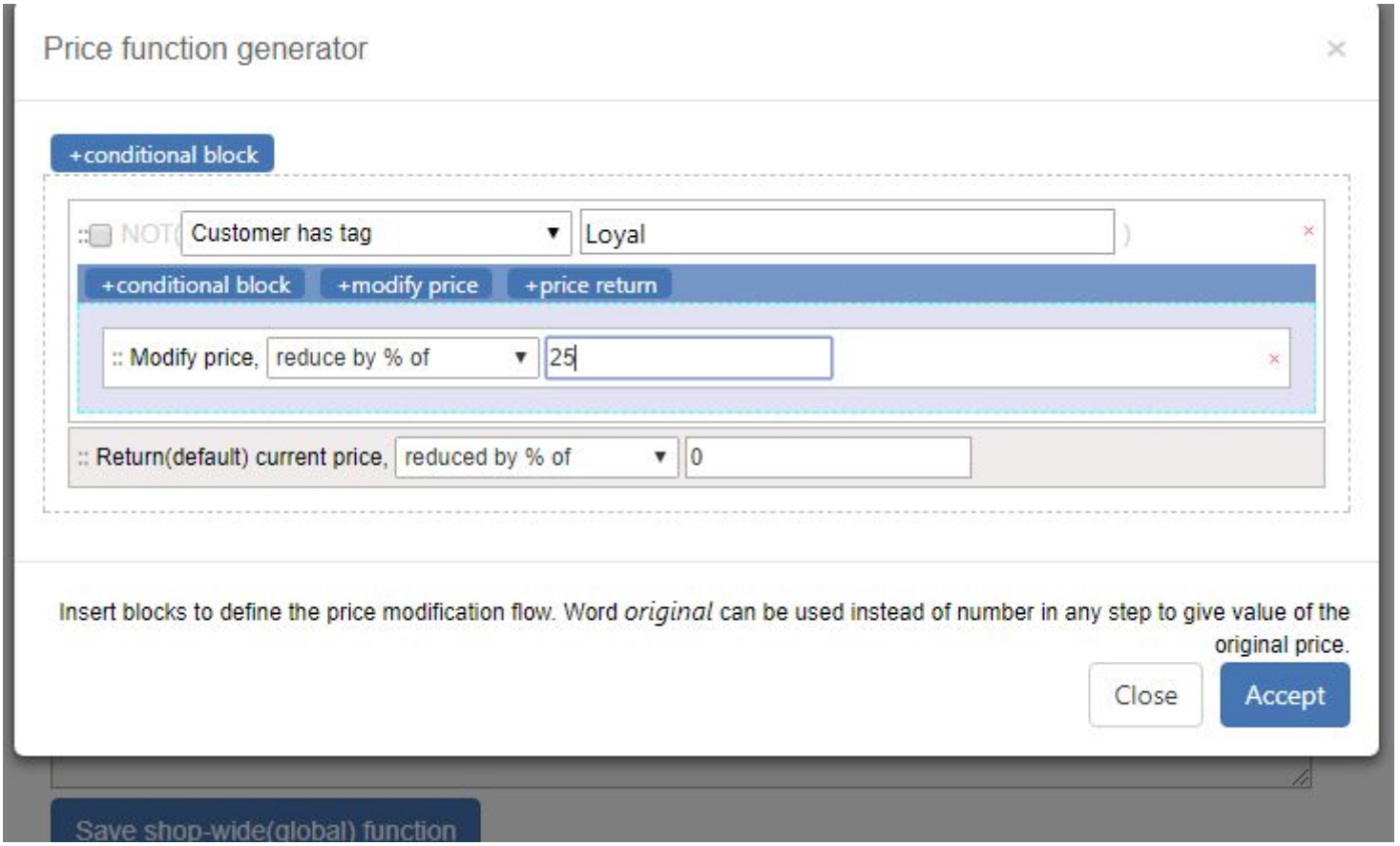
For each nested *condition block* you have to define condition to be satisfied in order of entering its nested blocks. Depending of rules you want to apply you may chose to do something with price and immediately return value like that (you use “Return price” block in this case) or you may choose to modify it and continue with other checks if it depends upon more that one parameter (you use “Modify price” block in this case).

Each function composition ends with final(default) return. You can not change its position or remove it.

When function graphical composition has empty block or *return* that prevents execution of some other block you will get warning message. For example if you put running price modificator after return in same block modificator will never be able to execute becue if program flow reaches *return* it will exit whole function immediately. To compose valid function you have to eliminate such conflicts if they appear.



4. Shop-wide (global) formula



(Image shows example of reducing prices by 25% for all customers having tag "Loyal")

The global formula is used to modify all product prices based on global variables (see section 6).
On a product page, this formula is applied before price parms or product formula.

Value of this box can be considered as the body of the javascript function (and that is actually is) looking like this:

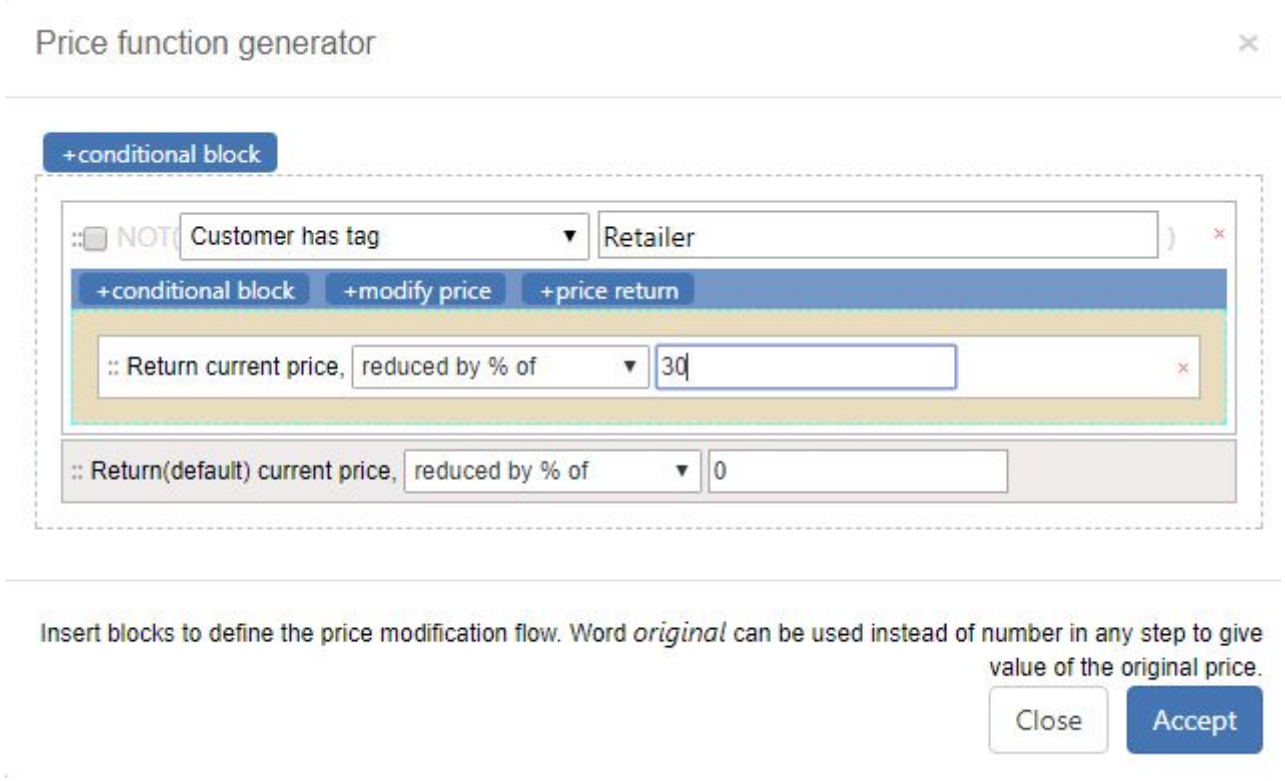
```
function (value, parameter ){
//Here comes the code form box
}
```

Argument *value* is original product (or parameter) price. Second argument *parameter* will always be false unless **Apply formulas on parameter values** are checked and function is applying on the parameter. For product price, it will be always false.

Here is the example of very easy retailer implementation. Enable log-in for your shop. Create accounts for your retailers (if not exist).
Edit this account and assign tag "Retailer" to them. To reduce the price by 30% for your retailers enter a global formula like this:

*if(CustomerTags['Retailer']) return value * 0.7;*
*return value * 1;*

This function would look as on next image in function generator:

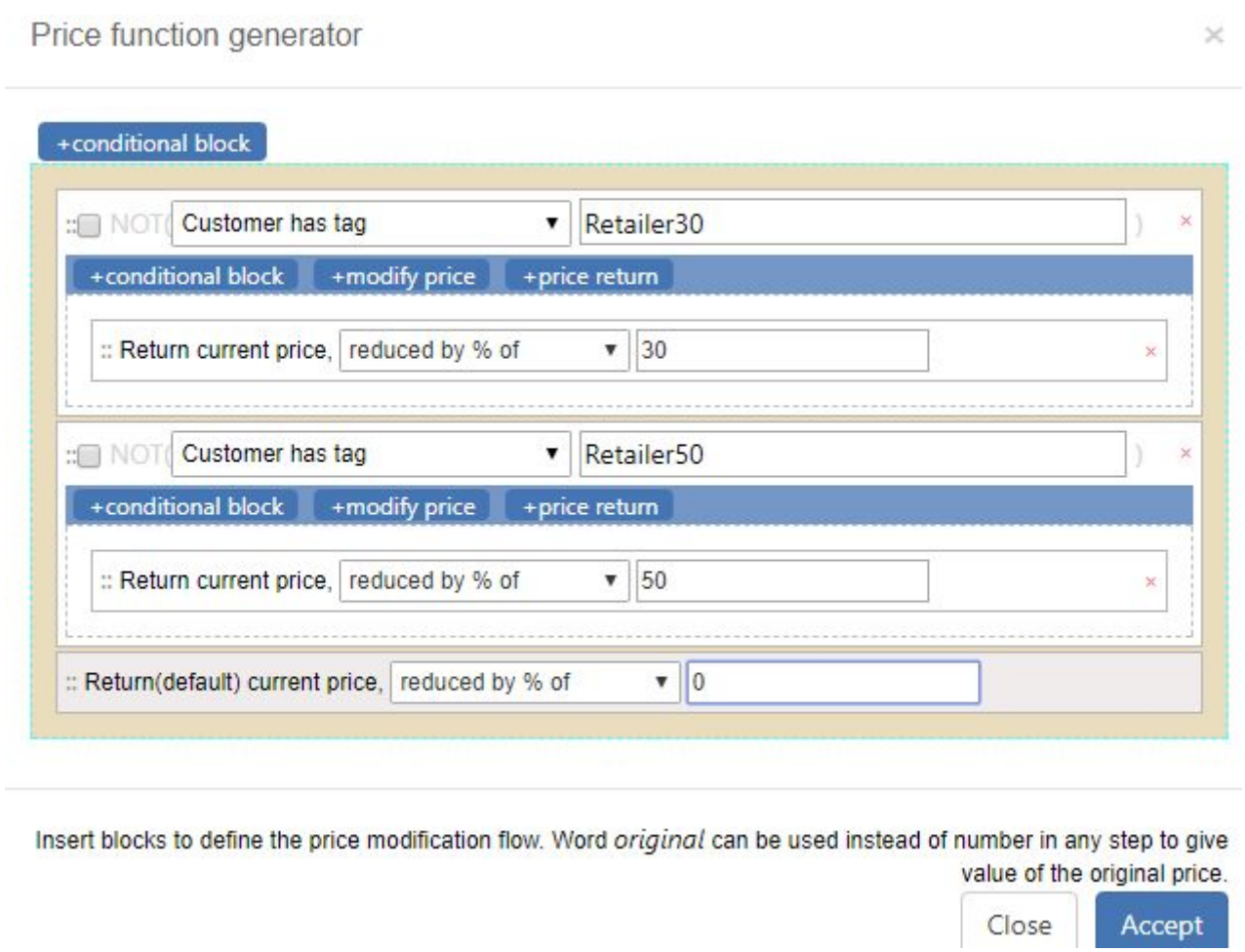


You would see that after they log in that all prices are reduced by 30%.

You can have multiple kinds of retailers. All you need to do is to introduce tag for each. Like:

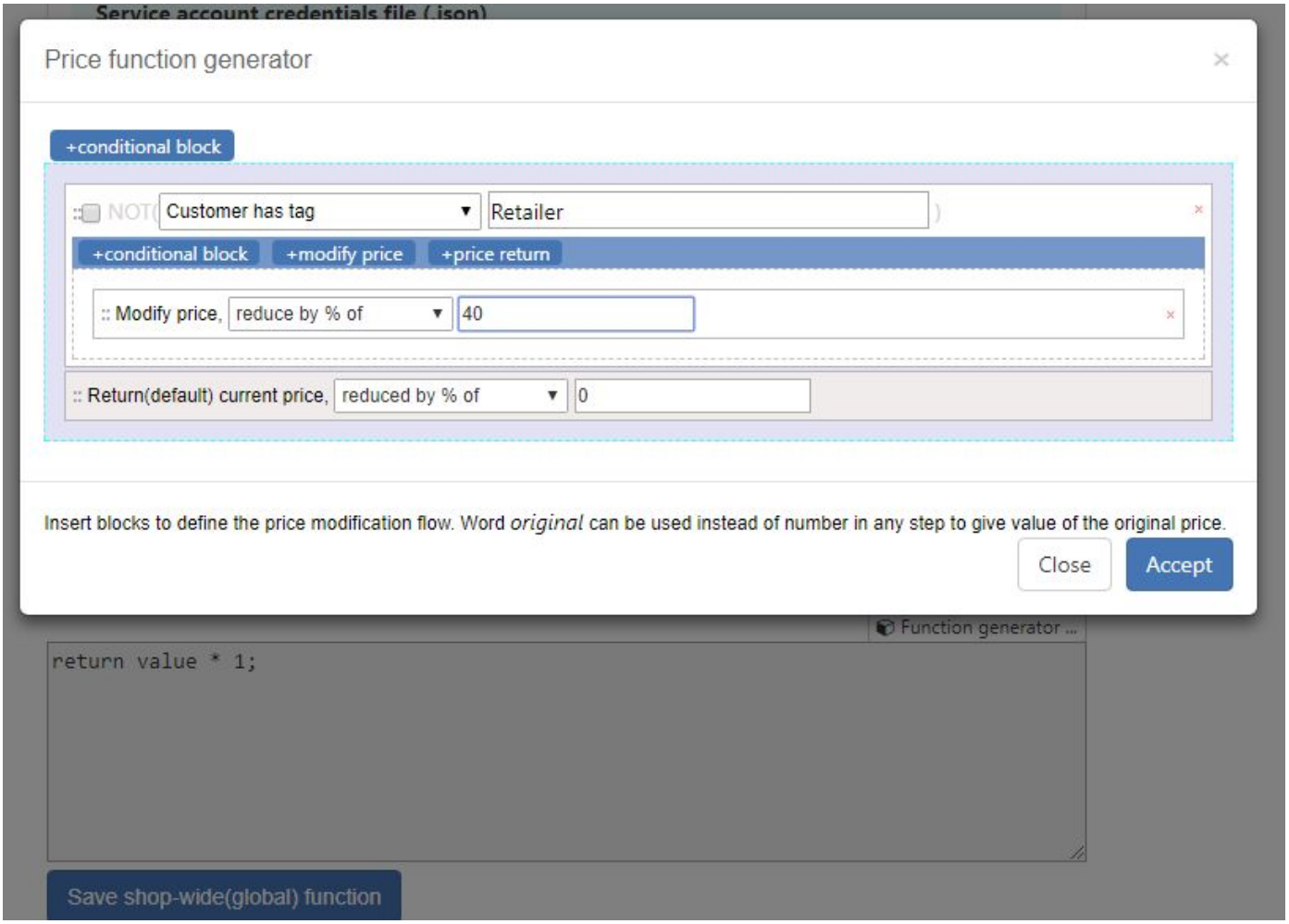
*if(CustomerTags['Retailer30']) return value * 0.7;*
*if(CustomerTags['Retailer50']) return value * 0.5;*
*return value * 1;*

This formula would look as on next image in function generator:



Graphical function generator (Shop-wide / global)

To make function definition easier we provided “Function Generator” option. Pop-up that opens the graphical function generator is located in top right corner. Using this tool you can generate almost every rule needed.



The function generator allows 3 types of elements:

1. Conditional block - creates separated scope of program flow entered if its criteria is met. It can contain any of elements.
2. Return value - immediately leaves price function no matter of nesting level within conditional blocks. It allows final price modification.
3. Price modification - applies some calculation rule on current price value. You can have multiple of this blocks in sequence if needed.

You can insert *conditional blocks* anyware to control calculation flow. Inside each *calculation block* you can insert more nested *conditional blocks* and also modificators of running value or immediate returns from formula.

For each nested *condition block* you have to define condition to be satisfied in order of entering its nested blocks. Depending of rules you want to apply you may chose to do something with price and immediately return value like that (you use “Return price” block in this case) or you may choose to modify it and continue with other checks if it depends upon more that one parameter (you use “Modify price” block in this case).

Each function composition ends with final(default) return. You can not change its position or remove it.

When function graphical composition has empty block or *return* that prevents execution of some other block you will get warning message. For example if you put running price modificator after return in same block modificator will never be able to execute becue if program flow reaches *return* it will exit whole function immediately. To compose valid function you have to eliminate such conflicts if they appear.

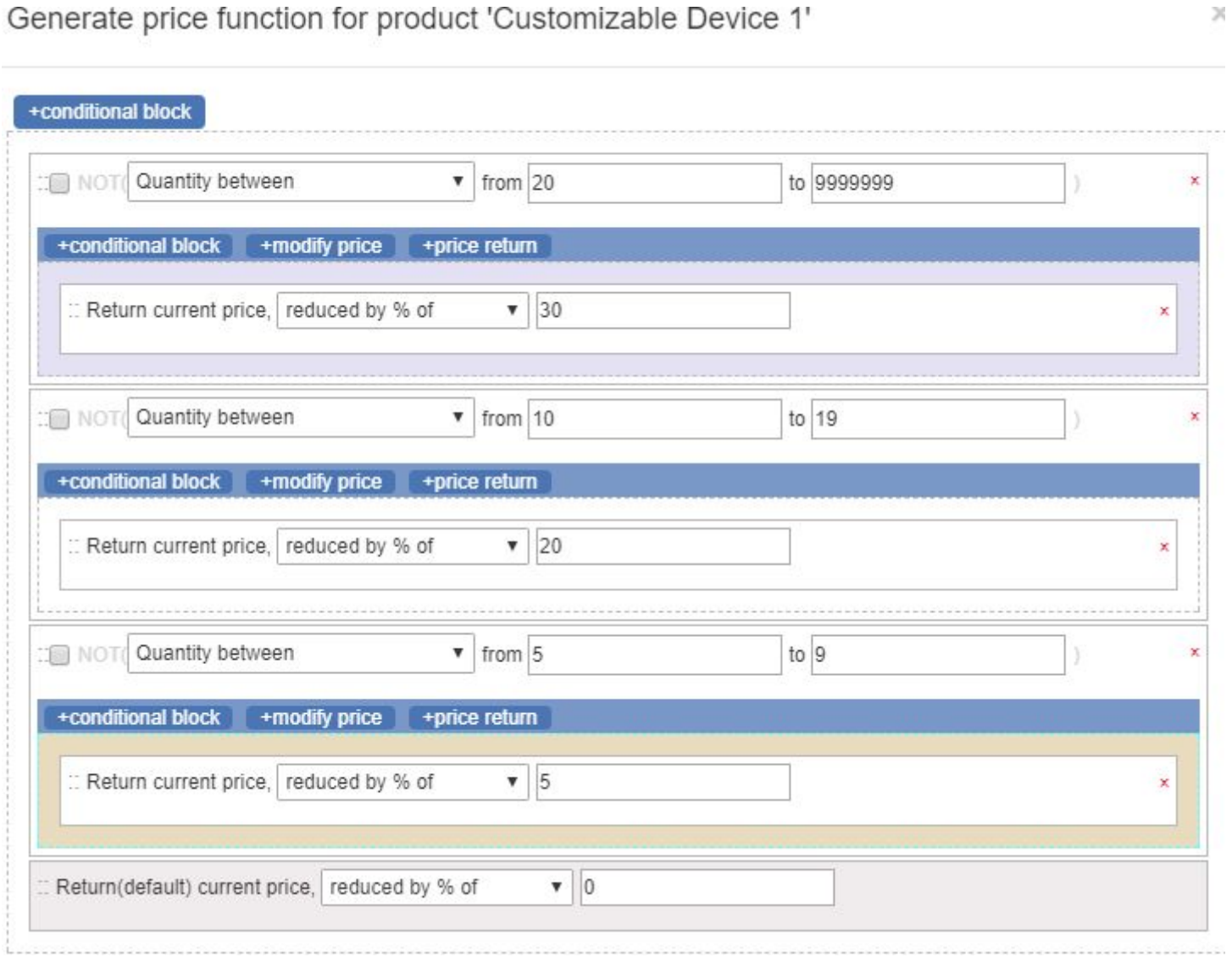
5. Examples

5.1 Basic additional parameters example

Title	Parm1	Parm2	Parm3	Input1	Function
Customizable Device 1	* 1 Shield	3 Gold Letters (Parm1)	Extra battery	2 Shield engrave name (Parm1)	if(Quantity > 20) return value * 0.7; if(Quantity > 10) re
14GB / No / Yes	;Red:+6;Blue:+8	10	20		
16GB / Yes / No	;Red:+6;Blue:+8	11	21		
5GB / Yes / Yes	;Red:+6;Blue:+8	12	20		
12GB / Yes / No	;Red:+7;Blue:+9;Yellow:+10	13	21		
12GB / No / No	;Red:+7;Blue:+9	14	20		
4GB / No / Yes	;Red:+7;Blue:+9	15	21		
4GB / Yes / Yes	;Red:+7;Blue:+9	15	1800mAh:23;3000mAh:25		

The product has 3 parameters. Shield, Text to put on a shield, Gold Letters, Extra battery.
If users order more than 20 pieces it has discount of 30%
If users order more than 10 pieces it has discount of 20%
If users order more than 5 pieces it has discount of 5%
The customer must also upload 2 design files if he selects Shield.

Function in FN generator looks like this:



Live example:

https://bulk-product-manager.myshopify.com/products/example-customizable-device

5.2 Reduced price for loyal customers (shop-wide formula)

Problem: We want to reduce all prices for customers that bought goods from us previously in total value greater than 1000.00

Solution: Use global formula, enter :

*if(CustomerTotalSpent > 1000.00) return value * 0.9;*
*return value * 1;*

In FN generator this formula looks like this:

5.3 Extra price for each design upload

Problem: we want to charge user 10$ per each PSD he uploads. Max is 3.
Solution:

In configuration enable sufficient number of uploads
(**Max. number of file uploads per product =** 3).

Enable uploads for a product by entering values is cells:
For Upload1 enter Design 1
For Upload2 enter Design 2
For Upload3 enter Design 3

In Function cell enter:

return value + (Upload1.files[0] ? 10 : 0) + (Upload2.files[0] ? 10 : 0) + (Upload3.files[0] ? 10 : 0) ;

5.4 Engraving price per letter

Problem: we want to enable a customer to enter text for engraving. Engraving price for Variant1 is 1$ per letter and for Variant2 is 2$ per letter.

Title	Parm1	Input1	Function
Product that has engraving	Engraving, price per letter	Engraving text (Parm1)	return value + (Input1.value.length - 1) * Parm1;
Variant2	1		
Variant1	2		

Solution: Enable Parm 1 by entering “Engraving, price per letter” in product Parm1 cell.
Set prices per letter for each variant. Enable Input1 that will show only if *Engraving, price per letter* is checked. So input *Engraving text (Parm1).* At last define product function as:

*return value + (Input1.value.length - 1) * Parm1;*

Note: for this function we could not use just FN generator because it uses value of HTML element **Input1.value.length**.

SEO:

Product that has engraving

62.00 RSD

Variant2

Engraving, price per letter +2.00

Engraving text

Hello!

Quantity

1

Add to Cart

A live example can bee is seen here:
https://bulk-product-manager.myshopify.com/products/engraving-example

6. Environment variables listing

Environment parameter	Available globally	Available on product page	Requires customer log-in	Description	Example values
CountryName	yes	yes		County name in English (IP DB)	Germany
CountryCode2	yes	yes		2 character uppercase country code (IP DB)	US
CountryCode3	yes	yes		3 character uppercase country code (IP DB)	USA
Time.Hour	yes	yes		Current hour in 24h format	15
Time.Minutes	yes	yes		Current minute of hour	34
Time.Date	yes	yes		Current date in month	26
Time.Month	yes	yes		Current month in year	11
Time.Year	yes	yes		Current year	2017
Time.WeekDay	yes	yes		Day of week 1 -> 7 representing Sun->Sat	3
CustomerTags[*'Name of tag'*]	yes	yes	yes	Register of logged in customer tags. It can be accessed as CustomerTags.Retailer or CustomerTags['Retaler'] (if tag name has more that	1

				one word you will obviously have to use second way)	
CustomerOrderCount	yes	yes	yes	Logged in customer previous orders count	10
CustomerTotalSpent	yes	yes	yes	Logged in customer previous orders total sum	1200.99
CustomerAcceptsMarketing	yes	yes	yes	True if logged in customer has accepted marketing from you	true
CustomerHasVerifiedEmail	yes	yes	yes	True if logged in customer has verified email	false
Quantity	no	yes		Quantity customer selected on product page	5
Parm1...Parm*N*	no	yes		*As explained under 3.1.1*	99.99
Input1...Input*N*	no	yes		*As explained under 3.2*	HTMLInput
Upload1...Upload*N*	no	yes		*As explained under 3.2*	HTMLInput
hecp_product	no	yes		*Currently selected product full JS object. See Shopify help*	{...}
hecp_variant	no	yes		*Currently selected variant full JS object. See Shopify help*	{...}
hecp_variant_id	no	yes		Currently selected variant system id	43573378957
productMeta	no	yes		Any meta of current product can be accessed as: ***productMeta['namespace_key']***	100
variantMeta	no	yes		Any meta of current variant can be accessed as: ***variantMeta['namespace_key']***	200

7. How to uninstall

If you just need to prevent app from operating it is enough to uncheck parameter "App is enabled(in front-end)" and save. No code from APP will be executed in this case.
If you want to completely remove app you need to unistall is from shop "APPs" and there is only one line (<script> HTML tag) in theme.liquid that starts like this:

*<script id='**he_cp_script_tag**' type='text/javascript' src='...*

Remove this line and save.

Note that time of first installation and payment data stays recorded so you can install the APP again not having to pay for it again. Also if you are in trial mode, starting time will be one recorded on first installation.

8. Troubleshoot

COMMON PROBLEMS:

Optionally after installing and configuring app you may go to front-end and refresh few times before you start to test it.

In some very rare cases on some very specific templates, it may happen that some features don't work. In such cases contact or support immediately and we will find a way to make the app compatible with your specific theme.

Product thumbnails on checkout page are not displayed. This app uses its own cart object that is extended version of default cart. At checkout app will generate “draft order” and lead customer to its payment. Because of technical limitations it is not possible to display product thumbs during checkout. If the cart contains no products with *“Calculated Prices”* parameters your customer will be lead to default order payment.

If you need to change color or font of cart elements because you don’t like them or they are not visible enough you can use this css entities (you can add css rules to your *theme.scss.liquid*):

```
.hecp-cart hecpcartline Iteminfo a.vtitle{
    /*item name in cart*/
}

.hecp-cart hecpcartline a.hecp_remove span{
    /*remove button”*/
}

.hecp-cart hecpcartline lineproperties propertyname{
    /*added property name*/
}

.hecp-cart hecpcartline lineproperties propertyvalue{
    /*added property value*/
}

.hecp-cart hecpcartline linequantity input{
    /*quantity text box*/
}

.hecp-cart hecpcartline.subtotal linepricetotal{
    /*subtotal*/
}

.hecp-cart .st_note{
    /*note box*/
}

.hecp-cart .btn-hecp-checkout{
    /*checkout button*/
}
```

9. Frequently Asked Questions

Are there any changes this APP makes to my shop that are not directly related to its primary functionalities?

APP alters your side cart and cart page because it needs to display additional information and modified prices. Also, at checkout, if there are no products in the cart that are customized for a client by this app client will use regular checkout. If there are such products the APP will create draft order and lead a customer to its payment. There are slight differences between checkout pages in this two cases. For example in second case

images don’t have thumbs. Both are strictly controlled by Shopify so there is no way to change that. Here is the list of differences:

- Discount codes cannot be used (but you can obviously create some sort of discount codes using this app).
- The cart navigation can no longer be used in the checkout with the draft orders API.
- The store logo may not be seen at checkout / does not link back to the shop outside checkout.
- The custom domain cannot be seen at checkout.

In what way order made with this APP is handled?

When there are products orders customized by this APP, APP will create so-called “Draft Order”. This is same as when you create an invoice from backend and send it to the customer. On this “Draft Order” payment Shopify will automatically create regular order out of it.

What are differences between product function and global function?

The global function is applied to all product prices on your site no matter on which page they are shown. Product function applies only while your customer is on the product page. This also implies set of conditional parameters that can be used in them. The global function cannot use Quantity parameter for example because that is related to a particular product order.

Product functions should be only used in cases when you need to calculate a price based on user input of parameters for a particular product. Such example is a price for engraving when each letter is charged or order quantity based pricing.

I’m not a programmer how will I handle functions myself?

Function generator can generate almost every function customer may need. There are some specific cases when something special should be taken into account (like cost per engraving letter example). In this cases, you can turn to our support staff (support@holest.com) and you will get an answer soon as possible.