JavaScript és Angular alapok

ES6 alapok · TypeScript · Angular Komponensek · Binding · Routing

Miért JavaScript + Angular?

- **JavaScript**: a web natív nyelve
- **TypeScript**: típusbiztonság, hibák megelőzése
- **Angular**: struktúrált, nagy projekteknél bevált keretrendszer
  - Egységes, robusztus keretrendszer
  - Nagy ökoszisztéma, rengeteg támogatás

ES6+ alapok (modern JavaScript)

- let és const - változók definiálása
- Arrow függvények
- Template literalok
- Destrukturálás
- Spread operátor
- Modulok

Rövid ES6 példák

```
const user = { name: "Anna", age: 25 };

const { name } = user;

const numbers = [1, 2, 3];
const more = [...numbers, 4, 5];

const double = x => x * 2;
```

```
let counter = 0;
counter++;
```

```
const name = 'Anna';
const message = `Hello ${name}!`;
```

Mi az a TypeScript?

- **JavaScript** superset (**kiterjesztett** változata)
- **Típusok** segítségével már fordításkor hibát jelez
- Fordítás JavaScript-re
- Angular teljesen erre épül
- Osztályok, interfészek, dekorátorok támogatása
- Kiszámíthatóbb kód

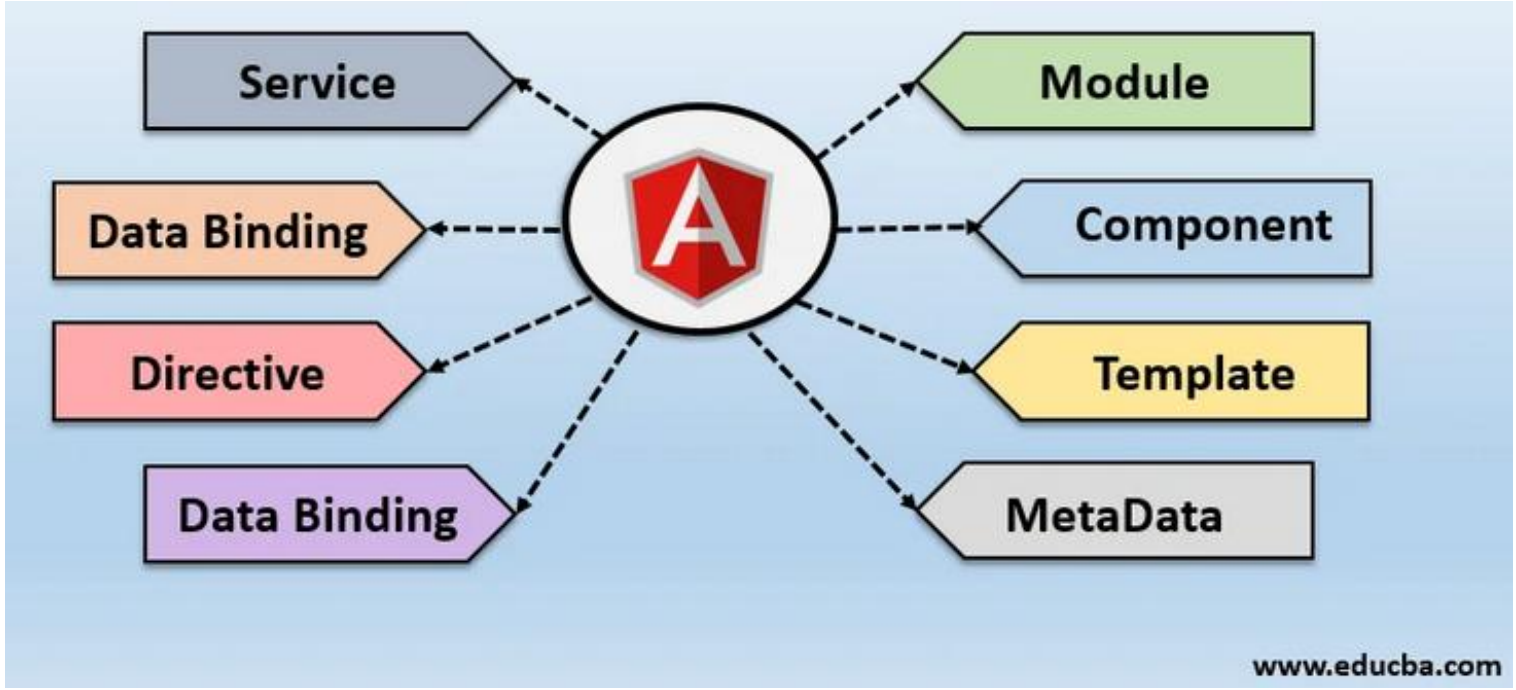
TypeScript példák

```typescript
let age: number = 25;

interface User {
  name: string;
  age: number;
}

function greet(user: User): string {
  return `Hello ${user.name}`;
}
```

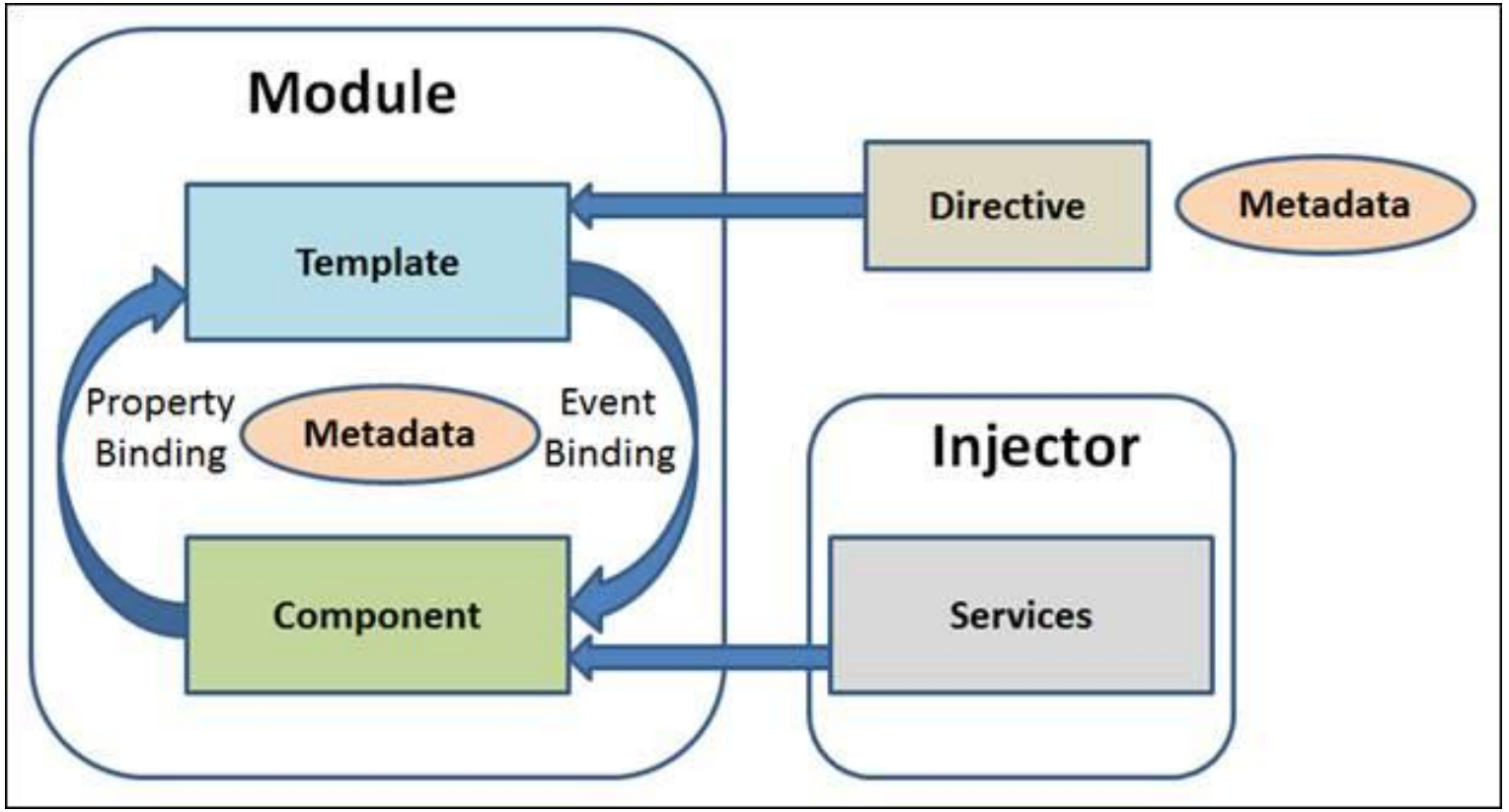
Angular architektúra áttekintés



- Komponens
- Modul
- Service
- Template
- MetaData (Decorator)
- Data Binding
- Directive
- Pipe
- Routing
- Dependency Injection

https://www.educba.com/angular-2-architecture/

Angular architektúra - diagram



Angular Component (komponens)

Az Angular **komponens** egy újrahasznosítható **építőelem**, amely a felhasználói felület egy részét **vezérli**, és a hozzá tartozó logikát, sablont és stílust egységbe foglalja.

Mit tartalmaz:

- **TypeScript** osztály
- **HTML template**
- **CSS** stílus
- **Decorator** (@Component)

Minden vizuális felület Angularban komponensekre van bontva.

Angular Component példa

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

@Component({
  selector: 'app-hello',
  template: `<h1>Hello {{ name }}</h1>`
})
export class HelloComponent {
  name = 'World';
}
```

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

@Component({
  selector: 'app-hello',
  templateUrl: './hello.component.html'
})
export class HelloComponent {
  name = 'World';
}
```

```html
<!-- hello.component.html -->
<h1>Hello {{ name }}</h1>
```

Data Binding típusok

- **Interpoláció** – {{ value }}
- **Property binding** – [property]="value"
- **Event binding** – (click)="doSomething()"
- **Two-way binding** – [(ngModel)]="value"

Routing alapok

- Single Page Application (**SPA**) működés
- **Routes**: Útvonalak konfigurálása
- **RouterModule**
- **RouterLink** a navigációhoz
- **router-outlet** a megjelenítéshez

Routing példa

```
const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'users', component: UsersComponent }
];
```

```
<nav>
  <a routerLink="/home">Home</a> |
  <a routerLink="/users">Users</a>
</nav>

<hr>

<router-outlet></router-outlet>
```

Service és Dependency Injection

- Üzleti logika és adatkezelés
- API hívások
- Számítások
- Állapotkezelés

A service-k egységesítik a komponenseket.

Dependency Injection: A service-k injektálására a komponensbe.

Service példa

```ts
// user.service.ts
import { Injectable } from '@angular/core';

@Injectable({ providedIn: 'root' })
export class UserService {
  getUsers(): string[] {
    return ['Alice', 'Bob', 'Charlie'];
  }
}
```

```ts
// users.component.ts
import { Component } from '@angular/core';
import { UserService } from './user.service';

@Component({
  selector: 'app-users',
  templateUrl: './users.component.html'
})
export class UsersComponent {
  users = this.userService.getUsers();

  constructor(private userService: UserService) {}
}
```

```html
<h2>Users</h2>

<ul>
  <li *ngFor="let u of users">
    {{ u }}
  </li>
</ul>
```

Összefoglalás

- **ES6** → modern JS
- **TypeScript** → hibamentesebb kód
- **Komponensek** → UI építőkockák
- **Data Binding** → adatkapcsolat
- **Routing** → navigáció
- **Szervizek** → logika és API hívások