Reactive Forms!

```html
<form [formGroup]="expenseForm" (submit)="addExpense()">
  <div class="row">
    <div class="col">
      <input
        type="text"
        class="form-control"
        placeholder="Expense name"
        name="name"
        formControlName="name"
      />
    </div>
```

```typescript
import { ReactiveFormsModule } from '@angular/forms';
@NgModule({
  declarations: [AppComponent, ExpenseFormComponent],
  imports: [HttpClientModule, ReactiveFormsModule],
  providers: [],
  bootstrap: [AppComponent],
})
```

Reactive Forms

- More documentation under https://angular.io/guide/reactive-forms
- Observables to track value changes and updates.
- Immutable - return new form state on every change.
- Allow validation in the code. Easier to customise.

Reactive Forms

- Made up of FormControls (Single FormControl), FormGroups (Map of Form Controls) and FormArrays (Arrays of FormGroups).

```
expenseForm: FormGroup = new FormGroup({
  name: new FormControl(''),
  date: new FormControl(''),
  amount: new FormControl(''),
});
```

- Defined in the code - only visualised in the template:
- ‘[formGroup] added to form element:

```
<form [formGroup]="expenseForm" (submit)="addExpense()">
```

- ‘formControlName’ attribute binds HTML form control to code:

```
<input
 type="text"
 class="form-control"
 placeholder="Expense name"
 name="name"
 formControlName="name"
/>
```

- “FormBuilder” can be injected for easier creation of forms:

```
  constructor(private fb: FormBuilder) {
    this.expenseForm = this.fb.group({
      name: [''],
      date: [''],
      amount: [''],
    });
  }
```

Reactive Form Validation

```
name: ['', [Validators.required, Validators.pattern('Fuel')]],
```

```
get expenseName() {
  return this.expenseForm.get('name');
}
```

```
<input
 type="text"
 class="form-control"
 placeholder="Expense name"
 name="name"
 formControlName="name"
/>
<div
 [hidden]="expenseName?.valid || expenseForm.controls['name'].pristine"
>
 <div *ngIf="expenseName?.errors?.['required']">
  Expense name is required
 </div>
 <div *ngIf="expenseForm.controls['name'].errors?.['pattern']">
  Only expenses for fuel allowed
 </div>
</div>
```

Template driven to Reactive template

```
<input
 type="text"
 class="form-control"
 placeholder="Expense name"
 name="name"
 [(ngModel)]="model.name"
 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 to Reactive template

```html
<input
 type="text"
 class="form-control"
 placeholder="Expense name"
 name="name"
 formControlName="name"
/>
<div
 [hidden]="expenseName?.valid || expenseForm.controls['name'].pristine"
>
 <div *ngIf="expenseName?.errors?.['required']">
  Expense name is required
 </div>
 <div *ngIf="expenseForm.controls['name'].errors?.['pattern']">
  Only "Fuel" expenses are allowed
 </div>
</div>
```

Reactive Forms Task #1

- Convert template driven form to reactive.
- Convert validations from template driven to reactive.