

Tutorials

- [How do I add a custom price to a product with a module?](#)

How do I add a custom price to a product with a module?

In this mini tutorial, we'll add checkboxes to a product page that allow customers to select additional charges. The product page will update the additional price dynamically as options are selected, and the total will continue to update when the product price changes, for example when a different variant is selected.

This tutorial assumes you already have a module set up, or are happy working directly from your app service provider using the `boot` method. See [here](#) for information on setting up a module.

1. Add the extra prices

We'll start by defining the available extra prices in our module service provider. For this example, we'll use an array as the source of truth rather than database records:

```
protected $prices = [
    [
        'id' => 1,
        'name' => 'Pay £1 more',
        'price' => [
            'inc' => 100,
            'ex' => 83.33,
        ],
    ],
    [
        'id' => 2,
        'name' => 'Pay £5 more',
        'price' => [
            'inc' => 500,
            'ex' => 416.66,
        ],
    ],
    [
        'id' => 3,
```

```
'name' => 'Pay £10 more',  
'price' => [  
    'inc' => 1000,  
    'ex' => 833.33,  
],  
],  
];
```

2. Add the prices data to the products page

Next, we'll make the extra prices available to the product page and add the JavaScript that will handle the checkboxes.

First, create a `javascript.twig` view in your module's resources directory. We'll populate this later.

We can then extend `Aero\Store\Http\Responses\ProductPage` to add the `$prices` data to the product page:

```
\Aero\Store\Http\Responses\ProductPage::extend(function  
(\Aero\Store\Http\Responses\ProductPage $page, \Closure $next) {  
    $page->setData('extra_prices', $this->prices);  
  
    \Aero\Store\Pipelines\ContentForBody::extend(function (string $content, \Closure $next) {  
        $content .= view('acme::javascript');  
  
        return $next($content);  
    });  
  
    return $next($page);  
});
```

3. Add the extra price checkboxes

In `product.twig`, we can display the product's total price and the additional price selected by the customer:

```

Total Price
<p v-text="total_price.inc"></p>

Total Extra Price
<p v-text="additional_prices['extra-prices'].inc" v-if="additional_prices['extra-
prices']"></p>

{% for extra in extra_prices %}
    <div>
        <input type="checkbox" id="extraPrice{{ extra.id }}" value="{{ extra.id }}" data-extra-
price-inc="{{ extra.price.inc }}" data-extra-price-ex="{{ extra.price.ex }}">
        <label for="extraPrice{{ extra.id }}">&nbsp;{{ extra.name }}</label>
    </div>
{% endfor %}

```

4. Update the additional price with JavaScript

Now we can add the code to `javascript.twig`

There are two things we need to handle:

- When a checkbox changes, calculate the total of all selected extras and pass it to Aero using the `product.set-additional-price` event.
- When the product is added to the cart, include the IDs of the selected extras in the request.

We'll initialise the additional price when `product.loaded` fires as well. This means the calculation is also correct if any of the checkboxes are selected by default:

```

<script>
    window.AeroEvents.on('product.loaded', function () {
        var extraPriceElements = document.querySelectorAll('input[type="checkbox"][data-extra-
price-inc]');

        for (var i = 0; i < extraPriceElements.length; i++) {
            extraPriceElements[i].addEventListener('input', updateAndSetAdditionalPrices);
        }
    });

```

```

function updateAndSetAdditionalPrices() {
    var price = { key: 'extra-prices', inc: 0, ex: 0 };
    var extraPriceElements =
document.querySelectorAll('input[type=checkbox]:checked[data-extra-price-inc]');

    for (var i = 0; i < extraPriceElements.length; i++) {
        price.inc += parseInt(extraPriceElements[i].dataset.extraPriceInc);
        price.ex += parseInt(extraPriceElements[i].dataset.extraPriceEx);
    }

    window.Aero.runEvent('product.set-additional-price', price);
}

updateAndSetAdditionalPrices();
});

window.AeroEvents.on('product.add-to-cart', function (data) {
    var extras = [];
    var extraPriceElements = document.querySelectorAll('input[type=checkbox]:checked[data-
extra-price-inc]');

    for (var i = 0; i < extraPriceElements.length; i++) {
        extras.push(extraPriceElements[i].value);
    }

    data.extras = extras;

    return data;
});
</script>

```

5. Add the selected extras to the cart item

We can extend `Aero\Store\Pipelines\CartItemBuilder` and read the `extras` array from the request. Each ID is matched against the `$prices` array, with invalid IDs filtered out before creating the cart item options.

Each valid extra becomes an `Aero\Cart\CartItemOption` with its corresponding inclusive price.

```

\Aero\Store\Pipelines\CartItemBuilder::extend(function (\Aero\Cart\CartItem $item) {
    collect(request()->input('extras', []))
        ->map(function ($extra) {
            return collect($this->prices)->firstWhere('id', $extra);
        })
        ->filter()
        ->each(function ($extra) use ($item) {
            $option = \Aero\Cart\CartItemOption::create($extra['name']);
            $option->setPriceInc($extra['price']['inc']);

            $item->addOption($option);
        });
});

```

Complete Service Provider

```

<?php

namespace Aerocargo\Acme;

use Aero\Cart\CartItem;
use Aero\Cart\CartItemOption;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Store\Http\Responses\ProductPage;
use Aero\Store\Pipelines\CartItemBuilder;
use Aero\Store\Pipelines\ContentForBody;

class ServiceProvider extends ModuleServiceProvider
{
    protected $prices = [
        [
            'id' => 1,
            'name' => 'Pay £1 more',
            'price' => [
                'inc' => 100,
                'ex' => 83.33,
            ],
        ],
    ],

```

```

[
    'id' => 2,
    'name' => 'Pay £5 more',
    'price' => [
        'inc' => 500,
        'ex' => 416.66,
    ],
],
[
    'id' => 3,
    'name' => 'Pay £10 more',
    'price' => [
        'inc' => 1000,
        'ex' => 833.33,
    ],
],
];

public function setup()
{
    $this->loadViewsFrom(__DIR__.'/../resources/views', 'acme');

    ProductPage::extend(function (ProductPage $page) {
        $page->setData('extra_prices', $this->prices);

        ContentForBody::extend(function (&$content) {
            $content .= view('acme::javascript');
        });
    });

    CartItemBuilder::extend(function (CartItem $item) {
        collect(request()->input('extras', []))
            ->map(function ($extra) {
                return collect($this->prices)->firstWhere('id', $extra);
            })
            ->filter()
            ->each(function ($extra) use ($item) {
                $option = CartItemOption::create($extra['name']);
                $option->setPriceInc($extra['price']['inc']);
            });
    });
}

```

```
        $item->addOption($option);
    });
});
}
}
```