
TRP.
407
try
ember

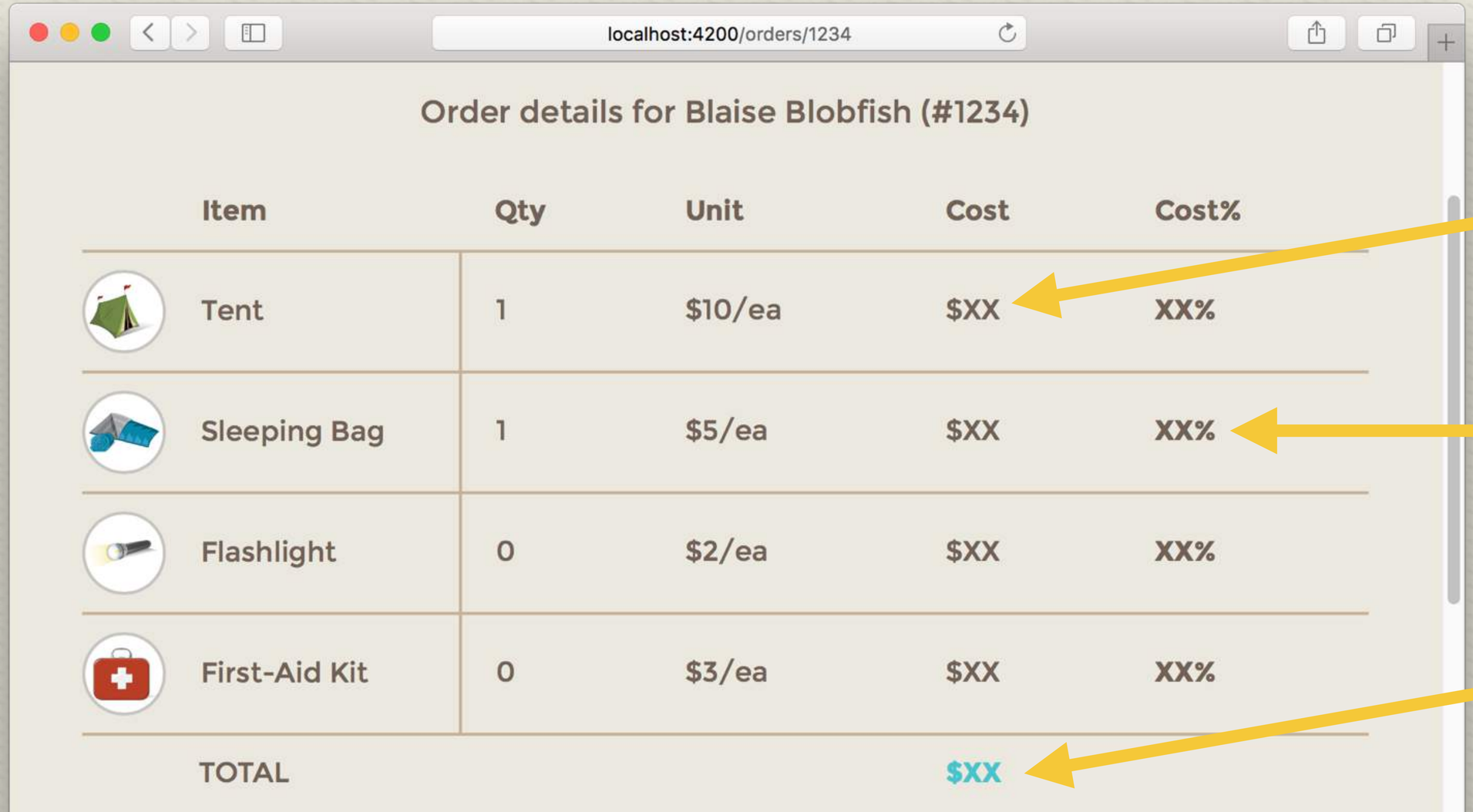
Level 5.1

Properties and Components

Computed Properties

Finishing Off the Receipt

The app is nearly complete — there are just a few pieces left to implement on the order receipt.



Calculate LineItem costs

Calculate and style the cost percentages

Missing total price

Cleaning Up the Receipt

Reaching through one object to work with another is bad (the Law of Demeter).

app/templates/orders/order.hbs

```
{{#each model.items as |lineItem|}}
  <tr>
    <td>{{lineItem.product.title}}</td>
    <td>{{lineItem.quantity}}</td>
    <td>{{lineItem.product.price}}</td>
    <td>$XX</td>
    <td>XX%</td>
  </tr>
{{/each}}
```

What we want

app/templates/orders/order.hbs

```
{{#each model.items as |lineItem|}}
  <tr>
    <td>{{lineItem.title}}</td>
    <td>{{lineItem.quantity}}</td>
    <td>{{lineItem.unitPrice}}</td>
    <td>$XX</td>
    <td>XX%</td>
  </tr>
{{/each}}
```

The Law of Demeter is the “principle of least knowledge.”
Units should only talk to their immediate friends.

Introducing Computed Properties

Computed properties are function-calculated, cached properties.

app/models/line-item.js

```
import Ember from 'ember';

export default Ember.Object.extend({
  title: Ember.computed('product.title', function() {
    return this.get('product.title');
  })
});
```

```
this.get('title')
```

Like all properties, computed properties are queried with get().

The function's return value is cached as the LineItem's title property value.

Dependent properties of the function are listed. When their value changes, the property is recalculated.

Using Predefined Macros

Ember ships with about 30 predefined computed property macros.

app/models/line-item.js

```
import Ember from 'ember';

export default Ember.Object.extend({
  title: Ember.computed('product.title', function() {
    return this.get('product.title');
  })
});
```

These are largely equivalent.

```
  title: Ember.computed.alias('product.title')
```

Macros

- alias
- collect
- empty
- equal
- filterBy
- mapBy
- sort
- sum
- uniq

These are just a few of the many macros provided.

Adding the Unit Price

The unit price is just another property alias.

app/models/line-item.js

```
import Ember from 'ember';

export default Ember.Object.extend({
  title: Ember.computed.alias('product.title'),

  unitPrice: Ember.computed.alias('product.price')
});
```

The local property name (unitPrice) doesn't have to match the aliased property name (price).

Using the Aliases

The template can now be updated to use the lineItem title and unitPrice.

app/templates/orders/order.hbs

```
{{#each model.items as |lineItem|}}
  <tr>
    <td>{{lineItem.title}}</td>
    <td>{{lineItem.quantity}}</td>
    <td>{{lineItem.unitPrice}}</td>
    <td>$XX</td>
    <td>XX%</td>
  </tr>
{{/each}}
```

Determining the LineItem Price

The LineItem cost is the quantity multiplied by the unitPrice.

localhost:4200/orders/1234

Order details for Blaise Blobfish (#1234)

Item	Qty		Unit		Cost	Cost%
Tent	1	x	$10/ea	=	$XX $10	XX%
Sleeping Bag	1	x	$5/ea	=	$XX $5	XX%
Flashlight	0	x	$2/ea	=	$XX $0	XX%
First-Aid Kit	0	x	$3/ea	=	$XX $0	XX%
TOTAL					$XX	

Calculating the LineItem Price

The LineItem cost is the quantity multiplied by the unitPrice.

app/models/line-item.js

```
import Ember from 'ember';

export default Ember.Object.extend({
  price: Ember.computed('quantity', 'unitPrice', function() {
    return parseInt(this.get('quantity'), 10) * this.get('unitPrice');
  }),

  title: Ember.computed.alias('product.title'),
  unitPrice: Ember.computed.alias('product.price')
});
```

parseInt() is used because quantity is getting set from a form input. Form inputs return string values. Adding the base 10 indicator is just a good practice.

Using the Computed Price

With the LineItem price calculated, we can add it to the template.

app/templates/orders/order.hbs

```
{{#each model.items as |lineItem|}}
  <tr>
    <td>{{lineItem.title}}</td>
    <td>{{lineItem.quantity}}</td>
    <td>{{lineItem.unitPrice}}</td>
    <td>${{lineItem.price}}</td>
    <td>XX%</td>
  </tr>
{{/each}}
```

Progressing Through the Receipt

The item title, quantity, unitPrice, and cost are now on the receipt.

localhost:4200/orders/1234

Order details for Blaise Blobfish (#1234)

Item	Qty	Unit	Cost	Cost%
Tent	1	$10/ea	$10	XX%
Sleeping Bag	1	$5/ea	$5	XX%
Flashlight	0	$2/ea	$0	XX%
First-Aid Kit	0	$3/ea	$0	XX%
TOTAL			$XX	

Calculate and style the cost percentages

Missing total price

Determining the Order Price

The order price should be the sum of the order's LineItem prices.

localhost:4200/orders/1234

Order details for Blaise Blobfish (#1234)

	Item	Qty	Unit	Cost	Cost%
	Tent	1	$10/ea	$10 $10	XX%
	Sleeping Bag	1	$5/ea	+ $5 = $15	XX%
	Flashlight	0	$2/ea	+ $0 = $15	XX%
	First-Aid Kit	0	$3/ea	+ $0 = $15	XX%
	TOTAL			= $XX $15	

Collecting the LineItem prices

The mapBy macro creates a new array containing mapped properties.

app/models/order.js

```
import Ember from 'ember';

export default Ember.Object.extend({
  itemPrices: Ember.computed.mapBy('items', 'price')
});
```

This will be an array of LineItem prices.

Name of the collection to use

The element property to map

```
this.get('itemPrices'); //=> [10, 5, 0, 0]
```

Now we need to sum the LineItem prices.

Macros

- alias
- collect
- empty
- equal
- filterBy
- **mapBy**
- sort
- sum
- uniq

The mapped array will automatically update if the items array changes or its price values change.

Performing the Summation

The sum macro calculates a summation value from an array of numerics.

app/models/order.js

```
import Ember from 'ember';

export default Ember.Object.extend({
  itemPrices: Ember.computed.mapBy('items', 'price'),
  price: Ember.computed.sum('itemPrices')
});
```

This will contain the order price.

The collection to sum

```
this.get('price'); //=> 15
```

The calculated sum will automatically update
itemPrices array changes.

Macros

- alias
- collect
- empty
- equal
- filterBy
- mapBy
- sort
- **sum**
- uniq

Exercising the Macros

The macros and computed properties are all working as expected.

localhost:4200/orders/1234

Order details for Blaise Blobfish (#1234)

	Item	Qty	Unit	Cost	Cost%
	Tent	1	$10/ea	$10	XX%
	Sleeping Bag	1	$5/ea	$5	XX%
	Flashlight	0	$2/ea	$0	XX%
	First-Aid Kit	0	$3/ea	$0	XX%
	TOTAL			$15	

Level 5.1

Properties and Components

Computed Properties