In Angular, when you use a Material **`matInput`**, you can detect value changes in **real-time** (as the user types) or **on blur** (when the field loses focus).
Here's a complete example showing both approaches.

1. Using `(input)` Event — Real-time detection

```
// app.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  onValueChange(event: Event) {
    const input = event.target as HTMLInputElement;
    console.log('Value changed:', input.value);
  }
}
```

```
<!-- app.component.html -->
<mat-form-field appearance="outline">
  <mat-label>Enter text</mat-label>
  <input matInput (input)="onValueChange($event)" placeholder="Type something">
</mat-form-field>
```

✅ Fires **on every keystroke**.

2. Using `FormControl.valueChanges` — Reactive Forms

```
// app.component.ts
import { Component } from '@angular/core';
import { FormControl } from '@angular/forms';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  myControl = new FormControl('');

  constructor() {
    this.myControl.valueChanges.subscribe(value => {
      console.log('Reactive form value changed:', value);
    });
  }
}
```

```
<!-- app.component.html -->
<mat-form-field appearance="outline">
  <mat-label>Enter text</mat-label>
  <input matInput [formControl]="myControl" placeholder="Type something">
</mat-form-field>
```

✅ Best for **form validation** and **reactive programming**.

3. Using `(ngModelChange)` — Template-driven Forms

```
// app.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  value = '';

  onModelChange(newValue: string) {
    console.log('ngModel value changed:', newValue);
  }
}
```

```
<!-- app.component.html -->
```

```
<mat-form-field appearance="outline">
  <mat-label>Enter text</mat-label>
  <input matInput [(ngModel)]="value" (ngModelChange)="onModelChange($event)"
placeholder="Type something">
</mat-form-field>
```

✅ Works well for **simple forms** without reactive form setup.

Recommendation

- Use **`(input)`** for quick, DOM-based detection.
- Use **`FormControl.valueChanges`** for reactive forms (preferred in large apps).
- Use **`(ngModelChange)`** for template-driven forms.

If you want, I can give you a **version that also debounces the input** so it doesn't fire on every keystroke but only after the user stops typing.
Do you want me to prepare that?