Template Driven Validation

- Uses html validator attributes.
- Required, minlength, maxlength, pattern, etc.

```
<input
 type="text"
 class="form-control"
 placeholder="Expense name"
 name="name"
 [(ngModel)]="model.name"
 required
/>
```

- Can use template variables to set validation related attributes:

```
<button class="btn btn-primary" [disabled]="!expenseForm.valid">
  Add expense
</button>
```

Template Driven State and Styling

State	Class if true	Class if false
The control has been visited.	ng-touched	ng-untouched
The control's value has changed.	ng-dirty	ng-pristine
The control's value is valid.	ng-valid	ng-invalid

```
.ng-invalid:not(form) {
  border-left: 5px solid red;
}
```

```
<input
 type="text"
 class="form-control"
 placeholder="Expense name"
 name="name"
 [(ngModel)]="model.name"
 required
 pattern="Fuel"
 #expenseName="ngModel"
 required
/>
<div [hidden]="expenseName.valid || expenseName.pristine">
 <div *ngIf="expenseName.errors?.['required']">
  Expense name is required
 </div>
 <div *ngIf="expenseName.errors?.['pattern']">
  Only "Fuel" Expenses are allowed
 </div>
</div>
```

Template Driven Forms task #3: Validation.

- We don’t want lazy users to enter whatever they want. Add validation for the expense form controls.

Field	Validation	Message
Expense name	Only alphabetical characters	“Only alphabetical characters”
All	Required	“<field> is required”
Expense amount	5 symbols allowed	“Please enter no more than 5 numbers”

- (Extra): Add styling for invalid fields.
- Note: Try `[a-zA-Z]*` as the pattern for validation.