Shopify Scripts

The Script Editor app lets you create scripts that are run each time a customer adds items to their online cart. Scripts can have many uses, from discounting products with specific tags to running promotions such as "buy 2, get 1 free". Scripts are written with a Ruby API that allows a great deal of control and flexibility. You can use scripts in your online store only.

Are you unfamiliar with Ruby? Learn the basics ›

> **Tip**
>
> Scripts and the Script Editor app are available to Shopify Plus merchants only.

There are different script types. A script is assigned a type when you create the script in the Script Editor app, based on which script template you choose to start with:

Line item scripts

Line item scripts affect line items in the cart and can change prices and grant discounts.

Some methods can only be used in line item scripts.

Shipping scripts

Shipping scripts interact with shipping, and can change shipping methods and grant discounts on shipping rates.

Some methods can only be used in shipping scripts.

Payment Scripts

Payment scripts interact with payments, and can rename, hide and reorder payment gateways. Note that payment scripts do not interact with payment gateways shown before the checkout screen, like Apple Pay.

Some methods can only be used in payment scripts.

In this article

- General methods
- ShippingAddress
- Line item methods
- Shipping methods
- Payment methods
- Examples

General methods

The following methods are usable in any type of script:

Input

Method	Return type	Description
`.cart`	Cart	Returns a mutable cart object.

Cart

Method	Return type	Description
`.customer`	Customer	Returns the owner of the cart (if any).
`.shipping_address`	ShippingAddress	Returns the shipping address of the owner of the cart (if any).

Method	Return type	Description
`.discount_code`	varies	Returns: • `nil` if the cart has no discount code • `CartDiscount::FixedAmount` if the cart has a fixed discount amount • `CartDiscount::Percentage` if the cart has a percentage discount • `CartDiscount::Shipping` if the cart has a shipping discount `discount_code` will be present if a discount has been applied to the cart. This does not necessarily mean that the price of the cart will change. For example, if a discount applies to carts above $50, and a script reduces the cart price below $50, `discount_code` will be present but the price of the cart will not change. See an example of `discount_code`.
`.line_items`	List<LineItem>	Returns a list containing the line items in the cart.
`.subtotal_price`	Money	Returns the subtotal price of the cart after line item discounts are applied but before discount codes are applied.

CartDiscount::FixedAmount

Method	Return type	Description
`.code`	String	Returns the discount code used to apply the discount.
`.amount`	Money	Returns the money amount of the discount.

Method	Return type	Description
`.reject({ message: String })`	nil	Rejects the discount code applied to the cart. A `message` is required.
`.rejected?`	Boolean	Returns whether the discount code was rejected.

CartDiscount::Percentage

Method	Return type	Description
`.code`	String	Returns the discount code used to apply the discount.
`.percentage`	Integer	Returns the percentage amount of the discount.
`.reject({ message: String })`	nil	Rejects the discount code applied to the cart. A `message` is required.
`.rejected?`	Boolean	Returns whether the discount code was rejected.

CartDiscount::Shipping

Method	Return type	Description
`.code`	String	Returns the discount code used to apply the discount.
`.reject({ message: String })`	nil	Rejects the discount code applied to the cart. A `message` is required.
`.rejected?`	Boolean	Returns whether the discount code was rejected.

Customer

Method	Return type	Description
`.id`	Integer	Returns the customer's ID number.
`.email`	String	Returns the customer's email address.
`.tags`	List<Tag>	Returns a list of strings representing any tags set for a customer.
`.orders_count`	Integer	Returns the total number of orders a customer has placed.
`.total_spent`	Money	Returns the total amount that the customer has spent on all orders.
`.accepts_marketing?`	Boolean	Returns whether the customer accepts marketing.

LineItem

Method	Return type	Description
`.line_price`	Money	The price of the line item.
`.discounted?`	Boolean	Returns whether a discount has changed the price of the line item.
`.properties`	hash	Returns the properties that were specified for this line items.
`.variant`	Variant	Returns the specific product variant represented by the line item.
`.quantity`	Integer	Returns the quantity of this line item.
`.total_weight`	grams	Returns the total weight of this line item.

List

Method	Return type	Description
`.new`	List	Creates a new object to represent a list.
`.[]`	Element or nil	Returns the element at the specified index.
`.&`	List	Returns a new list containing elements common to the two lists, with no duplicates.
`.empty?`	Boolean	Returns `true` if the list contains no elements.
`.first`	Element or nil	Returns the first element or `nil` if the list is empty.
`.index(*args, &block)`	int or nil	Returns the index of the first element of the list. If a block is given instead of an argument, returns the index of the first element for which the block is true.
`.rindex(*args, &block)`	int or nil	Returns the index of the last element of the list. If a block is given instead of an argument, returns the index of the first element for which the block is true.
`.last`	Element or nil	Returns the last element or `nil` if the list is empty.
`.length`	int	Returns the number of elements in the list.
`.size`	int	Alias for length.
`.each(*args, &block)`	List	Calls a block once for each element in the list, passing the element as a parameter to the block.

ShippingAddress

Method	Return type	Description
`.name`	string	Returns the name of the of the person associated with the shipping address.
`.address1`	string	Returns the street address portion of the shipping address.
`.address2`	string	Returns the optional additional field of the street address portion of the shipping address.
`.phone`	string	Returns the phone number of the shipping address.
`.city`	string	Returns the city of the shipping address.
`.zip`	string	Returns the ZIP code of the shipping address.
`.province`	string	Returns the province/state of the shipping address.
`.province_code`	string	Returns the abbreviated value of the province/state of the shipping address.
`.country_code`	string	Returns the abbreviated value of the country of the shipping address.

Money

Method	Return type	Description
`.new`	Money	Creates a new object to represent a price.
`.zero`	Money	Creates a new object with a price of zero.
`+`	Money	Adds two `Money` objects.
`-`	Money	Subtracts one `Money` object from another.
`*`	Money	Multiplies a `Money` object by a number.

Money examples

```
Money.new(cents: 1000)
```

Creates a `Money` object representing 1000 cents, or $10.

```
Money.new(cents: 100) * 50
```

Creates a `Money` object representing $1, then multiplies that amount by 50. Returns a `Money` object representing $50.

Variant

Method	Return type	Description
`.id`	Integer	Returns the ID number of the variant.
`.price`	Money	Returns the unit price of the variant.
`.product`	Product	Returns the associated product of the variant.
`.skus`	List<String>	Returns the stock keeping units (SKUs) of the variant, which are often used for tracking inventory.
`.title`	String	Returns the title of the variant.

Product

Method	Return type	Description
`.id`	Integer	Returns the ID number of the product.
`.gift_card?`	Boolean	Returns whether the product is a gift card.

Method	Return type	Description
`.tags`	List<Tag>	Returns a list of strings representing the tags that are set for this product.
`.product_type`	String	A categorization that a product can be tagged with, commonly used for filtering and searching.
`.vendor`	String	Returns the vendor of this product.

Kernel

Kernel is a Ruby module that is included in every class. As a result, its methods are available to every object. These methods act in the same way as global functions act in other languages.

Method	Return type	Description
`.exit`	none	Ends execution of the current script without error. If this is run before anything is assigned to `Output.cart`, the script has no effect. This is a useful way to exit scripts, for example, if the customer is ineligible to run the script.

Kernel example

```
customer = Input.cart.customer
if customer && customer.email.end_with?("@mycompany.com")
  # Employees are not eligible for this promotion.
  exit
end
```

Line item methods

The following methods are only usable in **line item scripts**:

Cart

Method	Return type	Description
`.subtotal_price_was`	Money	Returns the subtotal price of the cart before any discounts were applied.
`.subtotal_price_changed?`	Boolean	Returns whether the subtotal price has changed.

LineItem

Method	Return type	Description
`.change_line_price(Money new_price, { message: String })`	Money	Change the price of the line item to the amount specified. A `message` is required. `new_price` must be lower than the current price.
`.original_line_price`	Money	Returns the original price of the line item before scripts and discounts were applied.
`.line_price_was`	Money	Returns the price of the line item before changes were applied by the current script.
`.line_price_changed?`	Boolean	Returns whether the price of the line item has changed.
`.change_properties(hash new_properties, { message: String })`	hash	Sets new properties for a line item. The original properties hash is stored in `properties_was` and the properties hash that is passed to the method becomes the new properties for the line item.

Method	Return type	Description
`.properties_was`	hash	Returns the original properties hash of the line item before any changes were applied.
`.properties_changed?`	Boolean	Returns whether the properties for the line item have been changed.
`.split({ take: Integer })`	LineItem	Splits a line item into two line items. `take` specifies what quantity to remove from the original line item to create the new line item.

.split example

```
if original_line_item.quantity >= 3
  new_line_item = original_line_item.split(take: 1)
  new_line_item.change_line_price(Money.new(cents: 500), message: "Third hat for 5 dollars")
  cart.line_items << new_line_item
end
```

This example script splits a line item called `original_line_item` into two line items. The new line item will have a quantity of 1 (specified by `take: 1`). The script then applies a discounted price to the new line item with the message "Third hat for 5 dollars".

Variant

Method	Return type	Description
`.compare_at_price`	Money	Returns the compare at price of the variant. Returns `nil` if the variant doesn't have a compare at price.

Shipping methods

The following methods are usable in **shipping scripts**:

Input

Method	Return type	Description
`.shipping_rates`	ShippingRateList	Returns a list of all the shipping rates.

ShippingRateList

Method	Return type	Description
`.delete_if`	ShippingRateList	Delete shipping rates using an optional code block. See the documentation for Ruby's `delete_if` method.
`.sort!`	ShippingRateList	Sort the shipping rates using the comparison operator or using an optional code block. See the documentation for Ruby's `sort!` method.
`.sort_by!`	ShippingRateList	Sort the shipping rates using an optional code block. See the documentation for Ruby's `sort_by!` method.

ShippingRate

Method	Return type	Description
`.code`	String	Returns the code of the shipping rate.
`.markup`	String	Returns the markup for shipping rate's description.
`.name`	String	Returns the name of the shipping rate.

Method	Return type	Description
`.price`	Money	Returns the price of the shipping rate.
`.source`	String	Returns the source (the carrier) associated with the shipping rate.
`.change_name(String new_name, { message: String })`	String	Changes the name of the shipping rate. A message is required.
`.apply_discount(Money discount, { message: String })`	Money	Applies a discount of the specified fixed amount. The price cannot be reduced below 0. A message is required.
`.phone_required?`	Boolean	Returns `true` if a phone number is required to get the shipping rate, or `false` if a phone number is not required.

Payment methods

The following methods are usable in **payment scripts**:

Input

Method	Return type	Description
`.payment_gateways`	PaymentGatewaysList	Returns a list of all the payment gateways in the store.

PaymentGatewayList

Method	Return type	Description

Method	Return type	Description
`.delete_if`	PaymentGatewayList	Delete payment gateways using an optional code block. See the documentation for Ruby's `delete_if` method.
`.sort!`	PaymentGatewayList	Sort the payment gateways using the comparison operator or using an optional code block. See the documentation for Ruby's `sort!` method.
`.sort_by!`	PaymentGatewayList	Sort the payment gateways using an optional code block. See the documentation for Ruby's `sort_by!` method.

PaymentGateway

Method	Return type	Description
`.name`	String	Returns the name of the payment gateway.
`.enabled_card_brands`	List<String>	If the payment gateway supports credit cards, returns a list of the credit card types allowed by the store. If the gateway doesn't support credit cards, returns an empty list.
`.change_name(String new_name)`	String	Changes the name of the payment gateway.

Examples

In the following line item script example, when a customer orders a product that is not a gift card, then the price of the product is reduced by $9. Also, the total amount

that the customer has spent throughout all visits to your store is shown:

```
customer = Input.cart.customer
Input.cart.line_items.each do |line_item|
  product = line_item.variant.product
  next if product.gift_card?
  line_item.change_line_price(line_item.line_price - Money.new(cents: 900),
message: customer.total_spent)
end

Output.cart = Input.cart
```