Validation basics

- Validators are functions, using the ‘ValidationErrors | null’ or ‘Observable<ValidationErrors | null>’ return types.
- ‘Null’ indicates a field is VALID.
- ‘ValidationErrors’ return an object with the error name as the key and an arbitrary value, which can then be used in the error message.

Custom Validators

- Functions adhering to a specific interface, which allow complex validation rules for fields.

```
import { AbstractControl, ValidationErrors } from '@angular/forms';
export const customValidator = (
  control: AbstractControl
): ValidationErrors | null => {
  const rand = control.value.startsWith('A') || Math.random();
  return rand || rand < 0.5 ? { customError: { value: control.value } } : null;
};
```

```
name: new FormControl('', {
 updateOn: 'blur',
 validators: [Validators.required, customValidator],
}),
```

```
<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 #3

- Add custom validator to expense name that marks names with explicit language as invalid. Explicit words should be provided via an array and the validator should check if the content includes any of those words.
- Add an error message saying “No explicit language allowed”