

# **Stripe Checkout Pages Using PHP**

## **Using Stripe JS and AJAX**

### **Bonus: The Name Your Price Example**

**With Lon Hosford**

**© 2016 Alonzo Hosford**

Copyright 2016 Alonzo L. Hosford. All Rights Reserved. [www.lonhosford.com](http://www.lonhosford.com)

This is a Visual Step by Step Workbook and voice transcript for accompanying video for this portion of the course.

# Bonus: The Name Your Price Example

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:



In this example you are creating a user interface that lets the user provide the amount. Somewhat like the advertising slogan for the US company PriceLine.com.


# Bonus: The Name Your Price Example

**Stripe Checkout Embedded Simple Using Ajax**

**Name Your Price Example**

Item: Widget Mystery Box

Amount:



Its not likely you will let your customers pick their price.  
More likely you might find this example a useful template for gifting or donation forms.

# Bonus: The Name Your Price Example

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

- Other
- \$10
- \$25**
- \$50
- ✓ \$100
- \$500
- \$1,000
- \$10,000

|             |
|-------------|
| Other       |
| \$10        |
| <b>\$25</b> |
| \$50        |
| ✓ \$100     |
| \$500       |
| \$1,000     |
| \$10,000    |



Your example will provide predefined values for selection.  
It will use a drop down menu for this.

# Bonus: The Name Your Price Example

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  Other ▾



You might allow the user to enter an amount that is not on the drop down menu. But that means that you need to check that the value is a valid amount.

# The Default Preset Amount

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descripton = "ACME-WIDGETS";
6 // Order Data
7 $description = "Widgets";
8 $quantity = 12;
9 $statement_descriptor = $bank_statement_descripton . ' ' . $quantity . ' ' .
    $description;
```

Open the checkout\_simple.php file in an editor.  
You are starting with the same Stripe code used for this section's example.

Snippet #1

# The Default Preset Amount

BEFORE

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets, Inc.";
5 $ba
6 //
7 $de
8 $qu
9 $st
```

**Snippet Guides!**

Copy and paste.

Open supplemental/snippets.txt file

```
6 // Preset amount.
7 $preset_amount = 50;
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
11 $statement_descriptor = $bank_statement_descriptor . ' ' . $quantity . ' ' .
    $description;
12 $amount = 20.00;
13 ?>
14 <!doctype html>
15 <html>
```

Snippets are available to copy and paste if you want to follow along without typing. Look for the guides in the top left corner.

**BEFORE**

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descripton = "ACME-WIDGETS";
6 // Order Data
7 $description = "Widgets";
8 $quantity = 12;
9 $statement_descriptor = $bank_statement_descripton . ' ' . $quantity . ' ' .
    $description;
```

**AFTER**

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descripton = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
11 $statement_descriptor = $bank_statement_descripton . ' ' . $quantity . ' ' .
    $description;
12 $amount = 20.00;
13 ?>
14 <!doctype html>
15 <html>
```

This line sets a preset value in PHP.  
You use this for a default preset amount when the form loads.



# The Preset URL Amount

BEFORE

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $ba
6 //
7 $pr
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
```

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)

AFTER

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descripton = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
11 $statement_descriptor = $bank_statement_descripton . ' ' . $quantity . ' ' .
    $description;
12 $amount = 20.00;
13 ?>
14 <!doctype html>
15 <html>
```

That preset amount will also have an override from the from the URL line.  
The URL parameter is named amount.

## BEFORE

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $ba
6 //
7 $pr
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
```

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)

## AFTER

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descriptor = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])){
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widgets";
13 $quantity = 12;
14 $statement_descriptor = $bank_statement_descriptor . ' ' . $quantity . ' ' .
    $description;
15 $amount = 20.00;
```

URL parameters appear as a key entry in the PHP `$_GET` super global variable. Then using the `isset` PHP function you can check if the URL amount parameter is present.

# The Preset URL Amount

BEFORE

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $ba
6 //
7 $pr
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
```

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)

AFTER

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descriptor = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])){
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widgets";
13 $quantity = 12;
14 $statement_descriptor = $bank_statement_descriptor . ' ' . $quantity . ' ' .
    $description;
15 $amount = 20.00;
```

If it is set, then your code can drop into an if block where you use it. In this case you will assign it to the `$preset_amount` variable value.

# The Preset URL Amount

BEFORE

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $ba
6 //
7 $pr
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
```

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)

AFTER

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descriptor = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])){
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widgets";
13 $quantity = 12;
14 $statement_descriptor = $bank_statement_descriptor . ' ' . $quantity . ' ' .
    $description;
15 $amount = 20.00;
```

It is also a good practice to sanitize the value to assure that it is a valid number.

The PHP `preg_replace` function can use a regular expression that culls out all the characters that do not represent a number.

# The Preset URL Amount

BEFORE

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $ba
6 //
7 $pr
8 // Order Data
9 $description = "Widgets";
10 $quantity = 12;
```

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)

AFTER

|             |
|-------------|
| Other       |
| \$10        |
| <b>\$25</b> |
| \$50        |
| ✓ \$100     |
| \$500       |
| \$1,000     |
| \$10,000    |

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descripton = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])){
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widgets";
13 $quantity = 12;
14 $statement_descriptor = $bank_statement_descripton . ' ' . $quantity . ' ' .
    $description;
15 $amount = 20.00;
```

You could also check to see if the amount is one of your preset values. We are skipping that for this example. You can add it for your own project practice.

## BEFORE

```
1 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widgets";
13 $quantity = 12;
14 $statement_descriptor = $bank_statement_descriptor . ' ' . $quantity . ' ' .
    $description;
15 $amount = 20.00;
```

## AFTER

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descriptor = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widget Mystery Box";
13 $quantity = 12;
14 $statement_descriptor = $bank_statement_descriptor . ' ' . $quantity . ' ' .
    $description;
15 $amount = 20.00;
```

Now for a small change to the description.

## BEFORE

```
1 $preset_amount = 50;
2
3
4
5
6
7
8 if (isset($_GET['amount'])) {
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widgets";
13 $quantity = 12;
14 $statement_descriptor = $bank_statement_descriptor . ' ' . $quantity . ' ' .
    $description;
15 $amount = 20.00;
```

## AFTER

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descriptor = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
11 // Order Data
12 $description = "Widget Mystery Box";
13 $statement_descriptor = $bank_statement_descriptor . $description;
14 ?>
15 <!doctype html>
16 <html>
```

Then you can remove the quantity and amount PHP variables from our base section example. Also remove the \$quantity variable from the \$statement\_descriptor expression.

# Checkpoint I

checkout\_simple.php

```
1 <?php
2 include_once 'common.inc.php';
3 // General website data
4 $company_name = "Acme Widgets Inc.";
5 $bank_statement_descripton = "ACME-WIDGETS";
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9     $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amou
10 }
11 // Order Data
12 $description = "Widget Mystery Box";
13 $statement_descriptor = $bank_statement_descripton . $descriptio
14 ?>
15 <!doctype html>
16 <html>
17 <head>
18 <meta charset="UTF-8">
19 <meta name="viewport" content="width=device-width, initial-scale=1">
```



The checkpoint\_01 folder contains the PHP coding changes to this point. You can use it to compare your work for lines 1 to 14.



## BEFORE

```
25 <div id="page-container" class="center-container">
26   <header><h1 class="center-text">Stripe Checkout Embedded Simple Using
    Ajax</h1></header>
27   <div class="form-container center-container">
28     <p>Item: <?=$description?></p>
29     <p>Quantity: <?=$quantity?></p>
30     <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/
    >
31     <input type="hidden" id="company-name" value="<?=$company_name?>"/>
32     <input type="hidden" id="quantity" value="<?=$quantity?>">
```

## AFTER

```
25 <div id="page-container" class="center-container">
26   <header>
27     <h1 class="center-text">Stripe Checkout Embedded Simple Using Ajax</
    h1>
28     <h2 class="center-text">Name Your Price Example</h2>
29   </header>
30   <div class="form-container center-container">
31     <p>Item: <?=$description?></p>
32     <p>Quantity: <?=$quantity?></p>
33     <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/
    >
34     <input type="hidden" id="company-name" value="<?=$company_name?>"/>
35     <input type="hidden" id="quantity" value="<?=$quantity?>">
36     <input type="hidden" id="amount" value="<?=$amount?>">
37     <input type="hidden" id="description" value="<?=$description?>">
38     <input type="hidden" id="statement-descriptor" value="<?=$
```

Next add a subtitle at the end of the page head element.

This is a purely an arbitrary step to distinguish this example from others we have done.

# Remove the quantity elements

## BEFORE

```
30 <div class="form-container center-container">
31 <p>Item: <?=$description?></p>
32 <p>Quantity: <?=$quantity?></p>
33 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/
  >
34 <input type="hidden" id="company-name" value="<?=$company_name?>"/>
35 <input type="hidden" id="quantity" value="<?=$quantity?>">
36 <input type="hidden" id="amount" value="<?=$amount?>">
37 <input type="hidden" id="description" value="<?=$description?>">
38 <input type="hidden" id="statement-descriptor" value="<?=$
```

checkout\_simple.php

## AFTER

```
30 <div class="form-container center-container">
31 <p>Item: <?=$description?></p>
32 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/
  >
33 <input type="hidden" id="company-name" value="<?=$company_name?>"/>
34 <input type="hidden" id="amount" value="<?=$amount?>">
35 <input type="hidden" id="description" value="<?=$description?>">
36 <input type="hidden" id="statement-descriptor" value="<?=$
  statement_descriptor?>">
37 <p id="checkout-loading-message" class="checkout-message center-text"
  >
38 Loading ...
39 </p>
40 <script src="https://checkout.stripe.com/checkout.js"></script>
41 <p id="checkout-btn-container" class="center-text">
42 <a id="checkout-btn" href="#" class="pav-btn">Checkout</a>
```

Remove the form quantity description and also the quantity hidden element.  
No quantity is used in this example.

# Add user input fields container element

checkout\_simple.php

## BEFORE

```
30 <div class="form-container center-container">
31 <p>Item: <?=$description?></p>
32 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
33 <input type="hidden" id="company-name" value="<?=$company_name?>"/>
34 <input type="hidden" id="amount" value="<?=$amount?>">
35 <input type="hidden" id="description" value="<?=$description?>">
36 <input type="hidden" id="statement-descriptor" value="<?=$
    statement_descriptor?>">
37 <p id="checkout-loading-message" class="checkout-message center-text"
```

## AFTER

```
30 <div class="form-container center-container">
31 <div id="input-fields">
32 <p>Item: <?=$description?></p>
33 </div>
34 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
35 <input type="hidden" id="company-name" value="<?=$company_name?>"/>
36 <input type="hidden" id="amount" value="<?=$amount?>">
37 <input type="hidden" id="description" value="<?=$description?>">
38 <input type="hidden" id="statement-descriptor" value="<?=$
    statement_descriptor?>">
39 <p id="checkout-loading-message" class="checkout-message center-text"
40 >
41 Loading ...
42 </p>
43 <script src="https://checkout.stripe.com/checkout.js"></script>
```

Then add a container element for the user input fields.  
You can use this to hide or show all the user input fields as needed.  
For example after the checkout process begins.

# Add user input field for amount

checkout\_simple.php

## BEFORE

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33   </div>
34   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
35   <input type="hidden" id="company-name" value="<?=$company_name?>"/>
36   <input type="hidden" id="amount" value="<?=$amount?>">
37   <input type="hidden" id="description" value="<?=$description?>">
38   <input type="hidden" id="statement-descriptor" value="<?=$
```

## AFTER

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     Amount: <input type="text" id="amount"/>
34   </div>
35   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
36   <input type="hidden" id="company-name" value="<?=$company_name?>"/>
37   <input type="hidden" id="amount" value="<?=$amount?>">
38   <input type="hidden" id="description" value="<?=$description?>">
39   <input type="hidden" id="statement-descriptor" value="<?=$
    statement_descriptor?>">
40   <p id="checkout-loading-message" class="checkout-message center-text"
41     >
42     Loading ...
43   </p>
```

Following the description line add the input field for the user to enter the amount. The id amount is already used in the Javascript for the Stripe checkout code.

# Multiple amount element ids

## BEFORE

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33   </div>
34   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
35   <input type="hidden" id="company-name" value="<?=$company_name?>"/>
36   <input type="hidden" id="amount" value="<?=$amount?>">
37   <input type="hidden" id="description" value="<?=$description?>">
38   <input type="hidden" id="statement-descriptor" value="<?=$
```

checkout\_simple.php

## AFTER

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     Amount: <input type="text" id="amount"/>
34   </div>
35   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
36   <input type="hidden" id="company-name" value="<?=$company_name?>"/>
37   <input type="hidden" id="amount" value="<?=$amount?>">
38   <input type="hidden" id="description" value="<?=$description?>">
39   <input type="hidden" id="statement-descriptor" value="<?=$
    statement_descriptor?>">
40   <p id="checkout-loading-message" class="checkout-message center-text"
41     >
42     Loading ...
```

But you have a duplicate id amount attribute value.  
And it uses a PHP variable we no longer use.  
Duplicate id attribute values create bugs in Javascript.

## Set duplicate amount id to preset\_amount

checkout\_simple.php

## BEFORE

```
30 <div class="form-container center-container">
31 <div id="input-fields">
32 <p>Item: <?=$description?></p>
33 Amount: <input type="text" id="amount"/>
34 </div>
35 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
36 <input type="hidden" id="company-name" value="<?=$company_name?>"/>
37 <input type="hidden" id="amount" value="<?=$amount?>">
38 <input type="hidden" id="description" value="<?=$description?>">
```

## AFTER

```
30 <div class="form-container center-container">
31 <div id="input-fields">
32 <p>Item: <?=$description?></p>
33 Amount: <input type="text" id="amount"/>
34 </div>
35 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
36 <input type="hidden" id="company-name" value="<?=$company_name?>"/>
37 <input type="hidden" id="preset-amount" value="<?=$preset_amount?>">
38 <input type="hidden" id="description" value="<?=$description?>">
39 <input type="hidden" id="statement-descriptor" value="<?=$
    statement_descriptor?>">
40 <p id="checkout-loading-message" class="checkout-message center-text"
41 >
41 Loading ...
42 </p>
```

You can reuse this hidden element to make the preset\_amount available to Javascript. So change both the id and the PHP variable for that.

# Add the preset amount choices

checkout\_simple.php

## BEFORE

```
30 <div class="form-container center-container">
31 <div id="input-fields">
32 <p>Item: <?=$description?></p>
33 Amount: <input type="text" id="amount"/>
34 </div>
35 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
36 <input type="hidden" id="company-name" value="<?=$company_name?>"/>
37 <input type="hidden" id="preset-amount" value="<?=$preset_amount?>"/>
38 <input type="hidden" id="description" value="<?=$description?>"/>
```

## AFTER

```
30 <div class="form-container center-container">
31 <div id="input-fields">
32 <p>Item: <?=$description?></p>
33 Amount: <input type="text" id="amount"/>
34 <select id="preset-amounts">
35 <option value="other">Other</option>
36 <option value="10">$10</option>
37 <option value="25">$25</option>
38 <option value="50">$50</option>
39 <option value="100">$100</option>
40 <option value="500">$500</option>
41 <option value="1000">$1,000</option>
42 <option value="10000">$10,000</option>
43 </select>
44 </div>
45 <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
```

Next add a select element for the preset values.  
The id element is set for direct access in Javascript.

# Add the preset amount choices

## BEFORE

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     Amount: <input type="text" id="amount"/>
34   </div>
35   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
36   <input type="hidden" id="company-name" value="<?=$company_name?>"/>
37   <input type="hidden" id="preset-amount" value="<?=$preset_amount?>"/>
38   <input type="hidden" id="description" value="<?=$description?>"/>
```

checkout\_simple.php

## AFTER

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     Amount: <input type="text" id="amount"/>
34     <select id="preset-amounts">
35       <option value="other">Other</option>
36       <option value="10">$10</option>
37       <option value="25">$25</option>
38       <option value="50">$50</option>
39       <option value="100">$100</option>
40       <option value="500">$500</option>
41       <option value="1000">$1,000</option>
42       <option value="10000">$10,000</option>
43     </select>
44   </div>
45   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
```

All but the first option value are the actual preset amounts.

The first option value helps the user recognize that they can enter their own amount.



# Add the preset amount choices

## BEFORE

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     Amount: <input type="text" id="amount"/>
34   </div>
35   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
36   <input type="hidden" id="company-name" value="<?=$company_name?>"/>
37   <input type="hidden" id="preset-amount" value="<?=$preset_amount?>"/>
38   <input type="hidden" id="description" value="<?=$description?>"/>
```

checkout\_simple.php

## AFTER

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     Amount: <input type="text" id="amount"/>
34     <select id="preset-amounts">
35       <option value="other">Other</option>
36       <option value="10">$10</option>
37       <option value="25">$25</option>
38       <option value="50">$50</option>
39       <option value="100">$100</option>
40       <option value="500">$500</option>
41       <option value="1000">$1,000</option>
42       <option value="10000">$10,000</option>
43     </select>
44   </div>
45   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
```

The option content text aids the user in selecting the choices.

For example you might use words describing donation levels with the amount.

# Add p element container for user inputs

checkout\_simple.php

## BEFORE

```
30 <div class="form-container center-container">
31 <div id="input-fields">
32 <p>Item: <?=$description?></p>
33 Amount: <input type="text" id="amount"/>
34 <select id="preset-amounts">
35 <option value="500">$500</option>
41 <option value="1000">$1,000</option>
42 <option value="10000">$10,000</option>
43 </select>
44 </div>
```

## AFTER

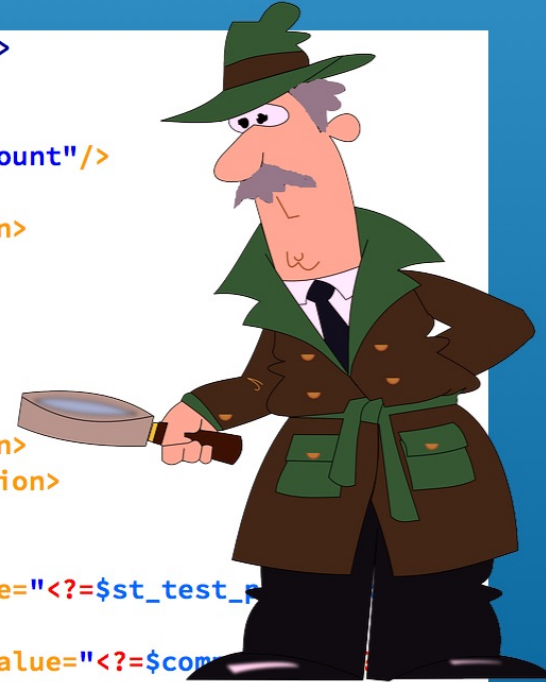
```
30 <div class="form-container center-container">
31 <div id="input-fields">
32 <p>Item: <?=$description?></p>
33 <p>Amount: <input type="text" id="amount"/>
34 <select id="preset-amounts">
35 <option value="other">Other</option>
36 <option value="10">$10</option>
37 <option value="25">$25</option>
38 <option value="50">$50</option>
39 <option value="100">$100</option>
40 <option value="500">$500</option>
41 <option value="1000">$1,000</option>
42 <option value="10000">$10,000</option>
43 </select></p>
44 </div>
45 <input type="hidden" id="stripe-pk" value="<?=$st test public key?>"/>
```

Then lets put both of these form elements into a p element.

# Checkpoint 2

checkout\_simple.php

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     <p>Amount: <input type="text" id="amount"/>
34     <select id="preset-amounts">
35       <option value="other">Other</option>
36       <option value="10">$10</option>
37       <option value="25">$25</option>
38       <option value="50">$50</option>
39       <option value="100">$100</option>
40       <option value="500">$500</option>
41       <option value="1000">$1,000</option>
42       <option value="10000">$10,000</option>
43     </select></p>
44   </div>
45   <input type="hidden" id="stripe-pk" value="<?=$st_test_pk">
46   >
47   <input type="hidden" id="company-name" value="<?=$company_name">
48   <input type="hidden" id="preset-amount" value="<?=$preset_amount?>">
```



Time to demo the changes.

The checkpoint\_02 folder contains the changes to this point.

You can use it to compare your work for lines 25 to 45.

# Check the form in the web browser

checkout\_simple.php

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  Other



Save the file and load into a web browser.  
Now you can use the new input fields.

# Check the form in the web browser

checkout\_simple.php

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  Other ▾

Checkout

|             |
|-------------|
| Other       |
| \$10        |
| <b>\$25</b> |
| \$50        |
| ✓ \$100     |
| \$500       |
| \$1,000     |
| \$10,000    |



At this point the select field does not update the amount input field.  
Javascript is needed to get that done.  
So lets work on it next.

## BEFORE

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Create a Stripe configuration object
8    */
9   var stripeConfig = {
10    key: $('#stripe-pk').val(),
```

checkout\_ui.js

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
11    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12    $('#amount').val(amount);
13    $('#amount').focus();
14  }
15  /**
16  * Create a Stripe configuration object
```

The first change to the Javascript file is to add a function that will set the input element.

# Add a function to set the amount input element

## BEFORE

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Create a Stripe configuration object
8    */
9   var stripeConfig = {
10    key: $('#stripe-pk').val(),
```

checkout\_ui.js

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
11    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12    $('#amount').val(amount);
13    $('#amount').focus();
14  }
15  /**
16   * Create a Stripe configuration object
```



The first line at first looks a bit awesome to dissect.

We can break it down.

Its job is to convert the function's amount argument to a two digit number.

# Add a function to set the amount input element

## BEFORE

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Create a Stripe configuration object
8    */
9   var stripeConfig = {
10    key: $('#stripe-pk').val(),
```

checkout\_ui.js

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
11    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12    $('#amount').val(amount);
13    $('#amount').focus();
14  }
15  /**
16  * Create a Stripe configuration object
```

First we are rounding the number to two decimal places. That uses the Javascript Math object's round method.



# Add a function to set the amount input element

## BEFORE

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Create a Stripe configuration object
8    */
9   var stripeConfig = {
10    key: $('#stripe-pk').val(),
```

checkout\_ui.js

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
11    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12    $('#amount').val(amount);
13    $('#amount').focus();
14  }
15  /**
16  * Create a Stripe configuration object
```

The rounded number is passed to the Javascript parseFloat function.

The parseFloat function converts a string to a float number which means decimals are allowed.

# Add a function to set the amount input element

## BEFORE

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Create a Stripe configuration object
8    */
9   var stripeConfig = {
10    key: $('#stripe-pk').val(),
```

checkout\_ui.js

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
11    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12    $('#amount').val(amount);
13    $('#amount').focus();
14  }
15  /**
16  * Create a Stripe configuration object
```

The toFixed method can be used on Javascript numbers.  
It gives you the formatting that would include trailing zero decimal values.

# Add a function to set the amount input element

## BEFORE

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Create a Stripe configuration object
8    */
9   var stripeConfig = {
10    key: $('#stripe-pk').val(),
```

checkout\_ui.js

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
11    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12    $('#amount').val(amount);
13    $('#amount').focus();
14  }
15  /**
16  * Create a Stripe configuration object
```

The next line in the function uses JQuery for setting the input value on the form. In the HTML, the element has the id named amount.

# Add a function to set the amount input element

## BEFORE

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Create a Stripe configuration object
8    */
9   var stripeConfig = {
10    key: $('#stripe-pk').val(),
```

checkout\_ui.js

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
11    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12    $('#amount').val(amount);
13    $('#amount').focus();
14  }
15  /**
16  * Create a Stripe configuration object
```

The last line gives the input element focus so the user can edit without the extra step of selecting it.

## BEFORE

```
8  * @param mixed amount
9  */
10 function setAmountTo(amount){
11     amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12     $('#amount').val(amount);
13     $('#amount').focus();
14 }
15 /**
16  * Create a Stripe configuration object
17  */
```

checkout\_ui.js

## AFTER

```
8  * @param mixed amount
9  */
10 function setAmountTo(amount){
11     amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
12     $('#amount').val(amount);
13     $('#amount').focus();
14 }
15 /**
16  * Set the #amount input form field and give it focus
17  */
18 function clearAmount(){
19     $('#amount').val('');
20     $('#amount').focus();
21 }
22 /**
23  * Create a Stripe configuration object
```

Next add the clearAmount function.

You can add it after the setAmountTo function.

It empties the amount input element on the form and gives it focus.

## BEFORE

```

15  /
16  *  Set the #amount input form field and give it focus
17  */
18  function clearAmount(){
19      $('#amount').val('');
20      $('#amount').focus();
21  }
22  /**
23  *  Create a Stripe configuration object
24  */

```

checkout\_ui.js

checkout\_simple.php

```

35  <select id="preset-amounts">
36      <option value="other">Other</option>
37      <option value="10">$10</option>
38      <option value="25">$25</option>
39      <option value="50">$50</option>
40      <option value="100">$100</option>
41      <option value="500">$500</option>
42      <option value="1000">$1,000</option>
43      <option value="10000">$10,000</option>
44  </select>

```

## AFTER

```

18  function clearAmount(){
19      $('#amount').val('');
20      $('#amount').focus();
21  }
22  /**
23  *  Create a Stripe configuration object
24  */
25  $( "#preset-amounts" ).change(function() {
26      var amount = $(this).val();
27      if(amount == 'other'){
28          clearAmount();
29      }else{
30          setAmountTo(amount);
31      }
32  });

```

After the clearAmount function add a JQuery change event handler for the preset-amounts select element. This triggers when the selected option changes.

# Add change handler for preset-amounts element

## BEFORE

```
15  /
16  *  Set the #amount input form field and give it focus
17  */
18  function clearAmount(){
19  ——— $('#amount').val('');
20  ——— $('#amount').focus();
21  }
22  /**
23  *  Create a Stripe configuration object
24  */
```

## AFTER

```
18  function clearAmount(){
19  ——— $('#amount').val('');
20  ——— $('#amount').focus();
21  }
22  /**
23  *  Create a Stripe configuration object
24  */
25  $( "#preset-amounts" ).change(function() {
26  ——— var amount = $(this).val();
27  ——— if(amount == 'other'){
28  ———   clearAmount();
29  ——— }else{
30  ———   setAmountTo(amount);
31  ——— }
32  ——— });
33  /**
```

checkout\_ui.js

checkout\_simple.php

```
35  <select id="preset-amounts">
36  ——— <option value="other">Other</option>
37  ——— <option value="10">$10</option>
38  ——— <option value="25">$25</option>
39  ——— <option value="50">$50</option>
40  ——— <option value="100">$100</option>
41  ——— <option value="500">$500</option>
42  ——— <option value="1000">$1,000</option>
43  ——— <option value="10000">$10,000</option>
44  </select>
```

The change handler extracts the selected items option element value attribute. The keyword this refers to the element triggering the event. It is the select element and we can call the JQuery val method to get the selected option.

# Add change handler for preset-amounts element

## BEFORE

```
15  /
16  *  Set the #amount input form field and give it focus
17  */
18  function clearAmount(){
19      $('#amount').val('');
20     $('#amount').focus();
21 }
22 /**
23  *  Create a Stripe configuration object
24  */
```

## AFTER

```
18  function clearAmount(){
19      $('#amount').val('');
20     $('#amount').focus();
21 }
22 /**
23  *  Create a Stripe configuration object
24  */
25  $( "#preset-amounts" ).change(function() {
26      var amount = $(this).val();
27     if(amount == 'other'){
28         clearAmount();
29     }else{
30         setAmountTo(amount);
31     }
32 });
```

checkout\_ui.js

checkout\_simple.php

```
35  <select id="preset-amounts">
36  <option value="other">Other</option>
37  <option value="10">$10</option>
38  <option value="25">$25</option>
39  <option value="50">$50</option>
40  <option value="100">$100</option>
41  <option value="500">$500</option>
42  <option value="1000">$1,000</option>
43  <option value="10000">$10,000</option>
44  </select>
```

If the 'other' option is selected, you call your clearAmount function to empty the input field.



# Add change handler for preset-amounts element

## BEFORE

```
15  /
16  *  Set the #amount input form field and give it focus
17  */
18  function clearAmount(){
19  _____ $('#amount').val('');
20  _____ $('#amount').focus();
21  }
22  /**
23  *  Create a Stripe configuration object
24  */
```

## AFTER

```
18  function clearAmount(){
19  _____ $('#amount').val('');
20  _____ $('#amount').focus();
21  }
22  /**
23  *  Create a Stripe configuration object
24  */
25  $( "#preset-amounts" ).change(function() {
26  _____ var amount = $(this).val();
27  _____ if(amount == 'other'){
28  _____ clearAmount();
29  _____ }else{
30  _____ setAmountTo(amount);
31  _____ }
32  _____ });
33  /**
```

checkout\_ui.js

checkout\_simple.php

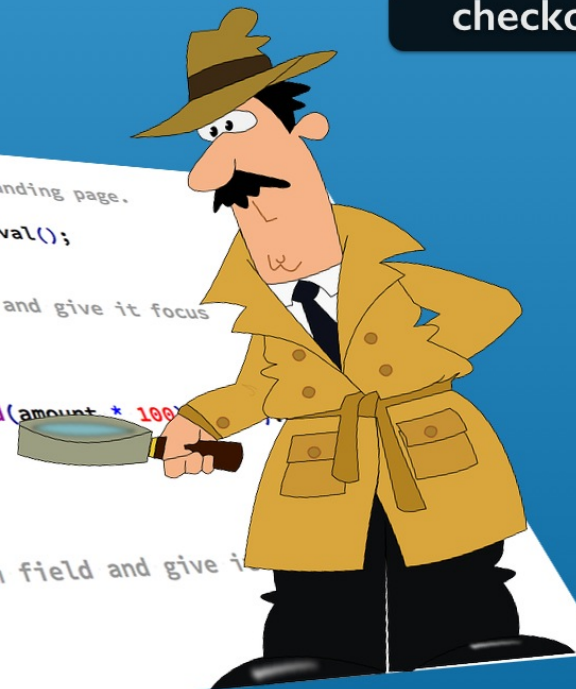
```
35  _____ <select id="preset-amounts">
36  _____ <option value="other">Other</option>
37  _____ <option value="10">$10</option>
38  _____ <option value="25">$25</option>
39  _____ <option value="50">$50</option>
40  _____ <option value="100">$100</option>
41  _____ <option value="500">$500</option>
42  _____ <option value="1000">$1,000</option>
43  _____ <option value="10000">$10,000</option>
44  _____ </select>
```

If not, you can populate the input field with the option value using the setAmountTo function we added.

# Checkpoint 3

checkout\_simple.php

```
6  /**
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the #amount input form field and give it focus
13 * @param mixed amount
14 */
15 function setAmountTo(amount){
16     amount = parseFloat(Math.round(amount * 100) / 100);
17     $('#amount').val(amount);
18     $('#amount').focus();
19 }
20 /**
21 * Set the #amount input form field and give it focus
22 */
23 function clearAmount(){
24     $('#amount').val('');
```



Save the file.

The checkpoint\_03 folder contains the changes to this point.

Now you are ready to test.

# Test changing the preset drop down menu

checkout\_simple.php

checkout Embedded Simple Using Ajaxpe Checkout Embedded Simple Using

**Name Your Price Example**

Item: Widget Mystery Box

Amount:

- ✓ Other
- \$10
- \$25
- \$50
- \$100**
- \$500
- \$1,000
- \$10,000

Item: Widget Mystery Box

Amount:



Reload the checkout\_simple.php file in the web browser.  
You should be able to set the input field to any currency choice in the drop down menu.

# Test changing the preset drop down menu

checkout\_simple.php

checkout Embedded Simple Using Ajaxipe Checkout Embedded Simple Using

**Name Your Price Example**

Item: Widget Mystery Box

Amount:

- Other
- \$10
- \$25
- \$50
- ✓ \$100
- \$500
- \$1,000
- \$10,000

**Name Your Price Example**

Item: Widget Mystery Box

Amount:

Other ▾



If you select the Other option, then the input amount field is cleared.  
In all cases the input focus moves to the Amount field.

# Set amount input field to preset amount

## BEFORE

checkout\_ui.js

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount){
```

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Harvest preset default amount from landing page.
8    */
9   var presetAmount = $('#preset-amount').val();
10  setAmountTo(presetAmount);
11  /**
12   * Set the #amount input form field and give it focus
13   * @param mixed amount
14   */
15  function setAmountTo(amount){
16    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
```

Next you will work on using the preset amount.  
Add these lines to the checkout\_ui.js file.

# Set the amount input field to static preset amount

## BEFORE

checkout\_ui.js

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input form field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount) {
```

checkout\_simple.php

```
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9   $preset_amount = preg_replace('/^[^0-9\.\-]+/', '', $_GET['amount']);
10 }
```

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Harvest preset default amount from landing page.
8    */
9   var presetAmount = $('#preset-amount').val();
10  setAmountTo(presetAmount);
11  /**
12   * Set the #amount input form field and give it focus
13   * @param mixed amount
14   */
15  function setAmountTo(amount) {
16    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
```

Recall that the checkout\_simple.php file has the PHP \$preset\_amount variable. This is set as a static value or from the URL line.

# Set the amount input field to static preset amount

## BEFORE

checkout\_ui.js

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount) {
```

checkout\_simple.php

```
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9   $preset_amount = preg_replace('/[^0-9\.\-]+/', '', $_GET['amount']);
10 }
37 <input type="hidden" id="preset-amount" value="<?=$preset_amount?>">
```

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Harvest preset default amount from landing page.
8    */
9   var presetAmount = $('#preset-amount').val();
10  setAmountTo(presetAmount);
11  /**
12   * Set the #amount input form field and give it focus
13   * @param mixed amount
14   */
15  function setAmountTo(amount) {
16    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
```

Then in the HTML we make the \$preset\_amount PHP variable available to the DOM for access in Javascript.

# Set the amount input field to static preset amount

BEFORE

checkout\_ui.js

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7   * Set the #amount input field and give it focus
8   * @param mixed amount
9   */
10  function setAmountTo(amount) {
```

checkout\_simple.php

```
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9   $preset_amount = preg_replace('/[^0-9\.\-]+/', '', $_GET['amount']);
10 }
37 <input type="hidden" id="preset-amount" value="<?=$preset_amount?>">
```

AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7   * Harvest preset default amount from landing page.
8   */
9   var presetAmount = $('#preset-amount').val();
10  setAmountTo(presetAmount);
11  /**
12  * Set the #amount input form field and give it focus
13  * @param mixed amount
14  */
15  function setAmountTo(amount) {
16    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
```

The first line that you added pulls that value into the Javascript presetAmount variable



# Set the amount input field to static preset amount

BEFORE

checkout\_ui.js

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Set the #amount input field and give it focus
8    * @param mixed amount
9    */
10  function setAmountTo(amount) {
```

checkout\_simple.php

```
6 // Preset amount.
7 $preset_amount = 50;
8 if (isset($_GET['amount'])) {
9   $preset_amount = preg_replace('/[^0-9\.\-]+/', '', $_GET['amount']);
10 }
37 <input type="hidden" id="preset-amount" value="<?=$preset_amount?>">
```

AFTER

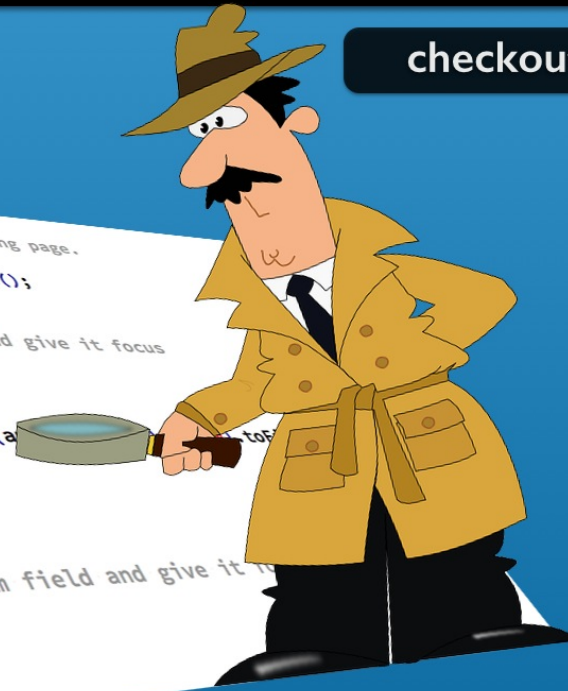
```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Harvest preset default amount from landing page.
8    */
9   var presetAmount = $('#preset-amount').val();
10  setAmountTo(presetAmount);
11  /**
12   * Set the #amount input form field and give it focus
13   * @param mixed amount
14   */
15  function setAmountTo(amount) {
16    amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);
```

Then you can call the setAmountTo function to initialize the amount input field.

# Checkpoint 4

checkout\_simple.php

```
6  /**
7  * Harvest preset default amount from landing page.
8  */
9
10 var presetAmount = $('#preset-amount').val();
11 setAmountTo(presetAmount);
12 /**
13 * Set the #amount input form field and give it focus
14 * @param mixed amount
15 */
15 function setAmountTo(amount){
16     amount = parseFloat(Math.round(a
17     $('#amount').val(amount);
18     $('#amount').focus();
19 }
20 /**
21 * Set the #amount input form field and give it
22 */
22 function clearAmount(){
23     $('#amount').val('');
24
```



Save the file.

The checkpoint\_04 folder contains the changes to this point.

Then you can begin testing your work.

# Test presetting the amount input field

checkout\_simple.php

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  Other ▾



Reload in the web browser.  
The default value of 50 dollars will appear.  
Also the amount field gets the input focus.

# Test presetting the amount input field

checkout\_simple.php

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  Other ▾

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)



Now add to the URL line the amount parameter equal to 100.  
The reload the web browser.  
The amount field gets pre-filled and focus again.

# Test presetting the amount input field

checkout\_simple.php

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)



You should notice the preset drop down menu could be set when there is a match with the preset.

# Test presetting the amount input field

checkout\_simple.php

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

|   |         |
|---|---------|
| Item: Widget Mystery Box                    |         |
| Amount: <input type="text" value="100.00"/> | Other ▾ |
| <input type="button" value="Checkout"/>     |         |

- ✓ Other
- \$10
- \$25
- \$50
- \$100**
- \$500
- \$1,000
- \$10,000

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)



If there is a match you can set to that value.

# Test presetting the amount input field

checkout\_simple.php

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  Other ▾

Checkout

|              |
|--------------|
| ✓ Other      |
| \$10         |
| \$25         |
| \$50         |
| <b>\$100</b> |
| \$500        |
| \$1,000      |
| \$10,000     |

|          |
|----------|
| ✓ Other  |
| \$10     |
| \$25     |
| \$50     |
| \$100    |
| \$500    |
| \$1,000  |
| \$10,000 |

[https://your\\_domain/checkout\\_simple.php?amount=123](https://your_domain/checkout_simple.php?amount=123)



If there is not a match, you can show the other value.  
So lets add the coding to get this done.

## BEFORE

checkout\_ui.js

```
7  /* Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12  * Set the #amount input form field and give it focus
13  * @param mixed amount
14  */
15 function setAmountTo(amount){
```

## AFTER

```
7  /* Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12  * Set the option for the #amounts select element
13  * @param mixed amount
14  */
15 function setAmountsTo(amount){
16     if($("#preset-amounts option[value='" + amount + "']").length == 1){
17         $("#preset-amounts").val(amount);
18     }else{
19         $("#preset-amounts").val('other');
20     }
21 }
22 /**
```

Here is the function that you can use.

Add after the call to the setAmountTo function where we absorb the preset amount from the form.

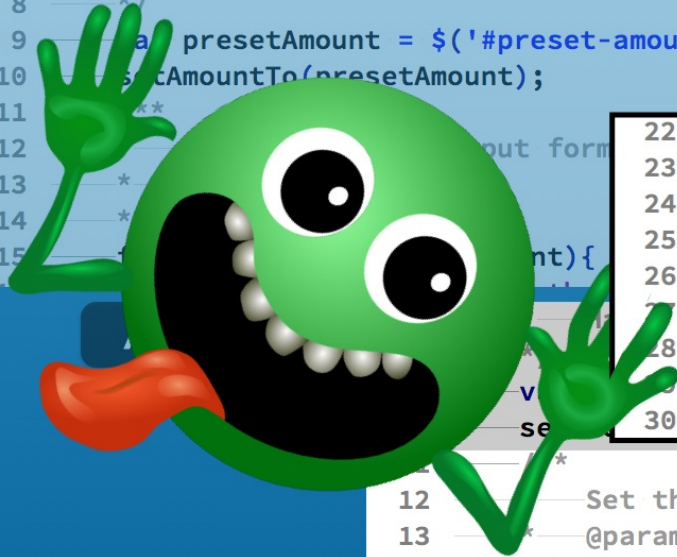


# Add a function to set preset-amounts element

BEFORE

checkout\_ui.js

```
7  * Harvest preset default amount from landing page.  
8  */  
9  presetAmount = $('#preset-amount').val();  
10 setAmountTo(presetAmount);  
11  
12  
13  
14  
15
```



```
22  /**  
23  * Set the #amount input form field and give it focus  
24  * @param mixed amount  
25  */  
26  function setAmountTo(amount){  
27      amount = parseFloat(Math.round(amount * 100) / 100).toFixed(2);  
28      $('#amount').val(amount);  
29      $('#amount').focus();  
30  }
```

```
12  Set the option for the #amounts select element  
13  * @param mixed amount  
14  */  
15  function setAmountsTo(amount){  
16      if($("#preset-amounts option[value='" + amount + "']").length == 1){  
17          $("#preset-amounts").val(amount);  
18      }else{  
19          $("#preset-amounts").val('other');  
20      }  
21  }  
22  /**
```

Careful we have two functions with almost the same name.  
One is setAmountTo and the new one is named setAmountsTo.

# Add a function to set preset-amounts element

## BEFORE

checkout\_ui.js

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the #amount input form field and give it focus
13 * @param mixed amount
14 */
15 function setAmountTo(amount){
```

## AFTER

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the option for the #amounts select element
13 * @param mixed amount
14 */
15 function setAmountsTo(amount){
16     if($("#preset-amounts option[value='" + amount + "']").length == 1){
17         $("#preset-amounts").val(amount);
18     }else{
19         $("#preset-amounts").val('other');
20     }
21 }
22 /**
```

The first line uses JQuery to select the option element with a value attribute equal to the function's amount argument.

# Add a function to set preset-amounts element

## BEFORE

checkout\_ui.js

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the #amount input form field and give it focus
13 * @param mixed amount
14 */
15 function setAmountTo(amount){
```

## AFTER

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the option for the #amounts select element
13 * @param mixed amount
14 */
15 function setAmountsTo(amount){
16     if($("#preset-amounts option[value='" + amount + "']").length == 1){
17         $("#preset-amounts").val(amount);
18     }else{
19         $("#preset-amounts").val('other');
20     }
21 }
22 /**
```

Then the length property tells you if JQuery matched a selection.

# Add a function to set preset-amounts element

## BEFORE

checkout\_ui.js

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the #amount input form field and give it focus
13 * @param mixed amount
14 */
15 function setAmountTo(amount){
```

## AFTER

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the option for the #amounts select element
13 * @param mixed amount
14 */
15 function setAmountsTo(amount){
16     if($("#preset-amounts option[value='" + amount + "']").length == 1){
17         $("#preset-amounts").val(amount);
18     }else{
19         $("#preset-amounts").val('other');
20     }
21 }
22 /**
```

For a match, you use the amount in selected element's val method. That will show the amount in the drop down menu on the form.

# Add a function to set preset-amounts element

## BEFORE

checkout\_ui.js

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the #amount input form field and give it focus
13 * @param mixed amount
14 */
15 function setAmountTo(amount){
```

## AFTER

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the option for the #amounts select element
13 * @param mixed amount
14 */
15 function setAmountsTo(amount){
16     if($("#preset-amounts option[value='" + amount + "']").length == 1){
17         $("#preset-amounts").val(amount);
18     }else{
19         $("#preset-amounts").val('other');
20     }
21 }
22 /**
```

For all other cases, the select element is set to 'other'.  
That will display the other option in the drop down menu.  
Now you can just call this function using the form's preset value.

## BEFORE

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 /**
12 * Set the option for the #amounts select element
13 * @param mixed amount
14 */
15 function setAmountsTo(amount){
16     if($("#preset-amounts option[value='" + amount + "']").length == 1){
```

## AFTER

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 setAmountsTo(presetAmount);
12 /**
13 * Set the option for the #amounts select element
14 * @param mixed amount
15 */
16 function setAmountsTo(amount){
17     if($("#preset-amounts option[value='" + amount + "']").length == 1){
18         $("#preset-amounts").val(amount);
19     }else{
20         $("#preset-amounts").val('other');
21     }
22 }
```

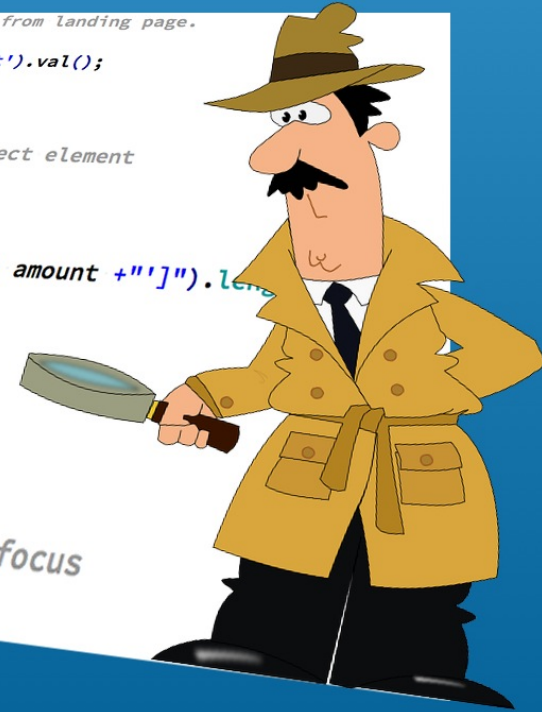
Add a line after the setAmountTo.

Insert the setAmountsTo function and use the same presetAmount variable.

# Checkpoint 5

checkout\_simple.php

```
7  * Harvest preset default amount from landing page.
8  */
9  var presetAmount = $('#preset-amount').val();
10 setAmountTo(presetAmount);
11 setAmountsTo(presetAmount);
12 /**
13  * Set the option for the #amounts select element
14  * @param mixed amount
15  */
16 function setAmountsTo(amount){
17     if($("#preset-amounts option[value='" + amount + "']").length)
18         $("#preset-amounts").val(amount);
19     }else{
20         $("#preset-amounts").val('other');
21     }
22 }
23 /**
24  * Set the #amount input form field and give it focus
25  * @param mixed amount
```



Save the file.  
The checkpoint\_05 folder contains the changes to this point.  
On to testing.

# Test Drop Down Menu Preset on Load

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



Reload in the web browser without any URL line query.  
The expected result is 50 dollars selected in the drop down menu.  
The amount input field has that amount with two decimals and is the input focus.



# Test Drop Down Menu Preset on Load

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

[https://your\\_domain/checkout\\_simple.php?amount=100](https://your_domain/checkout_simple.php?amount=100)



Add the amount equal to 100 on the URL line.

The drop down menu is expected to show 100 dollars selected.

The amount input field has focus with the 100 value formatted with two decimals

# Test Drop Down Menu Preset on Load

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  Other ▾

[https://your\\_domain/checkout\\_simple.php?amount=123.45](https://your_domain/checkout_simple.php?amount=123.45)



Try a URL command line amount that is not in the drop down menu.  
The value shows in the amount input field.  
The drop down menu will show other.

# Test Drop Down Menu Preset on Load

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

[https://your\\_domain/checkout\\_simple.php?amount=123.4](https://your_domain/checkout_simple.php?amount=123.4)



Lets make sure that the decimal formatting works if only one decimal is supplied on the URL line. For example you could use 123.4 to test that.

# Test Drop Down Menu Preset on Load

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

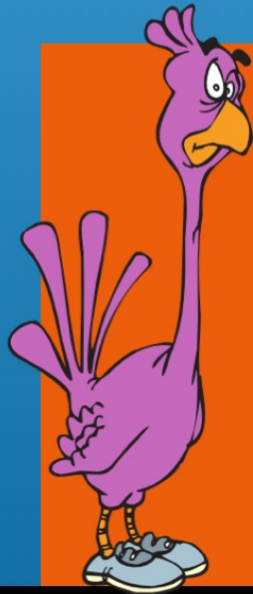
Amount:   ▾

[https://your\\_domain/checkout\\_simple.php?amount=123.455](https://your_domain/checkout_simple.php?amount=123.455)



And you could use 123.455 to test more than two decimals and rounding.

# Test Drop Down Menu Preset on Load



## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

|   |  |
|---|--|
| Item: Widget Mystery Box                    |  |
| Amount: <input type="text" value="123.46"/> | <input type="button" value="Other"/> ▾ |
| <input type="button" value="Checkout"/>     |  |

[https://your\\_domain/checkout\\_simple.php?amount=123.455](https://your_domain/checkout_simple.php?amount=123.455)

Normally you and not the user would use the URL line to preset the form value. You would create links from other pages, email campaigns or in social media posts for example.

# Dangers of User Input

checkout\_ui.js

```
99  ——/**
100 —— *—— Create a Stripe form handler object
101 —— */
102 —— var stripeFormHandler = StripeCheckout.configure(stripeConfig);
103 —— /**
104 —— *—— Click event for checkout
105 —— */
106 —— $('#checkout-btn').click(function(e){
107 ——   console.log("#checkout-btn");
108 ——   // Open Stripe checkout form
109 ——   stripeFormHandler.open({
110 ——     amount: $('#amount').val() * 100
111 ——   });
112 ——   e.preventDefault();
113 —— });
114 —— /**
115 —— *—— Close Checkout on page navigation
116 —— */
117 —— $(window).on('popstate', function() {
```

Now on to moving the amount into the Stripe checkout process.  
This is the checkout button click handler.

# Dangers of User Input

checkout\_ui.js

```
99  /**
100 *   Create a Stripe form handler object
101 */
102 var stripeFormHandler = StripeCheckout.configure(stripeConfig);
103 /**
104 *   Click event for checkout
105 */
106 $('#checkout-btn').click(function(e){
107   console.log("#checkout-btn");
108   // Open Stripe checkout form
109   stripeFormHandler.open({
110     amount: $('#amount').val() * 100
111   });
112   e.preventDefault();
113 });
114 /**
115 *   Close Checkout on page navigation
116 */
117 $(window).on('popstate', function() {
```



Item: Widget Mystery Box

Amount:

You are allowing the user to enter any value.  
That presents issues such as not entering numbers or a number too small.  
For example what if they enter 7 cents.

# Adding a number validation function

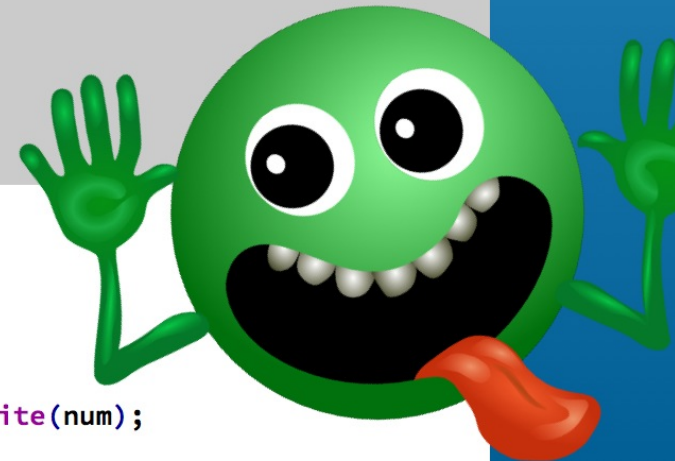
checkout\_ui.js

## BEFORE

```
106 —— $('#checkout-btn').click(function(e){
107 —— console.log($"('#checkout-btn')");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Close Checkout on page navigation
```

## AFTER

```
107 —— console.log($"('#checkout-btn')");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Validate is a number
116 —— * @param mixed num value to evaluate
117 —— * @return boolean
118 —— */
119 —— function isValidNumber(num){
120 —— return !isNaN(parseFloat(num)) && isFinite(num);
121 —— }
122 —— /**
```



First add a number validation function.  
It does look a bit complicated.



# Adding a number validation function

## BEFORE

```
106 —— $('#checkout-btn').click(function(e){
107 —— console.log($"('#checkout-btn')");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Close Checkout on page navigation
```

## checkout\_ui.js

## AFTER

```
107 —— console.log($"('#checkout-btn')");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Validate is a number
116 —— * @param mixed num value to evaluate
117 —— * @return boolean
118 —— */
119 —— function isValidNumber(num){
120 —— return !isNaN(parseFloat(num)) && isFinite(num);
121 —— }
122 —— /**
```

First the argument value is supplied to the Javascript parseFloat function. It will return a floating point number or the constant value NaN (not a number).

# Adding a number validation function

## BEFORE

```
106 —— $('#checkout-btn').click(function(e){
107 —— console.log("$.#checkout-btn");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Close Checkout on page navigation
```

## checkout\_ui.js

## AFTER

```
107 —— console.log("$.#checkout-btn");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Validate is a number
116 —— * @param mixed num value to evaluate
117 —— * @return boolean
118 —— */
119 —— function isValidNumber(num){
120 —— return !isNaN(parseFloat(num)) && isFinite(num);
121 —— }
122 —— /**
```

The parseFloat return value is passed to the Javascript isNaN function. This will detect if the parseFloat function returned a NaN value. The inverse of the isNaN function becomes the return value.

# Adding a number validation function

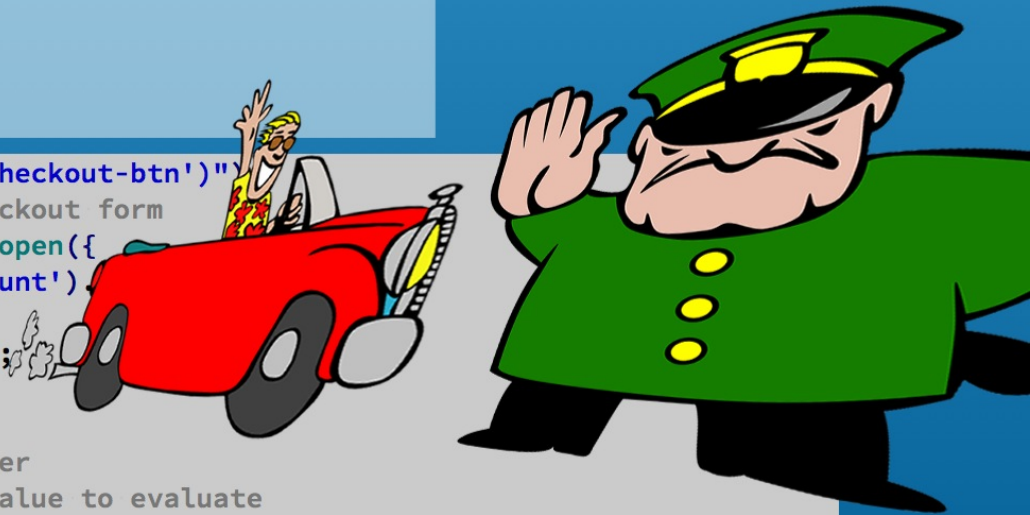
## BEFORE

```
106 ——— $('#checkout-btn').click(function(e){
107 ——— console.log("$.#checkout-btn");
108 ——— // Open Stripe checkout form
109 ——— stripeFormHandler.open({
110 ———   amount:$('#amount').val() * 100
111 ——— });
112 ——— e.preventDefault();
113 ——— });
114 ——— /**
115 ——— * Close Checkout on page navigation
```

## AFTER

```
107 ——— console.log("$.#checkout-btn");
108 ——— // Open Stripe checkout form
109 ——— stripeFormHandler.open({
110 ———   amount:$('#amount')
111 ——— });
112 ——— e.preventDefault();
113 ——— });
114 ——— /**
115 ——— * Validate is a number
116 ——— * @param mixed num value to evaluate
117 ——— * @return boolean
118 ——— */
119 ——— function isValidNumber(num){
120 ———   return !isNaN(parseFloat(num)) && isFinite(num);
121 ——— }
122 ——— /**
```

checkout\_ui.js



But not so fast!

One more test is added in the expression using the isFinite Javascript function.

# Adding a number validation function

## BEFORE

```
106 —— $('#checkout-btn').click(function(e){
107 —— console.log($"('#checkout-btn')");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Close Checkout on page navigation
```

checkout\_ui.js

## AFTER

```
107 —— console.log($"('#checkout-btn')");
108 —— // Open Stripe checkout form
109 —— stripeFormHandler.open({
110 —— amount:$('#amount').val() * 100
111 —— });
112 —— e.preventDefault();
113 —— });
114 —— /**
115 —— * Validate is a number
116 —— * @param mixed num value to evaluate
117 —— * @return boolean
118 —— */
119 —— function isValidNumber(num){
120 —— return !isNaN(parseFloat(num)) && isFinite(num);
121 —— }
122 —— /**
```



The isFinite function basically lets us know the number is one Javascript can use. Now you are good to go.

# Adding a minimum value.

## BEFORE

checkout\_ui.js

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Harvest preset default amount from landing page.
8    */
9   var presetAmount = $('#preset-amount').val();
10  setAmountTo(presetAmount);
```

## AFTER

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   /**
5    * The minimum amount
6    */
7   var minimumAmount = 5;
8   /**
9    * Preset form messaging fields.
10  */
11  $('#checkout-loading-message').hide();
12  $('#checkout-btn-container').show();
13  /**
14   * Harvest preset default amount from landing page.
15   */
16  var presetAmount = $('#preset-amount').val();
```

Next add a minimum number for the amount.

Place that at the top of the script so it is easy to find later if you want to change it.

# Adding a minimum value.

## BEFORE

checkout\_ui.js

```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   $('#checkout-loading-message').hide();
5   $('#checkout-btn-container').show();
6   /**
7    * Harvest preset default amount from landing page.
8    */
9   var presetAmount = $('#preset-amount').val();
10  setAmountTo(presetAmount);
```

## AFTER



```
1 // JQ shorthand for $(document).ready()
2 $(function() {
3   console.log("HELLO JQ");
4   /**
5    * The minimum amount
6    */
7   var minimumAmount = 5;
8   /**
9    * Preset form messaging fields.
10  */
11  $('#checkout-loading-message').hide();
12  $('#checkout-btn-container').show();
13  /**
14  * Harvest preset default amount from landing page.
15  */
16  var presetAmount = $('#preset-amount').val();
```

Keep in mind that validating values on the client side is only for the convenience of the user. It also helps reduce bad data going to your server.

You also need to check for data on the server and reject bad request values.

## BEFORE

```
112  /*/  
113  $('#checkout-btn').click(function(e){  
114  console.log($("#checkout-btn"));  
115  // Open Stripe checkout form  
116  stripeFormHandler.open({  
117  amount:$('#amount').val() * 100  
118  });  
119  e.preventDefault();  
120  });  
121  /**
```

checkout\_ui.js

## AFTER

```
110  /**  
111  * Click event for checkout  
112  */  
113  $('#checkout-btn').click(function(e){  
114  console.log($("#checkout-btn"));  
115  var amount = $( "#amount" ).val();  
116  console.log('Amount: ', amount);  
117  // Open Stripe checkout form  
118  stripeFormHandler.open({  
119  amount:amount * 100  
120  });  
121  e.preventDefault();  
122  });  
123  /**  
124  * Validate is a number  
125  * @param mixed num value to evaluate
```

The validation can be done in the checkout-btn element's click handler. Start with creating an amount variable from the input element. This way you can work with it before the Stripe form gets it.

## BEFORE

```
112  /*
113  * $('#checkout-btn').click(function(e){
114  *   console.log($('#checkout-btn'));
115  *   var amount = $( "#amount" ).val();
116  *   console.log('Amount: ', amount);
117  *   // Open Stripe checkout form
118  *   stripeFormHandler.open({
119  *     amount:amount * 100
120  *   });
121  *   e.preventDefault();
```

checkout\_ui.js

## AFTER

```
110  /**
111  *   * Click event for checkout
112  * */
113  * $('#checkout-btn').click(function(e){
114  *   console.log($('#checkout-btn'));
115  *   var amount = $( "#amount" ).val();
116  *   console.log('Amount: ', amount);
117  *   if (!isValidNumber(amount) || amount < minimumAmount){
118  *     // Update messaging on form
119  *   }else{
120  *     // Open Stripe checkout form
121  *     stripeFormHandler.open({
122  *       amount:amount * 100
123  *     });
124  *   }
125  *   e.preventDefault();
```

Next you can construct an if else code block to handle invalid and valid input amounts.



# Testing for a valid amount

## BEFORE

```
112  /*
113  *— $('#checkout-btn').click(function(e){
114  *—   console.log("$('#checkout-btn)");
115  *—   var amount = $( "#amount" ).val();
116  *—   console.log('Amount: ', amount);
117  *—   // Open Stripe checkout form
118  *—   stripeFormHandler.open({
119  *—     amount:amount * 100
120  *—   });
121  *—   e.preventDefault();
```

checkout\_ui.js

## AFTER

```
110  /**
111  *— Click event for checkout
112  */
113  *— $('#checkout-btn').click(function(e){
114  *—   console.log("$('#checkout-btn)");
115  *—   var amount = $( "#amount" ).val();
116  *—   console.log('Amount: ', amount);
117  *—   if (!isValidNumber(amount) || amount < minimumAmount){
118  *—     // Update messaging on form
119  *—   }else{
120  *—     // Open Stripe checkout form
121  *—     stripeFormHandler.open({
122  *—       amount:amount * 100
123  *—     });
124  *—   }
125  *—   e.preventDefault();
```

The first block handles an invalid input amount. It uses the `isValidNumber` function you added. If false, the block is executed.

# Testing for a valid amount

## BEFORE

```
112  /*
113  *— $('#checkout-btn').click(function(e){
114  *— console.log($('#checkout-btn'));
115  *— var amount = $( "#amount" ).val();
116  *— console.log('Amount: ', amount);
117  *— // Open Stripe checkout form
118  *— stripeFormHandler.open({
119  *—   amount:amount * 100
120  *— });
121  *— e.preventDefault();
```

checkout\_ui.js

## AFTER

```
110  /**
111  *— Click event for checkout
112  */
113  *— $('#checkout-btn').click(function(e){
114  *—   console.log($('#checkout-btn'));
115  *—   var amount = $( "#amount" ).val();
116  *—   console.log('Amount: ', amount);
117  *—   if (!isValidNumber(amount) || amount < minimumAmount){
118  *—     // Update messaging on form
119  *—   }else{
120  *—     // Open Stripe checkout form
121  *—     stripeFormHandler.open({
122  *—       amount:amount * 100
123  *—     });
124  *—   }
125  *—   e.preventDefault();
```

The other possibility of a bad input amount is being below the minimum value. The or operator adds that test by comparing to the minimumAmount variable added to the top of the script.

# Testing for a valid amount

## BEFORE

```
112  /*
113  $('#checkout-btn').click(function(e){
114  console.log("#$('#checkout-btn')");
115  var amount = $( "#amount" ).val();
116  console.log('Amount: ', amount);
117  // Open Stripe checkout form
118  stripeFormHandler.open({
119  amount:amount * 100
120  });
121  e.preventDefault();
```

checkout\_ui.js

## AFTER

```
110  /**
111  * Click event for checkout
112  */
113  $('#checkout-btn').click(function(e){
114  console.log("#$('#checkout-btn')");
115  var amount = $( "#amount" ).val();
116  console.log('Amount: ', amount);
117  if (!isValidNumber(amount) || amount < minimumAmount){
118  // Update messaging on form
119  }else{
120  // Open Stripe checkout form
121  stripeFormHandler.open({
122  amount:amount * 100
123  });
124  }
125  e.preventDefault();
```

If those two tests for an invalid amount prove false, the else code block is processed. Nothing new in the block yet.

# Checkpoint 6

checkout\_simple.php

```
110  /**
111  * Click event for checkout
112  */
113  $('#checkout-btn').click(function(e){
114      console.log("#checkout-btn");
115      var amount = $("#amount").val();
116      console.log('Amount: ', amount);
117      if (!isNaN(amount) || amount < minimumAmount){
118          // Update messaging on form
119      }else{
120          // Open Stripe checkout form
121          stripeFormHandler.open({
122              amount: amount * 100
123          });
124      }
125      e.preventDefault();
126  });
127  /**
128  * Validate is a number
129  * @param mixed num value to evaluate
```



Ready to test?

Save the file.

Changes are found in the checkpoint\_06 folder if you need to check your work.

# Test Detect Valid and Invalid Amounts

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

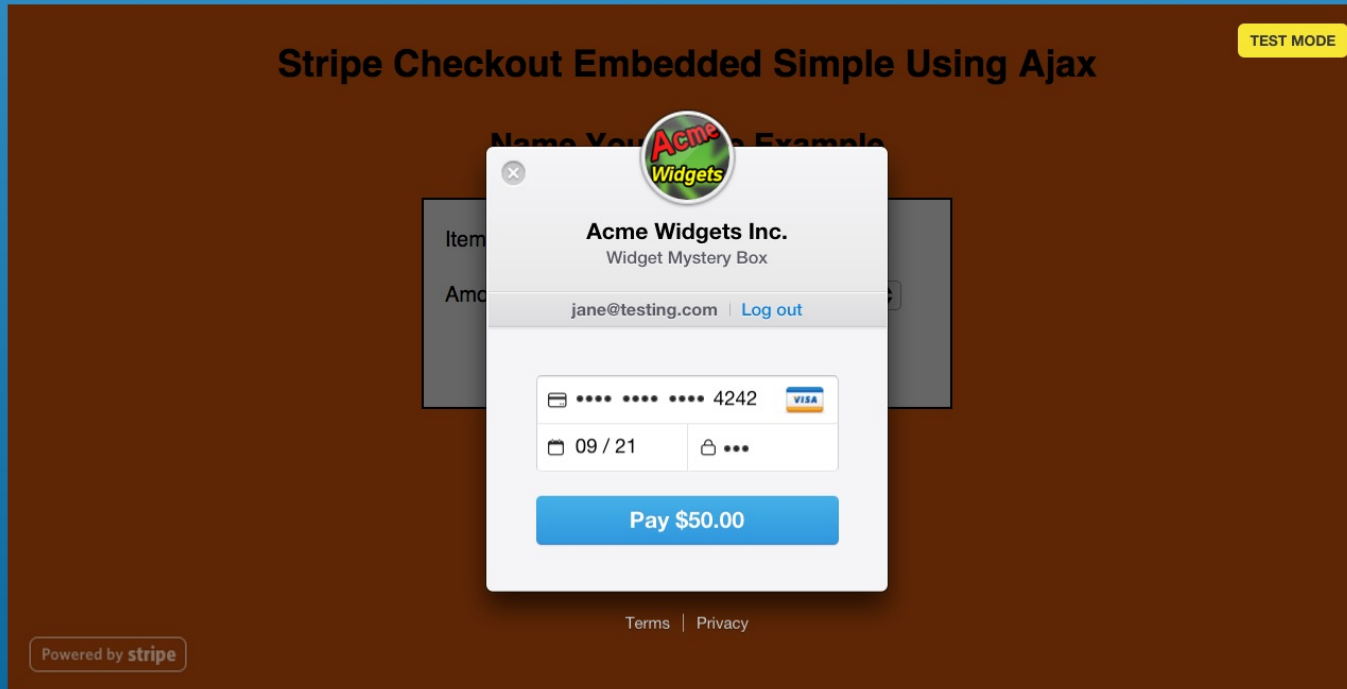
Amount:

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



Reload with a clean URL without the amount parameter.  
Click the Checkout button

# Test Detect Valid and Invalid Amounts



[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



And you will see the Checkout form.  
This screen has a Stripe ready test customer.  
You might need to create a test customer to speed up your testing.

# Test Detect Valid and Invalid Amounts

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

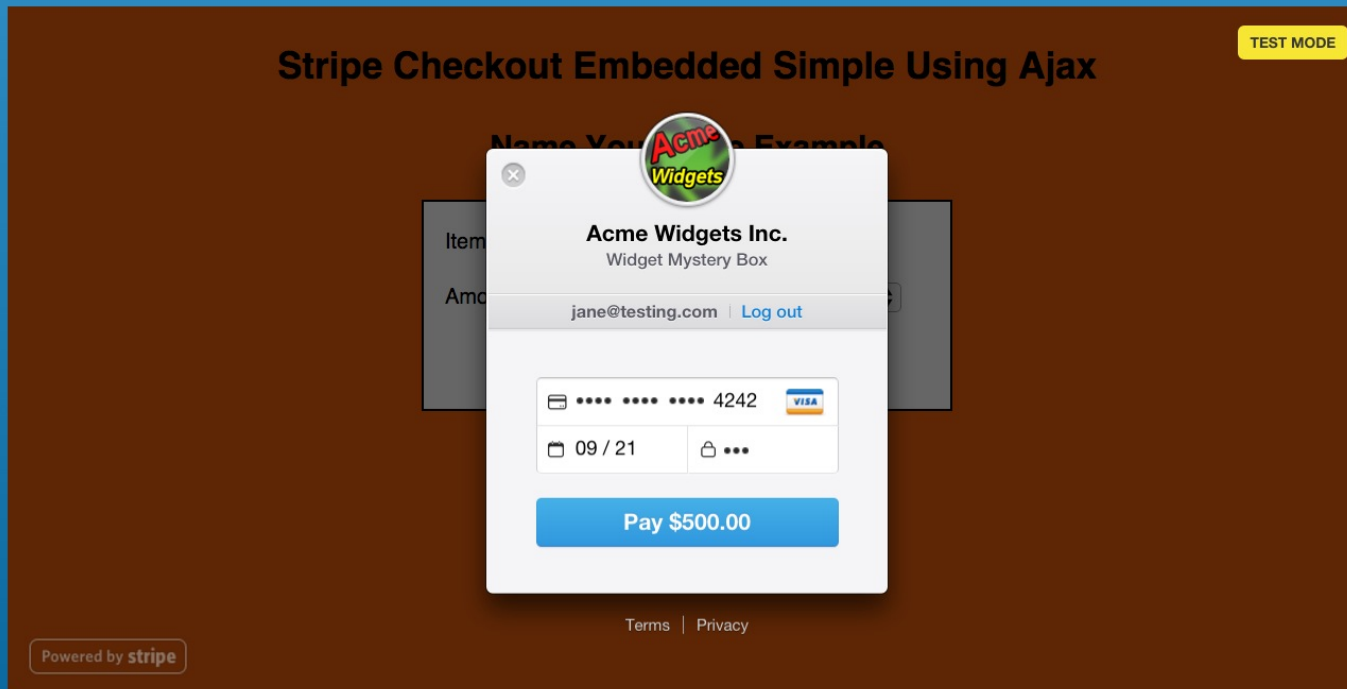
|   |              |
|---|--------------|
| Item: Widget Mystery Box                    | Other        |
| Amount: <input type="text" value="500.00"/> | \$10         |
| <input type="button" value="Checkout"/>     | \$25         |
|   | ✓ \$50       |
|   | \$100        |
|   | <b>\$500</b> |
|   | \$1,000      |
|   | \$10,000     |

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



Cancel and choose a different amount.

# Test Detect Valid and Invalid Amounts



[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



You should see amounts in the pay button.



# Test Detect Valid and Invalid Amounts

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:  \$500

Checkout

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



Cancel and enter anything that is not a pure float number.  
Clicking on the Checkout button will not open the Stripe checkout form.

# Test Detect Valid and Invalid Amounts

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)




Cancel and this time try a value below five dollars.  
No Stripe checkout form should appear if you try to checkout.

# Test Detect Valid and Invalid Amounts

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example



|   |                                    |
|---|------------------------------------|
| Item: Widget Mystery Box                |                                    |
| Amount: <input type="text" value="1"/>  | <input type="text" value="\$500"/> |
| <input type="button" value="Checkout"/> |                                    |

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)

You might want to provide user feedback why that happens.  
The messaging can be added to the HTML.  
So lets start there.

# Adding message for invalid amount

## BEFORE

```
60 <p id="checkout-success-message" class="checkout-message center-text"
61 >
62 Thank you for your order.
63 </p>
64 <p id="checkout-fail-message" class="checkout-message center-text
65 error-message-text">
66 Something Went Horribly Wrong!
67 </p>
68 </div>
69 </div>
```

checkout\_simple.php

## AFTER

```
59 </p>
60 <p id="checkout-success-message" class="checkout-message center-text"
61 >
62 Thank you for your order.
63 </p>
64 <p id="bad-amount-message" class="checkout-message center-text error-
65 message-text">
66 Input amount has an error.
67 </p>
68 <p id="checkout-fail-message" class="checkout-message center-text
69 error-message-text">
70 Something Went Horribly Wrong!
71 </p>
72 </div>
73 </div>
74 <script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
```

In the checkout\_simple.php file add these lines to display an error message and save.

# Adding message for invalid amount

## BEFORE

```
60 <p id="checkout-success-message" class="checkout-message center-text"
61 >
62 Thank you for your order.
63 </p>
64 <p id="checkout-fail-message" class="checkout-message center-text
65 error-message-text">
66 Something Went Horribly Wrong!
67 </p>
68 </div>
69 </div>
```

checkout\_simple.php

## AFTER

```
59 </p>
60 <p id="checkout-success-message" class="checkout-message center
61 >
62 Thank you for your order.
63 </p>
64 <p id="bad-amount-message" class="checkout-message
65 message-text">
66 Input amount has an error.
67 </p>
68 <p id="checkout-fail-message" class="checkout-messa
69 error-message-text">
70 Something Went Horribly Wrong!
71 </p>
72 </div>
73 </div>
74 <script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
```

base.css

```
72 /*--
73 Form message contain
74 --*/
75 .checkout-message{
76 display:none;
77 font-weight:bold;
78 }
79 /*--
80 Text style for error
```

The CSS checkout-message class hides the p element you added. You will show that p element with Javascript when an input error occurs.

# Adding message for invalid amount

## BEFORE

```
60 <p id="checkout-success-message" class="checkout-message center-text"
61 >
62 Thank you for your order.
63 </p>
64 <p id="checkout-fail-message" class="checkout-message center-text
65 error-message-text">
66 Something Went Horribly Wrong!
67 </p>
68 </div>
69 </div>
```

checkout\_simple.php

## AFTER

```
59 </p>
60 <p id="checkout-success-message" class="checkout-message center-text"
61 >
62 Thank you for your order.
63 </p>
64 <p id="bad-amount-message" class="checkout-message center-text error-
65 message-text">
66 Input amount has an error.
67 </p>
68 <p id="checkout-fail-message" class="checkout-message center-text
69 error-message-text">
70 Something Went Horribly Wrong!
71 </p>
72 </div>
73 </div>
74 <script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
```

You can use the id attribute that gives you direct access to the paragraph.

## BEFORE

```
113 ——— $('#checkout-btn').click(function(e){
114 ——— console.log("#checkout-btn");
115 ——— var amount = $( "#amount" ).val();
116 ——— console.log('Amount: ', amount);
117 ——— if (!isValidNumber(amount) || amount < minimumAmount){
118 ——— // Update messaging on form
119 ——— }else{
120 ——— // Open Stripe checkout form
121 ——— stripeFormHandler.open({
122 ——— amount:amount * 100
```

checkout\_ui.js

## AFTER

```
113 ——— $('#checkout-btn').click(function(e){
114 ——— console.log("#checkout-btn");
115 ——— var amount = $( "#amount" ).val();
116 ——— console.log('Amount: ', amount);
117 ——— if (!isValidNumber(amount) || amount < minimumAmount){
118 ——— if ($('#bad-amount-message').is(':visible')){
119 ——— $('#bad-amount-message').slideUp('slow',
120 ——— function(){
121 ——— $('#bad-amount-message').slideDown()
122 ——— });
123 ——— }else{
124 ——— $('#bad-amount-message').slideDown();
125 ——— }
126 ——— }else{
127 ——— // Open Stripe checkout form
128 ——— stripeFormHandler.open({
```



Now for the code to show this error message.

It may appear a bit fancy, but it is all JQuery and a Javascript if else block.

So lets break it down.

# Open and close invalid amount error message

## BEFORE

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($('#checkout-btn'));
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— // Update messaging on form
119 —— }else{
120 —— // Open Stripe checkout form
121 —— stripeFormHandler.open({
122 —— amount:amount * 100
```

checkout\_ui.js

## AFTER

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($('#checkout-btn'));
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— if ($('#bad-amount-message').is(':visible')){
119 —— $('#bad-amount-message').slideUp('slow',
120 —— function(){
121 —— $('#bad-amount-message').slideDown()
122 —— });
123 —— }else{
124 —— $('#bad-amount-message').slideDown();
125 —— }
126 —— }else{
127 —— // Open Stripe checkout form
128 —— stripeFormHandler.open({
```

The coding is all about the JQuery slideDown and slideUp animation methods. Each time an amount input error occurs the message slides open.



# Open and close invalid amount error message

BEFORE

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"#checkout-btn");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— // Update messaging on form
119 —— }else{
120 —— // Open Stripe checkout form
121 —— stripeFormHandler.open({
122 —— amount:amount * 100
```

checkout\_ui.js

AFT



```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"#checkout-btn");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— if ($('#bad-amount-message').is(':visible')){
119 —— $('#bad-amount-message').slideUp('slow',
120 —— function(){
121 —— $('#bad-amount-message').slideDown()
122 —— });
123 —— }else{
124 —— $('#bad-amount-message').slideDown();
125 —— }
126 —— }else{
127 —— // Open Stripe checkout form
128 —— stripeFormHandler.open({
```

If successive error attempts occur, you want it to slide close first and then back open. This helps alert the user the error is repeated.

# Open and close invalid amount error message

## BEFORE

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"#checkout-btn");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— // Update messaging on form
119 —— }else{
120 —— // Open Stripe checkout form
121 —— stripeFormHandler.open({
122 —— amount:amount * 100
```

checkout\_ui.js

## AFTER

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"#checkout-btn");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 ——     if ($('#bad-amount-message').is(':visible')){
119 ——         $('#bad-amount-message').slideUp('slow',
120 ——             function(){
121 ——                 $('#bad-amount-message').slideDown()
122 ——             });
123 ——     }else{
124 ——         $('#bad-amount-message').slideDown();
125 ——     }
126 —— }else{
127 —— // Open Stripe checkout form
128 —— stripeFormHandler.open({
```

The first line does that for you by detecting if the error message is visible.

# Adding message for invalid amount

## BEFORE

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"#checkout-btn");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— // Update messaging on form
119 —— }else{
120 —— // Open Stripe checkout form
121 —— stripeFormHandler.open({
122 —— amount:amount * 100
```

checkout\_ui.js

## AFTER

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"#checkout-btn");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— if ($('#bad-amount-message').is(':visible')){
119 —— $('#bad-amount-message').slideUp('slow',
120 —— function(){
121 —— $('#bad-amount-message').slideDown()
122 —— });
123 —— }else{
124 —— $('#bad-amount-message').slideDown();
125 —— }
126 —— }else{
127 —— // Open Stripe checkout form
128 —— stripeFormHandler.open({
```

The next line uses the `slideUp` method to animate the error message closing. The first argument to the `slideUp` method is duration. It takes millisecond values and a small list of speed keywords.

# Adding message for invalid amount

## BEFORE

```
113 ——— $('#checkout-btn').click(function(e){
114 ——— console.log($"#checkout-btn");
115 ——— var amount = $( "#amount" ).val();
116 ——— console.log('Amount: ', amount);
117 ——— if (!isValidNumber(amount) || amount < minimumAmount){
118 ——— // Update messaging on form
119 ——— }else{
120 ——— // Open Stripe checkout form
121 ——— stripeFormHandler.open({
122 ——— amount:amount * 100
```

checkout\_ui.js

## AFTER

```
113 ——— $('#checkout-btn').click(function(e){
114 ——— console.log($"#checkout-btn");
115 ——— var amount = $( "#amount" ).val();
116 ——— console.log('Amount: ', amount);
117 ——— if (!isValidNumber(amount) || amount < minimumAmount){
118 ——— if ($('#bad-amount-message').is(':visible')){
119 ——— $('#bad-amount-message').slideUp('slow',
120 ——— function(){
121 ——— $('#bad-amount-message').slideDown()
122 ——— });
123 ——— }else{
124 ——— $('#bad-amount-message').slideDown();
125 ——— }
126 ——— }else{
127 ——— // Open Stripe checkout form
128 ——— stripeFormHandler.open({
```

The second argument is a call back function that is executed when the animation finishes.

Both of these arguments are optional.

But we need to the call back to prevent other lines of code happening before the slideUp animation completes.

# Adding message for invalid amount

## BEFORE

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log("#'#checkout-btn'");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— // Update messaging on form
119 —— }else{
120 —— // Open Stripe checkout form
121 —— stripeFormHandler.open({
122 —— amount:amount * 100
```

checkout\_ui.js

## AFTER

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log("#'#checkout-btn'");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— if ($('#bad-amount-message').is(':visible')){
119 —— $('#bad-amount-message').slideUp('slow',
120 —— function(){
121 —— $('#bad-amount-message').slideDown()
122 —— });
123 —— }else{
124 —— $('#bad-amount-message').slideDown();
125 —— }
126 —— }else{
127 —— // Open Stripe checkout form
128 —— stripeFormHandler.open({
```

And that code is the slideUp animation method.  
No arguments are needed for the slideUp method.  
The default duration is 400 milliseconds.

# Adding message for invalid amount

## BEFORE

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($('#checkout-btn'));
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— // Update messaging on form
119 —— }else{
120 —— // Open Stripe checkout form
121 —— stripeFormHandler.open({
122 —— amount:amount * 100
```

checkout\_ui.js

## AFTER

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($('#checkout-btn'));
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— if ($('#bad-amount-message').is(':visible')){
119 —— $('#bad-amount-message').slideUp('slow',
120 —— function(){
121 —— $('#bad-amount-message').slideDown()
122 —— });
123 —— }else{
124 —— $('#bad-amount-message').slideDown();
125 —— }
126 —— }else{
127 —— // Open Stripe checkout form
128 —— stripeFormHandler.open({
```

If the error message is not visible, you only need to show it.  
So the else block calls the slideDown method without arguments.

# Adding message for invalid amount

BEFORE

```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"('#checkout-btn')");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— // Update messaging on form
119 —— }else{
120 —— // Open Stripe checkout form
121 —— stripeFormHandler.open({
122 —— amount:amount * 100
```

checkout\_ui.js

AFT



```
113 —— $('#checkout-btn').click(function(e){
114 —— console.log($"('#checkout-btn')");
115 —— var amount = $( "#amount" ).val();
116 —— console.log('Amount: ', amount);
117 —— if (!isValidNumber(amount) || amount < minimumAmount){
118 —— if ($('#bad-amount-message').is(':visible')){
119 —— $('#bad-amount-message').slideUp('slow',
120 —— function(){
121 —— $('#bad-amount-message').slideDown()
122 —— });
123 —— }else{
124 —— $('#bad-amount-message').slideDown();
125 —— }
126 —— }else{
127 —— // Open Stripe checkout form
128 —— stripeFormHandler.open({
```

Just remember that the slideDown method does not animate if the element is already showing. Same is true for all the JQuery methods that animate to a visible or hidden state.

## BEFORE

```
123 -----}else{
124 -----// Open Stripe checkout form
125 -----stripeFormHandler.open({
126 -----    amount:amount * 100
127 -----});
128 -----}
129 -----e.preventDefault();
130 -----});
131 -----/**
132 ----- * Validate is a number
```

## AFTER

```
126 -----}else{
127 -----    $('.checkout-message').slideUp();
128 -----// Open Stripe checkout form
129 -----    stripeFormHandler.open({
130 -----        amount:amount * 100
131 -----    });
132 -----}
133 -----e.preventDefault();
134 -----});
135 -----/**
136 ----- * Validate is a number
137 ----- * @param mixed num value to evaluate
138 ----- * @return boolean
139 ----- */
140 -----function isValidNumber(num){
141 -----    return !isNaN(parseFloat(num)) && isFinite(num);
```

checkout\_ui.js

Now you can turn your attention to opening the Stripe checkout form. It appears to be working. But here is a tweak that you might want to add.



# Preparing for the Stripe checkout form

## BEFORE

```
123 }else{
124 // Open Stripe checkout form
125 stripeFormHandler.open({
126 amount:amount * 100
127 });
128 }
129 e.preventDefault();
130 });
131 /**
132 * Validate is a number
```

checkout\_ui.js

## AFTER

```
126 }else{
127 $('.checkout-message').slideUp();
128 // Open Stripe checkout form
129 stripeFormHandler.open({
130 amount:amount * 100
131 });
132 }
133 e.preventDefault();
134 });
135 /**
136 * Validate is a number
137 * @param mixed num value to evaluate
138 * @return boolean
139 */
140 function isValidNumber(num){
141 return !isNaN(parseFloat(num)) && isFinite(num);
```

This line animates the closing all form messaging elements.  
That is done by applying the JQuery slideDown method to the checkout-message class.

# Preparing for the Stripe checkout form

BEFORE

```
123 }else{
124 // Open Stripe checkout form
125 stripeFormHandler.open({
126 amount:amount * 100
127 });
128 }
129 e.preventDefault();
130 });
131 /**
132 * Validate is a number
```

AFTER

```
126 }else{
127 $('.checkout-message').slideUp();
128 // Open Stripe checkout form
129 stripeFormHandler.open({
130 amount:amount * 100
131 });
132 }
133 e.preventDefault();
134 });
135 /**
136 * Validate is a number
137 * @param mixed num value to evaluate
138 * @return boolean
139 */
140 function isValidNumber(num){
141 return !isNaN(parseFloat(num)) && isFinite(num);
```

checkout\_ui.js

base.css

```
72 /*--
73 Form message contain
74 --*/
75 .checkout-message{
76 display:none;
77 font-weight:bold;
78 }
79 /*--
80 Text style for error
```

That class is used on all form messaging elements.

**BEFORE**

checkout\_ui.js

```
57  /**
58  *  Create a Stripe configuration object
59  */
60  var stripeConfig = {
61  key: $('#stripe-pk').val(),
62  image: "logo_128x128.png",
63  description: $('#quantity').val() + ' ' + $('#description').val(),
64  //panelLabel: 'Order total',
65  name : $('#company-name').val(),
66  //allowRememberMe: false,
```

**AFTER**

```
57  /**
58  *  Create a Stripe configuration object
59  */
60  var stripeConfig = {
61  key: $('#stripe-pk').val(),
62  image: "logo_128x128.png",
63  description: $('#description').val(),
64  //panelLabel: 'Order total',
65  name : $('#company-name').val(),
66  //allowRememberMe: false,
67  token: function(token){
68  console.log("token", token);
69  dataSend = {};
70  dataSend.stripeToken = token.id;
71  dataSend.amount = $('#amount').val();
72  dataSend.quantitv = $('#quantity').val();
```

The original example set the Stripe Checkout form description by including a quantity. This is found where the Stripe configuration object is set. You need to remove the quantity from that expression.

# Preparing for the Stripe checkout form

## BEFORE

checkout\_ui.js

```
57  /**
58  *  Create a Stripe configuration object
59  */
60  var stripeConfig = {
61  key: $('#stripe-pk').val(),
62  image: "logo_128x128.png",
63  description: $('#quantity').val() + ' ' + $('#description').val(),
64  //panelLabel: 'Order total',
65  name: $('#company-name').val(),
66  //allowRememberMe: false,
```

## AFTER

```
57  /**
58  *  Create a Stripe configuration object
59  */
60  var stripeConfig = {
61  key: $('#stripe-pk').val(),
62  image: "logo_128x128.png",
63  description: $('#description').val(),
64  //panelLabel: 'Order total',
65  name: $('#company-name').val(),
66  //allowRememberMe: false,
67  token: function(token){
68  console.log("token", token);
69  dataSend = {};
70  dataSend.stripeToken = token.id;
71  dataSend.amount = $('#amount').val();
72  dataSend.quantitv = $('#quantity').val();
```



Refactoring code always presents challenges that can stop you.  
In this case Javascript would fail silently.  
That would send you off into debugging.

# Preparing for the Stripe checkout form

## BEFORE

```
67 token: function(token) {
68   console.log("token", token);
69   dataSend = {};
70   dataSend.stripeToken = token.id;
71   dataSend.amount = $('#amount').val();
72   dataSend.quantity = $('#quantity').val();
73   dataSend.statement_descriptor = $('#statement-descriptor').val();
74   ;
75   dataSend.description = $("#description").val();
76   dataSend.receipt_email = token.email;
```

## AFTER

```
67 token: function(token) {
68   console.log("token", token);
69   dataSend = {};
70   dataSend.stripeToken = token.id;
71   dataSend.amount = $('#amount').val();
72   dataSend.statement_descriptor = $('#statement-descriptor').val();
73   ;
74   dataSend.description = $("#description").val();
75   dataSend.receipt_email = token.email;
76   console.log("dataSend", dataSend);
77   $('#checkout-btn').hide();
78   $('#checkout-processing-message').show();
79   $.ajax({
80     type: "post",
81     url: 'checkout_charge_card.php',
82     data: dataSend,
```

checkout\_ui.js



And here is one more gotcha.

You no longer need to send the quantity to the backend for processing.

So remove it from the dataSend object and save your file.

# Preparing for the Stripe checkout form

## BEFORE

```
22 — $charge = \Stripe\Charge::create(array(  
23 —   "amount" => $_POST['amount'] * 100, // Convert to cents  
24 —   "currency" => "usd",  
25 —   "source" => $_POST['stripeToken'],  
26 —   "receipt_email" => $_POST['receipt_email'],  
27 —   "statement_descriptor" => substr(strtoupper($_POST['  
   statement_descriptor']), 0, 22),  
28 —   "description" => $_POST['quantity'] . ' ' . $_POST['description'])  
29 — );  
30 — $return_data["success"] = true;
```

## AFTER

```
20 try{  
21 — // Create a Stripe\Charge OOP object  
22 — $charge = \Stripe\Charge::create(array(  
23 —   "amount" => $_POST['amount'] * 100, // Convert to cents  
24 —   "currency" => "usd",  
25 —   "source" => $_POST['stripeToken'],  
26 —   "receipt_email" => $_POST['receipt_email'],  
27 —   "statement_descriptor" => substr(strtoupper($_POST['  
   statement_descriptor']), 0, 22),  
28 —   "description" => $_POST['description'])  
29 — );  
30 — $return_data["success"] = true;  
31 — // $return_data["charge"]["id"] = $charge->id;  
32 — // $return_data["charge"]["status"] = $charge->status;  
33 // Invalid request errors arise when your request has invalid parameters.  
34 }catch(\Stripe\Error\InvalidRequest $e){
```

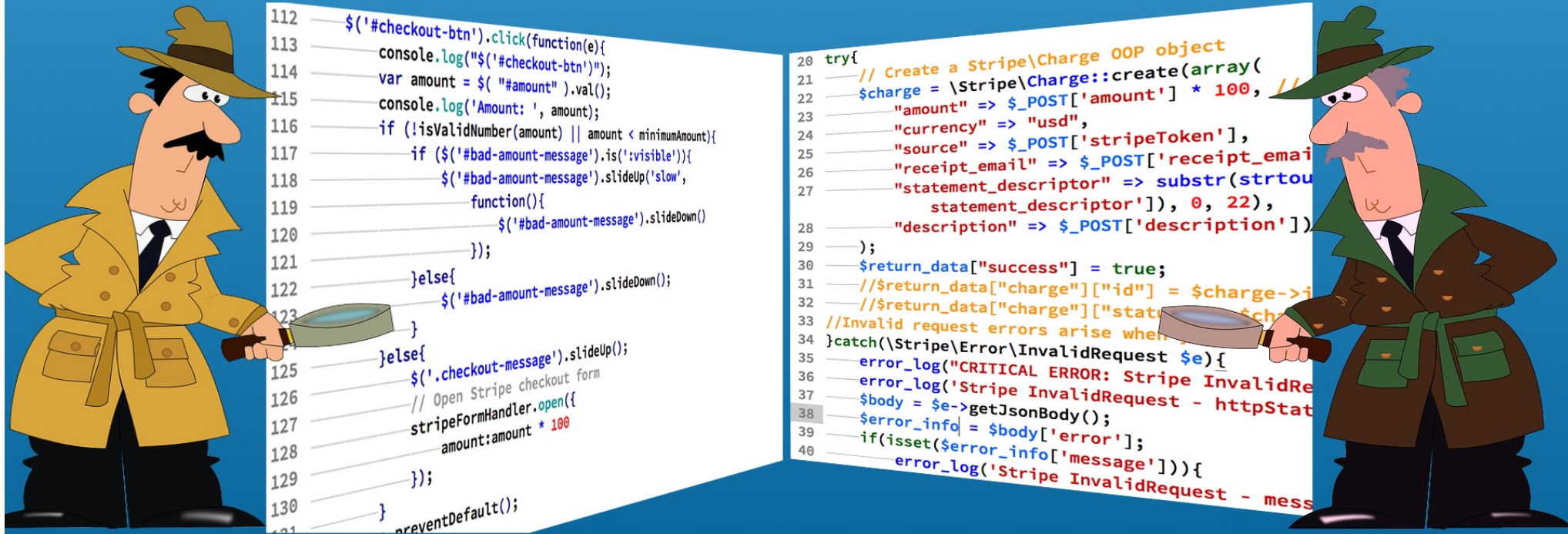
checkout\_charge.php



And just as if you think you got quantity removed, you find it one more time in the checkout\_charge.php file.

You no longer are sending the quantity via AJAX so the PHP `$_POST` variable will not have it. Remove it from the expression, then save your file.

# Checkpoint 7



Changes are found in the checkpoint\_07 folder if you need to check your work.

# Test Amount Input Error Messaging

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

**Input amount has an error.**

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



Now you can try invalid amount inputs.

You should see the error message.


Your error message should slide in and out if you repeat similar amount errors followed by clicking the Checkout button.



# Hiding Input Elements

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example



|  |                                     |
|--|-------------------------------------|
| Item: Widget Mystery Box                   |                                     |
| Amount: <input type="text" value="50.00"/> | <input type="button" value="\$50"/> |
| Processing ...                             |                                     |

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)

A last item that you may have noticed it that the input fields remain open as the checkout process begins.

# Hiding Input Elements

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

Thank you for your order.



[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)

This condition does not change after the checkout process completes.  
This can create confusion for the user.  
So you need to either disable the input fields or hide them.

# Hiding Input Elements

**Stripe Checkout Embedded Simple Using Ajax**

**Name Your Price Example**

Processing ...



**Stripe Checkout Embedded Simple Using Ajax**

**Name Your Price Example**

Thank you for your order.

You also replace them with printable receipt when the order is complete. There are many possibilities depending on the nature of the need. For this example, you can hide these fields.

# The input-fields element

checkout\_simple.php

```
30 <div class="form-container center-container">
31   <div id="input-fields">
32     <p>Item: <?=$description?></p>
33     <p>Amount: <input type="text" id="amount"/>
34     <select id="preset-amounts">
35       <option value="other">Other</option>
36       <option value="10">$10</option>
37       <option value="25">$25</option>
38       <option value="50">$50</option>
39       <option value="100">$100</option>
40       <option value="500">$500</option>
41       <option value="1000">$1,000</option>
42       <option value="10000">$10,000</option>
43     </select></p>
44   </div>
45   <input type="hidden" id="stripe-pk" value="<?=$st_test_public_key?>"/>
46   <input type="hidden" id="company-name" value="<?=$company_name?>"/>
47   <input type="hidden" id="preset-amount" value="<?=$preset_amount?>"/>
```

You may recall adding the input-fields container.

It has the input elements that you want to hide as children.

You can hide this container during processing.

You can unhide it should something go wrong and you wish to provide a retry opportunity to the user.

# Hiding the input-fields element

## BEFORE

checkout\_ui.js

```
75 console.log("dataSend", dataSend);
76 $('#checkout-btn').hide()
77 $('#checkout-processing-message').show();
78 $.ajax({
79     type:"post",
80     url:'checkout_charge_card.php',
81     data:dataSend,
82     dataType:'json',
83 })
84 .done(function(data, status){
```

## AFTER

```
75 console.log("dataSend", dataSend);
76 $( "#input-fields" ).slideUp('fast');
77 $('#checkout-btn').hide()
78 $('#checkout-processing-message').show();
79 $.ajax({
80     type:"post",
81     url:'checkout_charge_card.php',
82     data:dataSend,
83     dataType:'json',
84 })
85 .done(function(data, status){
86     console.log("$.ajax.done");
87     console.log("data: ", data);
88     console.log("status: ", status);
89     $('#checkout-processing-message').hide();
90     if (data.success){
```

Here is a place in the Javascript file that you can unhide the input-fields element reliably. Add a slideUp method for the input-fields element.

# Hiding the input-fields element

## BEFORE

checkout\_ui.js

```
85 ----- .done(function(data, status){
86 -----     console.log("$.ajax.done");
87 -----     console.log("data: ", data);
88 -----     console.log("status: ", status);
89 -----     $('#checkout-processing-message').hide();
90 -----     if (data.success){
91 -----         $('#checkout-success-message').show();
92 -----     }else{
93 -----         $('#checkout-fail-message').show();
94 -----     }
```

checkout\_charge.php

```
15 // Data to return
16 $return_data = array();
17 // Success or failure
18 $return_data["success"] = false;
```

## AFTER

```
85 ----- .done(function(data,
86 -----     console.log("$.aj
87 -----     console.log("data: ", data);
88 -----     console.log("status: ", status);
89 -----     $('#checkout-processing-message').hide();
90 -----     if (data.success){
91 -----         $('#checkout-success-message').show();
92 -----     }else{
93 -----         $( "#input-fields" ).slideDown();
94 -----         $('#checkout-fail-message').show();
95 -----     }
96 ----- })
97 ----- .fail(function(data, status, error){
98 -----     console.log("$.ajax.fail");
99 -----     console.log("data: ", data);
100 -----     console.log("status: ", status);
```

The server program returns a success value.  
When that is false you can show the input-fields element.  
You do that in the AJAX .done callback function.

# Checkpoint 8

```
76     $( "#input-fields" ).slideUp('fast');
77     $('#checkout-btn').hide()
78     $('#checkout-processing-message').show();
79     $.ajax({
80         type:"post",
81         url:'checkout_charge_card.php',
82         data:dataSend,
83         dataType:'json',
84     })
85     .done(function(data, status){
86         console.log("$.ajax.done");
87         console.log("data: ", data);
88         console.log("status: ", status);
89         $('#checkout-processing-mes
90         if (data.success){
91             $('#checkout-success-message').show():
92         }else{
93             $( "#input-fields" ).slideDown();
94             $('#checkout-fail-message').show();
95     }
```



Save your work.

Changes are found in the checkpoint\_08 folder if you need to check your work.

# Test End to End

## Stripe Checkout Embedded Simple Using Ajax

### Name Your Price Example

Item: Widget Mystery Box

Amount:

Thank you for your order.

[https://your\\_domain/checkout\\_simple.php](https://your_domain/checkout_simple.php)



Now you can reload in the web browser.  
Run some tests end to end for various scenarios.



# Wrap Up



There you have it from soup to nuts.  
The basic Stripe coding remained the same.  
Lots of details fell in the areas of the HTTP url line, PHP, JQuery and Javascript.

# **Stripe Checkout Pages Using PHP**

## **Using Stripe JS and AJAX**

### **Bonus: The Name Your Price Example**

**With Lon Hosford**

**© 2015 Alonzo Hosford**

Copyright 2015 Alonzo L. Hosford. All Rights Reserved. [www.lonhosford.com](http://www.lonhosford.com)

This is a Visual Step by Step Workbook and voice transcript for accompanying video for this portion of the course.

