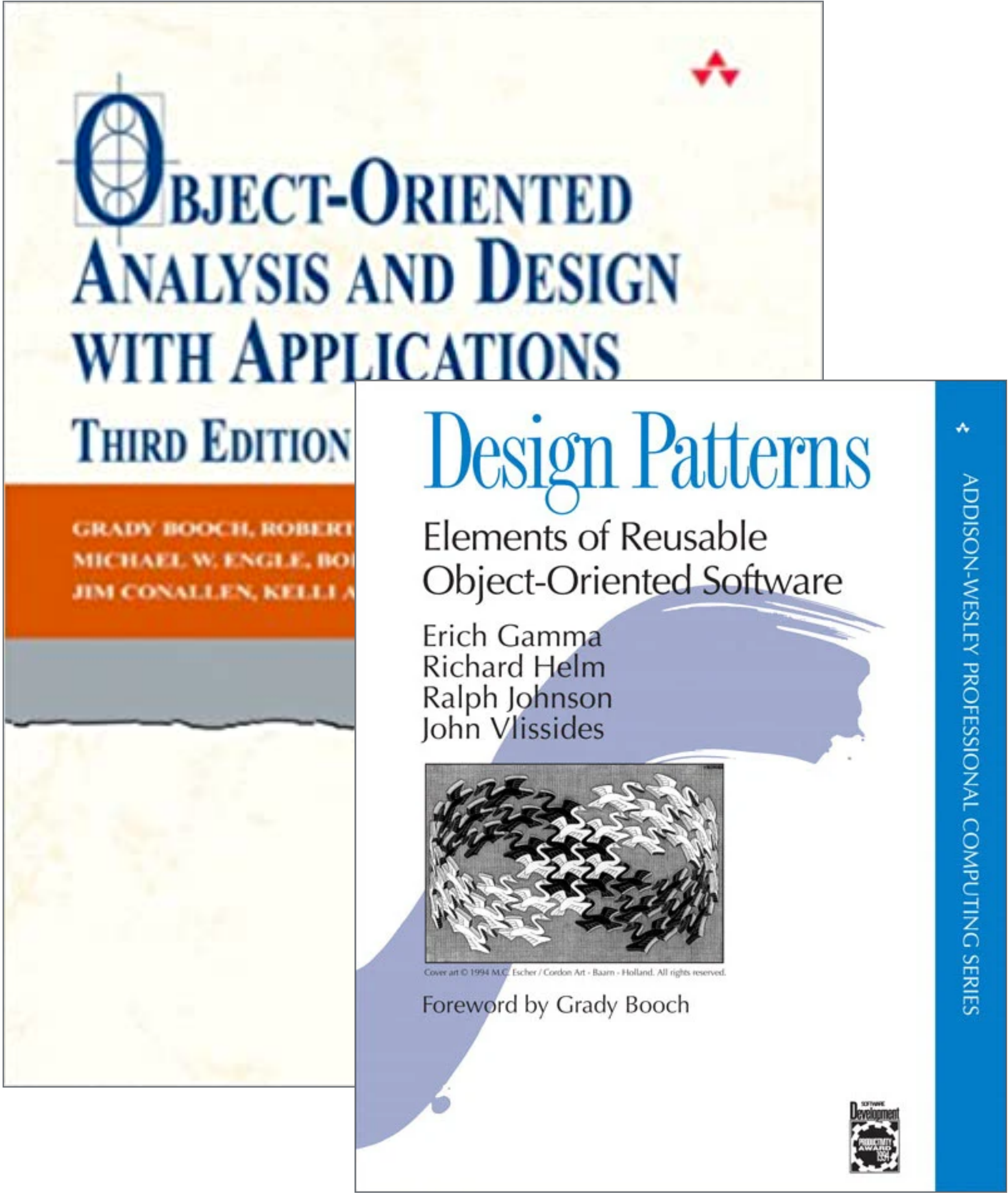


Въведение в Angular

Минко Гечев, май 2025

ФМИ, 2010





```
var newstr = null;
  var lastnewstr = null;
  var newclass = null;
  ...

function check(s, i) {
  p1 = document.getElementById('id_1_' + i);
  p2 = document.getElementById('id_2_' + i);
  if (s !== lastnewstr)
  {
    newclass2 = p1.className;
    newclass3 = p2.className;
    newclass = s.className;
    s.className = 'check';
    p1.className = 'viz';
    p2.className = 'viz';
    if (lastnewstr !== null)
    {
      lastnewstr.className = lastclass;
      lastnewstr2.className = lastclass2;
      lastnewstr3.className = lastclass3;
    }
  }
  lastnewstr = s;
  lastnewstr2 = p1;
  lastnewstr3 = p2;
  lastclass = newclass;
  lastclass2 = newclass2;
  lastclass3 = newclass3;
}
```



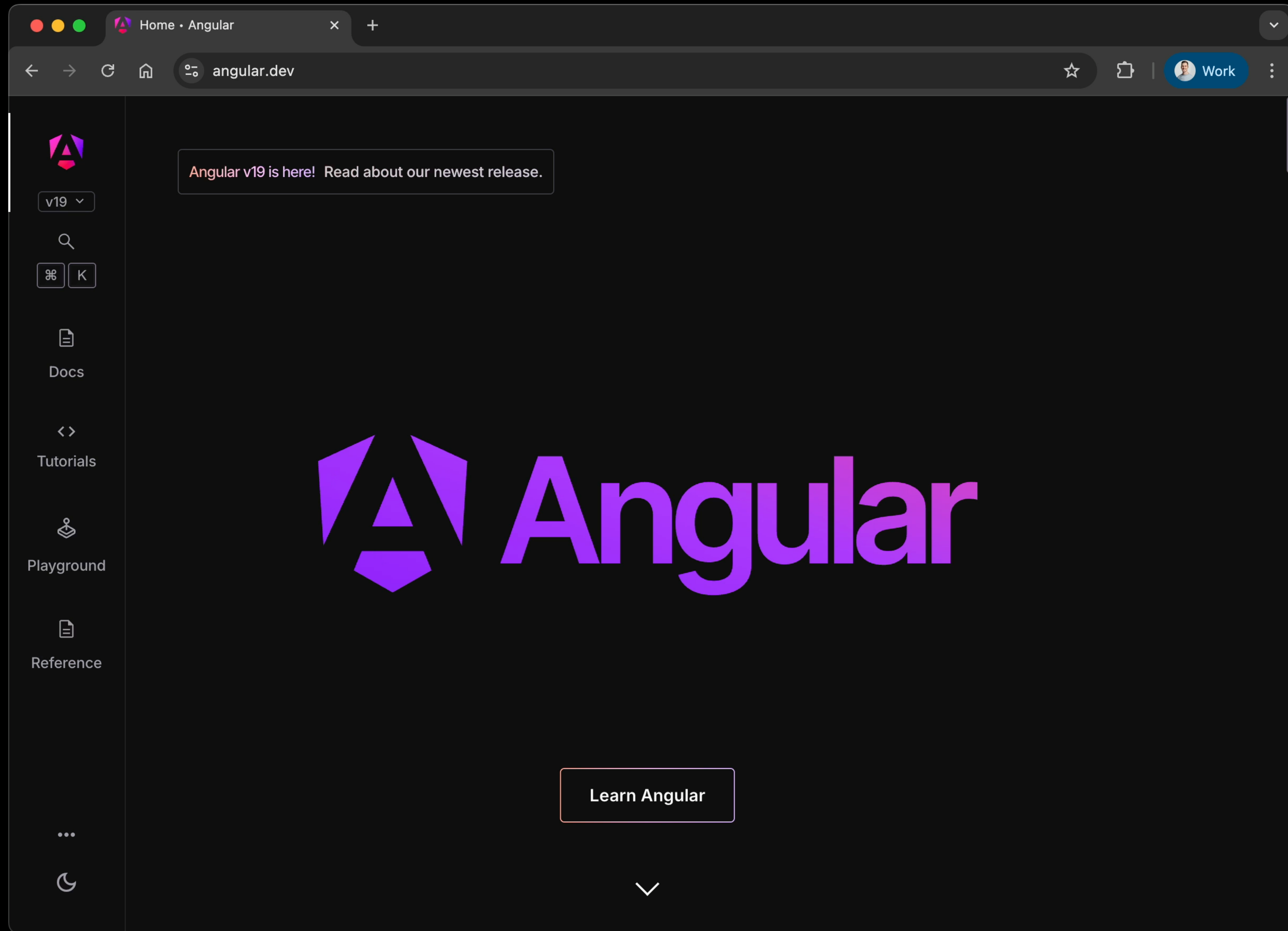




Angular ме научи на:

- Шаблони за дизайн
- Писане на тестваем код
- Разделяне на отговорностите
- Много други

Какво е Angular?



Gemini

Goldman
Sachs

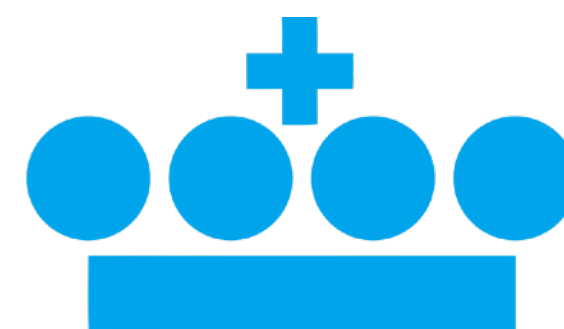
SPACEX



ClickUp



BUNDESLIGA



KLM



RYANAIR

Цели на Angular

- Улеснява създаването на уеб приложения
- Инструменти, които да подържат процеса ви на работа
- Добри практики от самото начало
- Инкрементално въведение на концепции
- Предвиден за скалируемост
- Автоматично (лесно) обновяване към нови версии

ОСНОВНИ ТЕХНОЛОГИИ

CLI

@angular/cli

Framework

@angular/core

В зависимост от нуждите ви:

- Примитиви за разработка на приложения (@angular/cdk)
- Рутер (@angular/router)
- Форми (@angular/forms)
- Компоненти (@angular/material)
- Progressive Web Apps (@angular/pwa)
- Рендериране на сървъра (@angular/ssr)
- ...

Не е нужно да знаете **ВСИЧКИ** тези
технологии, за да сте продуктивни

Концептуално въведение в Angular

- Компоненти (@Component)

Допълнителни концепции

...за по-добро разделяне на отговорностите

- Услуги (@Injectable)
- Директиви (@Directive)

Не е нужно да ги използвате от самото начало

Hello, world!

```
@Component({  
  selector: 'app-root',  
  template: 'Hello, world!'  
})  
export class App {}
```

Компоненти

- Имат селектор, който се използва за идентификацията им
- Имат темплейт и стилове, които се използват за рендериране на изгледа
- Могат да бъдат параметризирани (input)
- Могат да излъчват събития (output)

```
@Component({
  selector: 'grocery',
  template: '{{ title() }} <button (click)="delete.emit(title())">Delete</button>'
})
export class Grocery {
  title = input('');
  delete = output<string>();
}
```

```
@Component({
  imports: [Grocery],
  selector: 'grocery-list',
  template: `
    @for (g of groceries; track g) {
      <grocery
        [title]="g" (delete)="delete($event)">
      </grocery>
    }
  `
})
export class GroceryList {
  groceries = ['bananas', 'milk'];

  delete(item: string) {
    this.groceries.splice(this.groceries.indexOf(item), 1);
  }
}
```

```
@Component({
  selector: 'grocery',
  template: '{{ title() }} <button (click)="delete.emit(title())">Delete</button>'
})
export class Grocery {
  title = input('');
  delete = output<string>();
}
```

```
@Component({
  imports: [Grocery],
  selector: 'grocery-list',
  template: `
    @for (g of groceries; track g) {
      <grocery
        [title]="g" (delete)="delete($event)">
      </grocery>
    }
  `
})
export class GroceryList {
  groceries = ['bananas', 'milk'];

  delete(item: string) {
    this.groceries.splice(this.groceries.indexOf(item), 1);
  }
}
```

```
@Component({
  selector: 'grocery',
  template: `{{ title() }} <button (click)="delete.emit(title())">Delete</button>`
})
export class Grocery {
  title = input('');
  delete = output<string>();
}
```

```
@Component({
  imports: [Grocery],
  selector: 'grocery-list',
  template: `
    @for (g of groceries; track g) {
      <grocery
        [title]="g" (delete)="delete($event)">
      </grocery>
    }
  `
})
export class GroceryList {
  groceries = ['bananas', 'milk'];

  delete(item: string) {
    this.groceries.splice(this.groceries.indexOf(item), 1);
  }
}
```

Dependency injection

Dependency injection

```
@Injectable({
  providedIn: 'root'
})

export class Employee {
  delete(id: string): Promise<void> {
    return fetch( ... );
  }
}

@Component({
  selector: 'app-root',
  template: '<button (click)="delete()">Delete</button>'
})
export class AppComponent {
  private service = inject(Employee);

  delete() {
    this.service.delete( ... );
  }
}
```

Обобщение

- За създаване на приложения използвайте Angular CLI
- Приложенията са композиция от компоненти
- Компонентите имат:
 - Селектор
 - Темплейт и стилове
 - Параметри (input)
 - Събития (output)
- Декларирайте бизнес логика в услуги

За повече информация

angular.dev/tutorials/learn-angular



Демонстрация

Благодаря за вниманието :-)