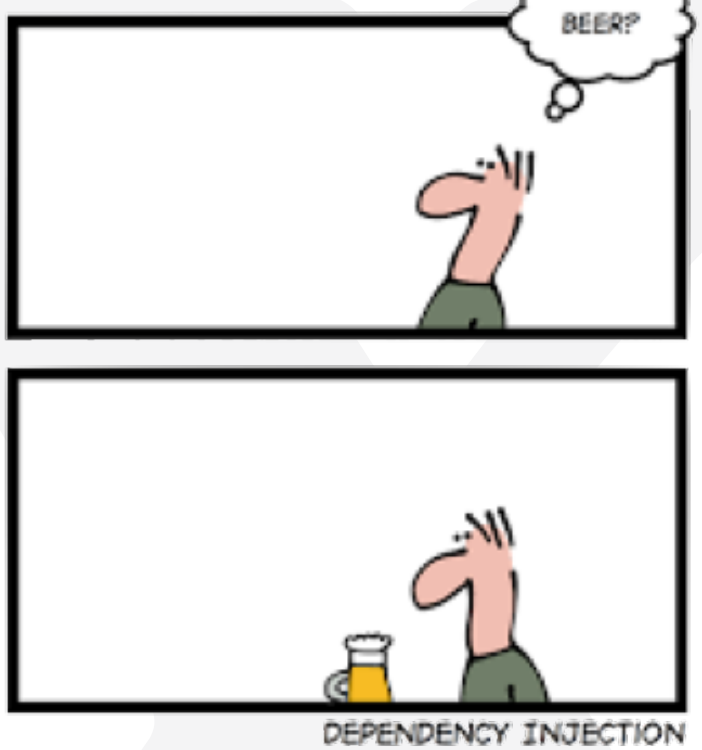
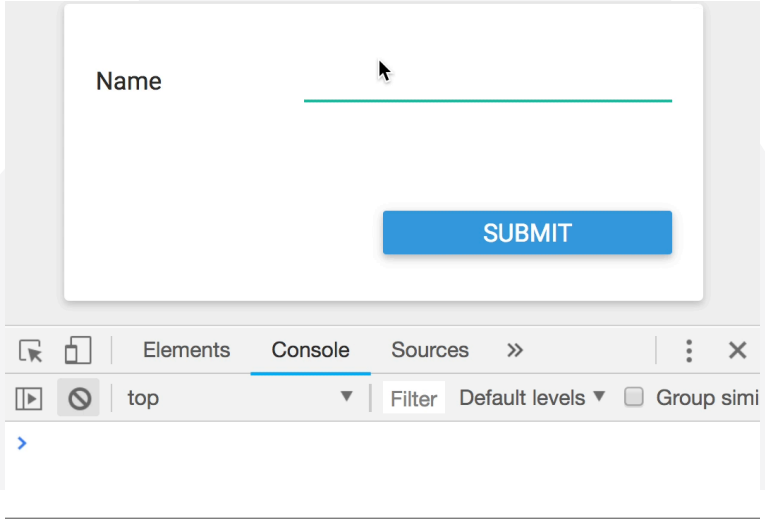
Recap Day 2

Day 2

- Dependency Injection and HTTP
- Testing
- Forms





Dependency injection

- Manage dependencies
- No manual instantiation
- Injected by Angular
- Share common tasks

Creating services

@Injectable()

```
@Injectable({providedIn: 'root'})
export class ContactsService {

 constructor() { }

  fetchContacts(): Contact[] {
    return [ {…}, {…}, {…} ];
  }
}
```

Dependency injection

Constructor injection

```
@Component({ selector: 'contacts' })
export class ContactsComponent implements OnInit {

  contacts: Contact[];

  constructor(private contactsService: ContactsService) {}

  ngOnInit() {
    this.contacts = this.contactsService.fetchContacts()
  }

}
```

Testing - Tools and environment

- Unit test runner
- Validate isolated code

- e2e test runner
- Simulate a user
- Validate UI

- Test structure
- Assertions
- Matchers

Forms

- Reactive vs Template Driven
- FormGroup, FormControl, FormBuilder
- Validators

Reactive Forms

```
@Component({ selector: 'contacts' })
export class ContactsComponent {

  contactForm = new FormGroup({
      name: new FormControl('',
            [ Validators.required,
              Validators.minLength(3)]
      )
  });

  get name() { return
    this.contactForm.get('name')
  }

}
```

```
<form [formGroup]="contactForm"
      (ngSubmit)="submit()">

  <input
    formControlName="name" />

  <span *ngIf="name.invalid">
    Name is required.
  </span>

</form>
```

Today

- Routing
- RxJS
- Material Design Library