Angular - recap Day 2

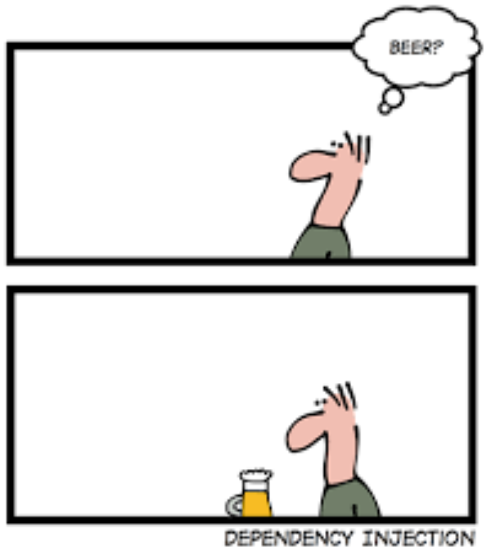
Day 1

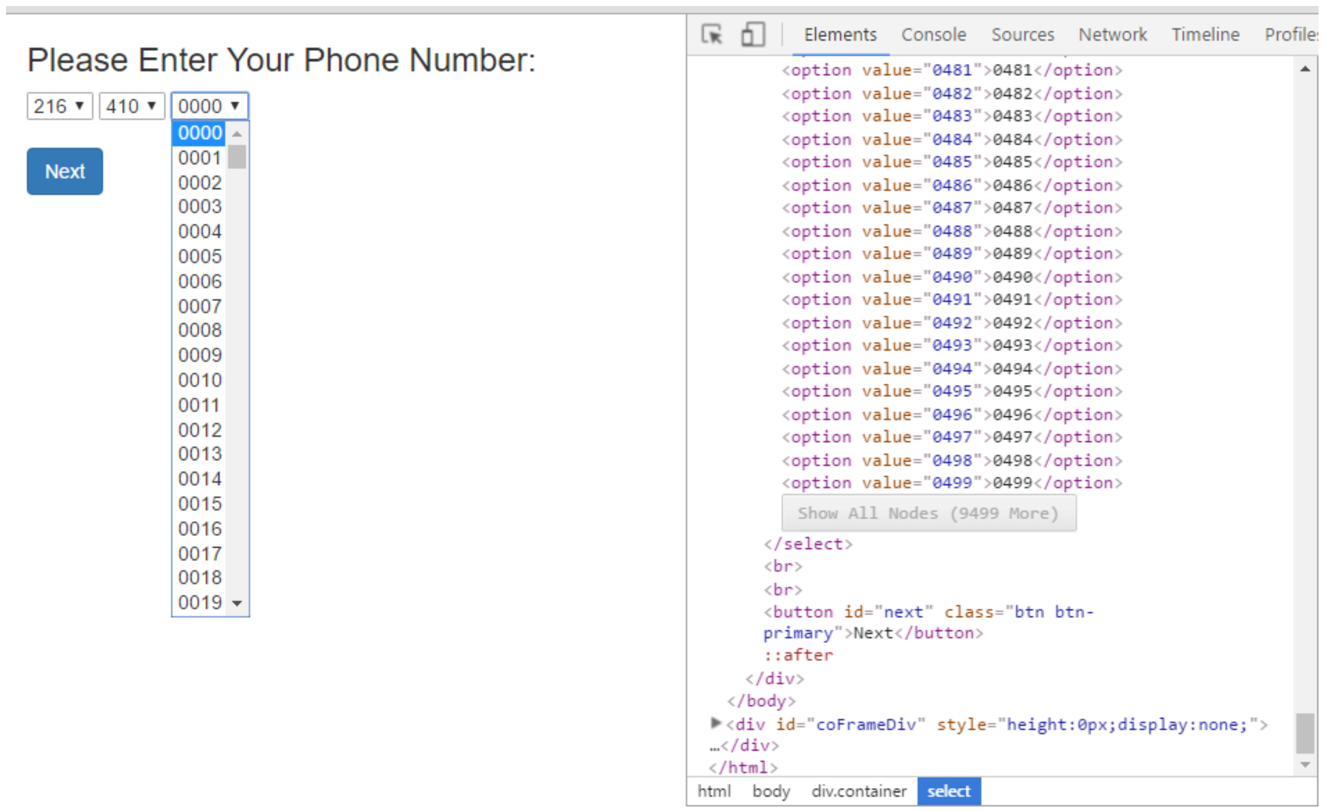
- Introduction
- Modules and Components
- Template Syntax

Yesterday

- Dependency Injection and HTTP
- Testing
- Forms

Recap





Please Enter Your Phone Number:

216 ▾ 410 ▾ 0000 ▾

0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019

Next

Elements Console Sources Network Timeline Profile

```
        <option value="0481">0481</option>
        <option value="0482">0482</option>
        <option value="0483">0483</option>
        <option value="0484">0484</option>
        <option value="0485">0485</option>
        <option value="0486">0486</option>
        <option value="0487">0487</option>
        <option value="0488">0488</option>
        <option value="0489">0489</option>
        <option value="0490">0490</option>
        <option value="0491">0491</option>
        <option value="0492">0492</option>
        <option value="0493">0493</option>
        <option value="0494">0494</option>
        <option value="0495">0495</option>
        <option value="0496">0496</option>
        <option value="0497">0497</option>
        <option value="0498">0498</option>
        <option value="0499">0499</option>
        Show All Nodes (9499 More)
      </select>
      <br>
      <br>
      <button id="next" class="btn btn-
      primary">Next</button>
      ::after
    </div>
  </body>
▶<div id="coFrameDiv" style="height:0px;display:none;">
…</div>
</html>
```

html body div.container select

Dependency injection

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

Dependency injection

Provided by Module

```
@NgModule({
   declarations: [ … ],
   providers: [ ContactsService ],
   imports: [],
   exports: []
})
export class AppModule {

}
```

ilionx
Your partner in digital business

Dependency injection

constructor injection

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

  contacts: Contact[];

  constructor(private contactsService: ContactsService) {}

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

}
```

onx
Your partner in digital business

Testing

- Karma
- Protractor
- Jasmine
- @angular/…/testing

Testing

Jasmine

```
describe(‘The tested feature', () => {
  beforeEach(() => { // setup code });

  describe(‘could be nested', () => {
    it('is a test', () => { … });
  });
});
```

ilionx
Your partner in digital business

Testing

TestBed

```
// mocking services
{ provide: BookService, useClass: MyBookMockService }

it(‘Instance has been injected',
  inject([ BookService ], () => {
    const bookService = TestBed.get(BookService);
    expect(bookService).toBeDefined();
}));
```

ilionx
Your partner in digital business

Forms

- Reactive vs Template Driven
- FormGroup, FormControl, FormBuilder
- Events (submit, change)
- Validators

Reactive Forms

FormGroup & FormControl

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

  contactForm = new FormGroup({
      name: new FormControl()
  });

}
```

```
<form [formGroup]="contactForm">

  <input
    formControlName="name" />

</form>
```

Reactive Forms

Submit

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

  contactForm = new FormGroup({
      name: new FormControl()
  });

  submit() {
    console.log(
      this.contactForm.value
    );
  }
}
```

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

  <input
    formControlName="name" />

  <button type="submit">
    Send
  </button>

</form>
```

Reactive Forms - validators

- Validators.required
- Validators.minLength(i)
- Validators.pattern(//)
- `(c: FormControl): ValidationErrors | null => { }`

ilionx
Your partner in digital business

Today

- Routing
- RxJS
- Material Design Library