Angular Service Injectors

Jim Cooper
Software Engineer

@jimthecoop | jcoop.io

Q: What are Service Injectors?

Injectors resolve dependencies and inject them into classes or components.

Injectors

```
@Injectable({ providedIn: 'root' })
class CartService { ... }
```

```
providers: [
  { provide: CartService, useFactory: () => { … } },
]
```

```
injector = Injector.create( providers );
```

Injectors

```
providers: [
  { provide: CartService, useFactory: () => {  cartService instance
]
```

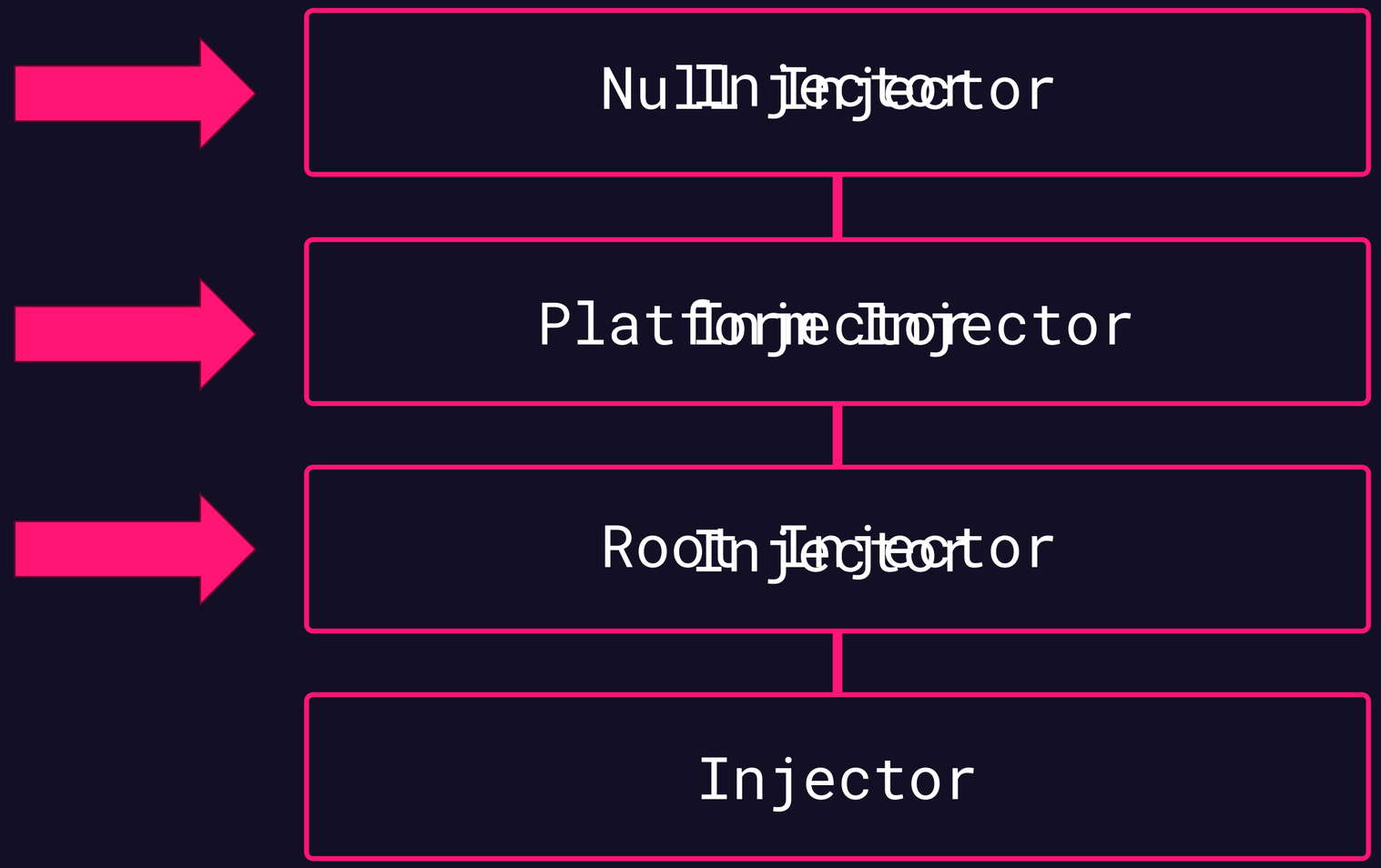
```
injector = Injector.create( providers );

class CatalogComponent {
  constructor (private  cartService instance  );
}

injector.get(CartService);
```

Injectors

```
providers: [
  { provide: CartService, useFactory: () => { … } },
  { provide: CartOptions, useValue: { ... } },
]
```

```
environmentInjector = Injector.create( providers );

class OtherComponent {
  constructor (private otherClass: OtherClass);
}
```

Hierarchical Injectors



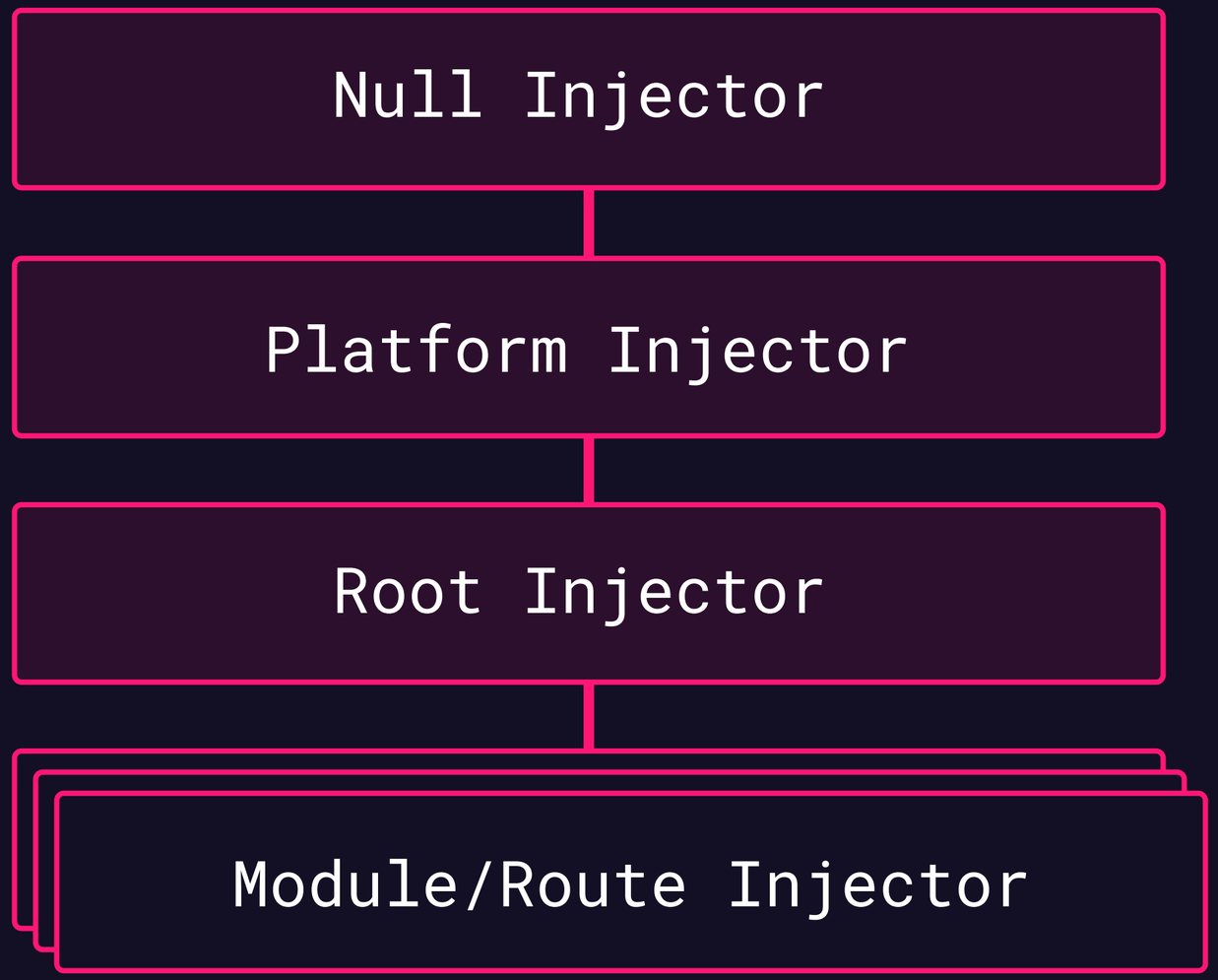
Hierarchical Injectors

Configuring Injectors

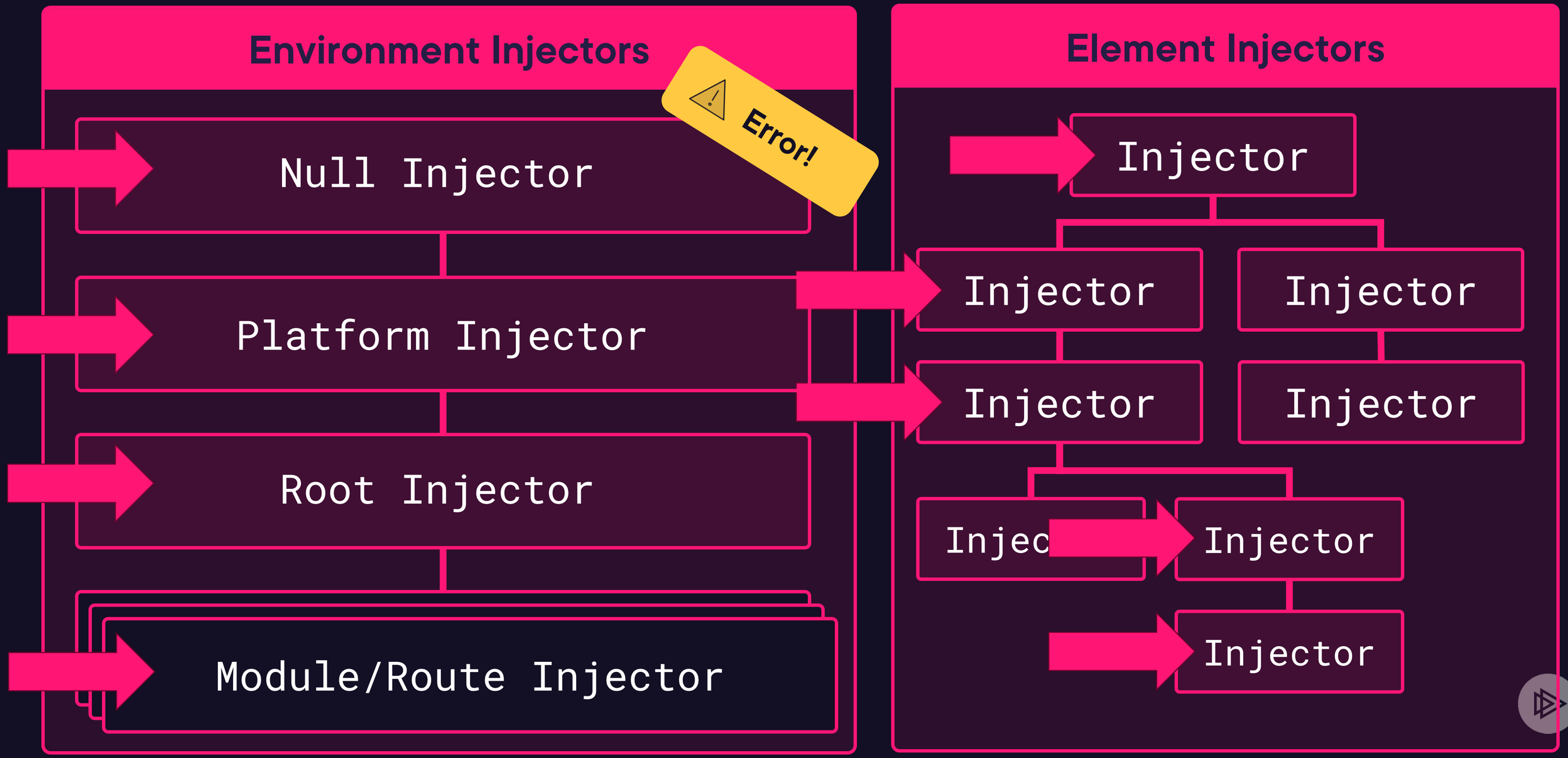
main.ts

```
// Modules
platformBrowserDynamic().bootstrapModule(AppModule);

  - or -

// Standalone project
bootstrapApplication(AppComponent, appConfig);
```

Hierarchical Injectors

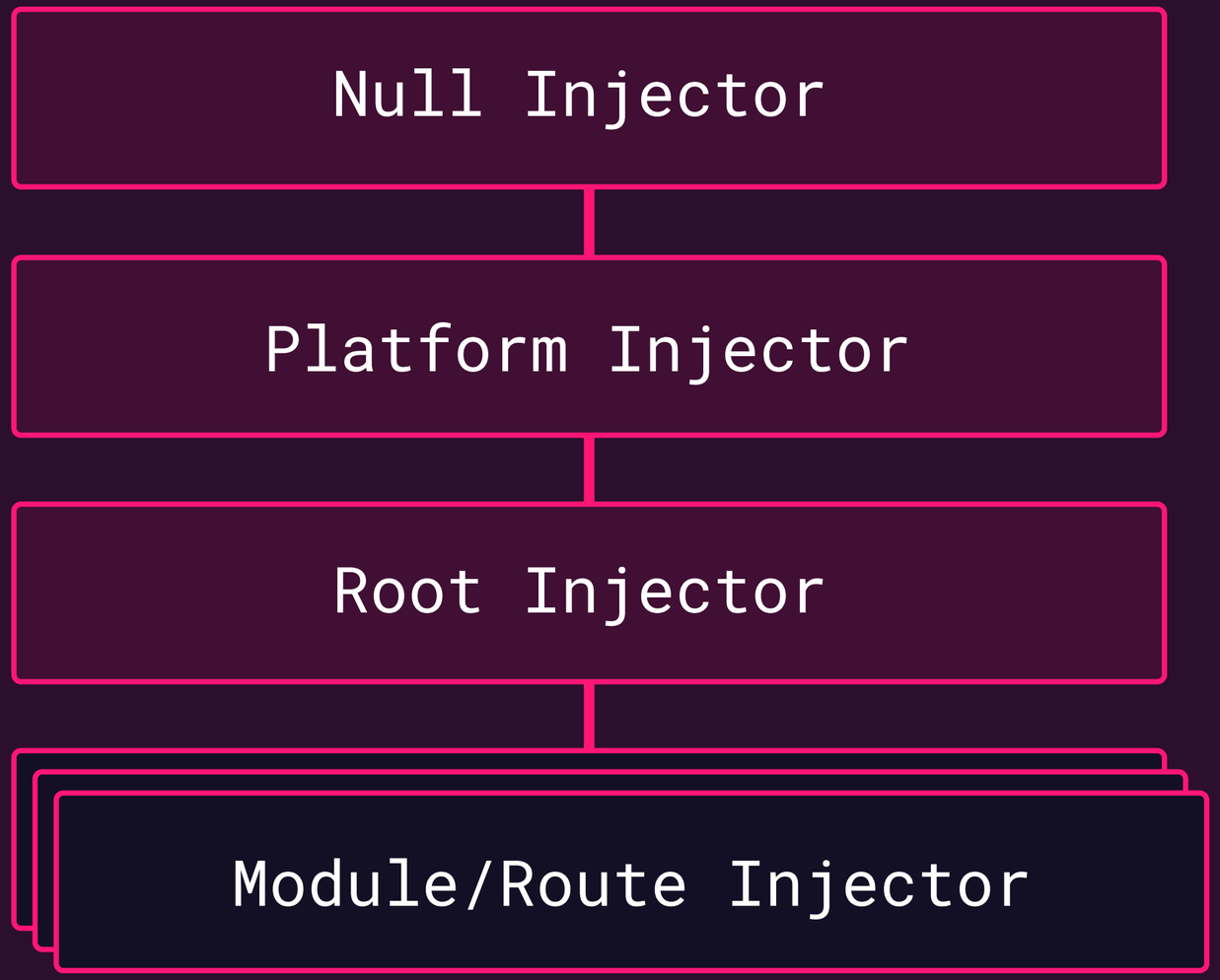


Hierarchical Injectors

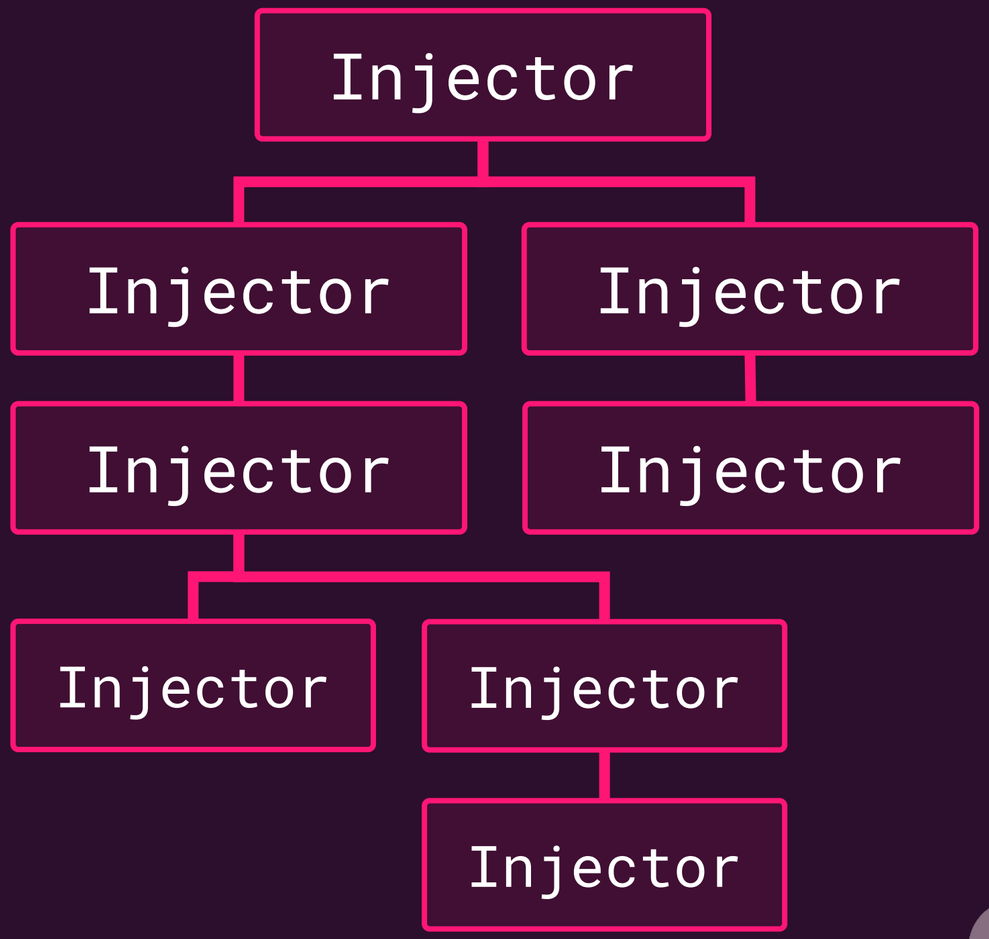


Configuring Injectors

Environment Injectors



Element Injectors

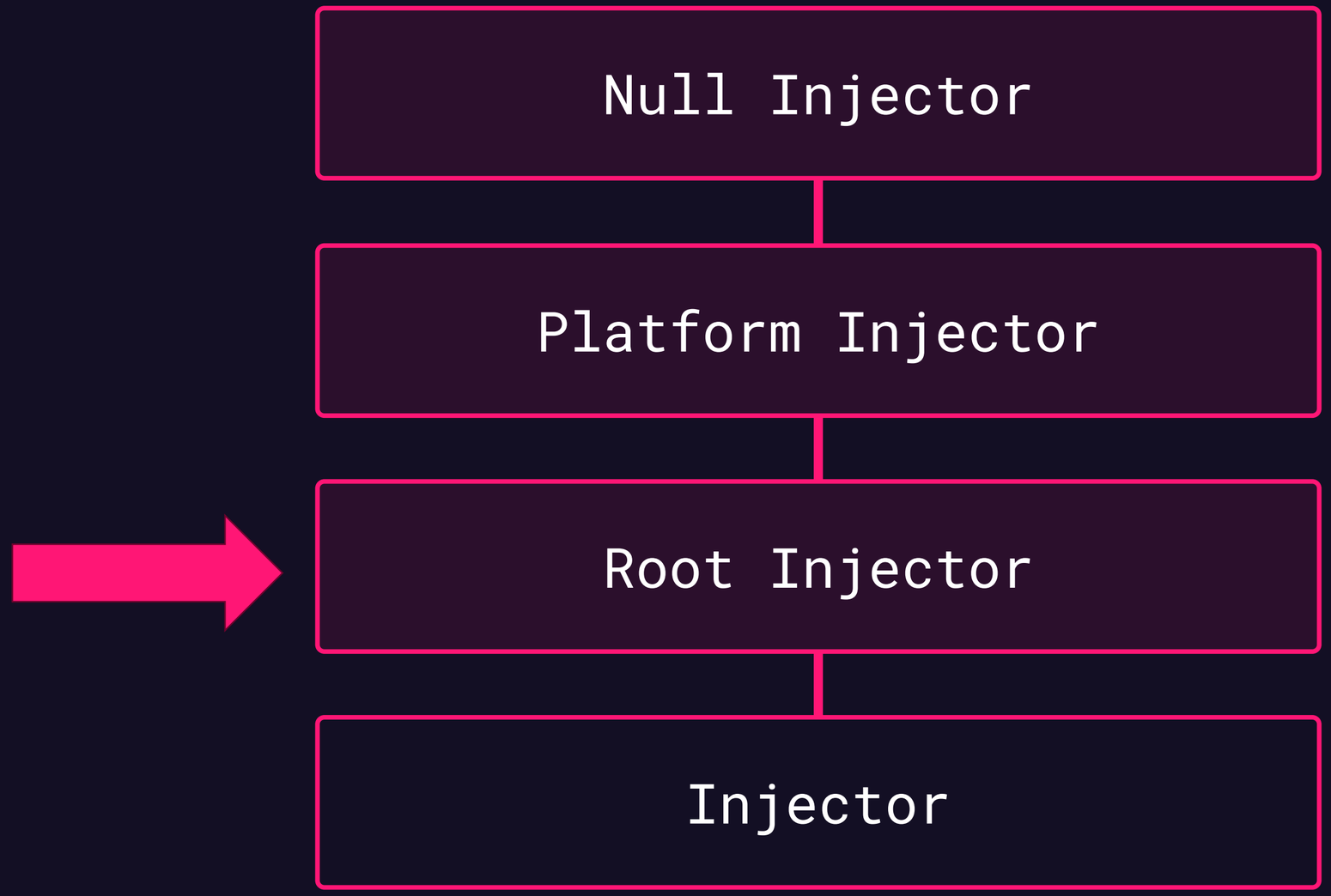


cart.service.ts - joes-robot-shop - Visual Studio Code

squad.module.ts | main.ts | cart.service.ts | app.module.ts

```
1  import { Inject, Injectable, InjectionToken, computed, signal } from "@angular/core";
2  import { Product } from "@shared/product.model";
3
4  export type CartOptions = {
5    persistenceType: string,
6    persistenceKey: string,
7  };
8
9  export const CART_OPTIONS_TOKEN = new InjectionToken<CartOptions>("CART_OPTIONS");
10
11 @Injectable({ providedIn: 'root' })
12 export class CartService {
13   private cartItems = signal<Product[]>([]);
14
15   constructor(@Inject(CART_OPTIONS_TOKEN) private cartOptions: CartOptions) {
16     if (this.cartOptions && this.cartOptions.persistenceType === 'local') {
17       const cartString = localStorage.getItem(this.cartOptions.persistenceKey);
18       const cart: Product[] = cartString ? JSON.parse(cartString) as Product[] : [];
19       this.cartItems.set(cart);
20     }
21   }
22
23   get cart() {
24     return this.cartItems.asReadonly();
25   }
26
27   add(product: Product) {
28     this.cartItems.update((oldCart) => [...oldCart, product]);
```

Configuring Injectors

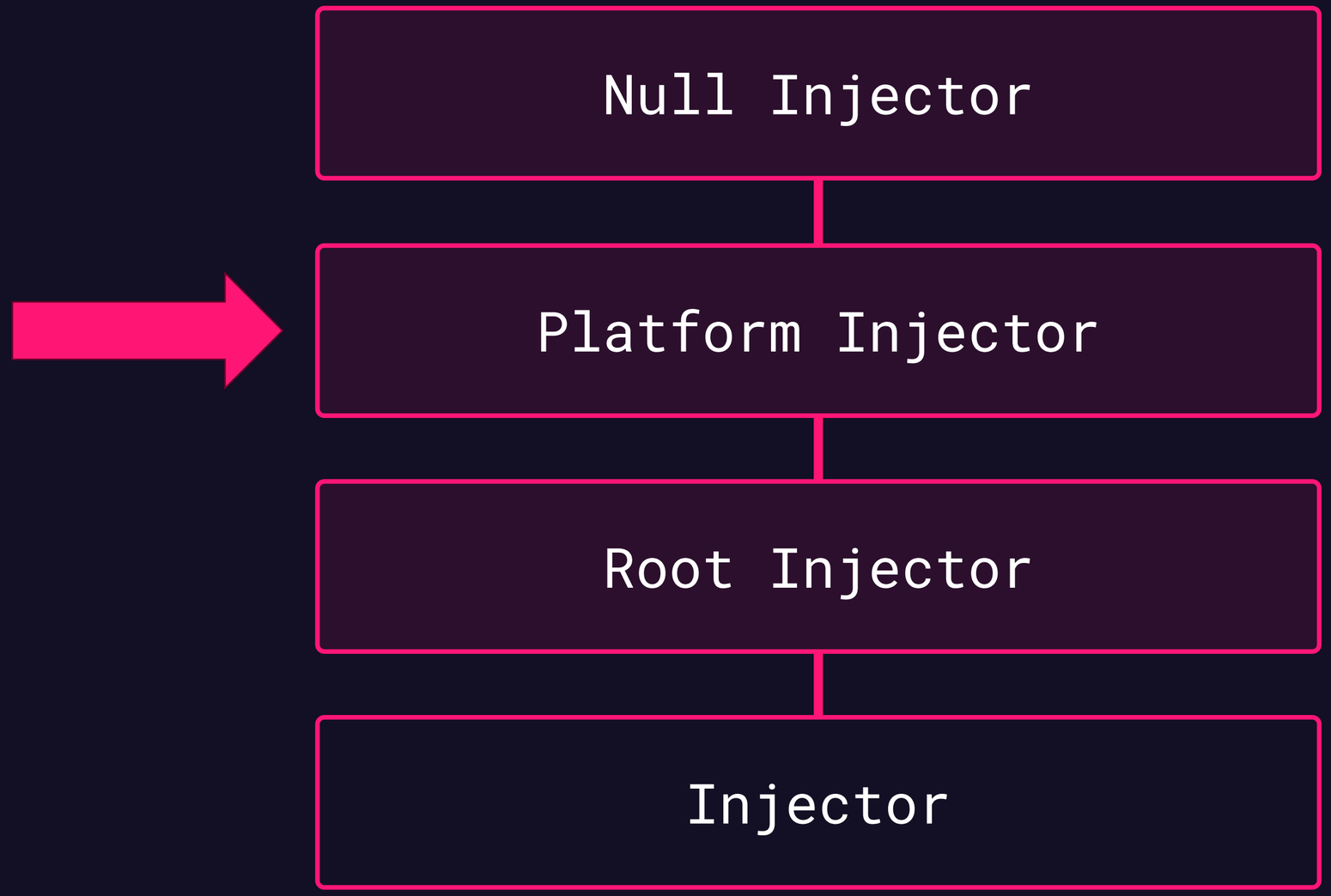


cart.service.ts - joes-robot-shop - Visual Studio Code

squad.module.ts | main.ts | cart.service.ts | app.module.ts

```
1  import { Inject, Injectable, InjectionToken, computed, signal } from "@angular/core";
2  import { Product } from "@shared/product.model";
3
4  export type CartOptions = {
5    persistenceType: string,
6    persistenceKey: string,
7  };
8
9  export const CART_OPTIONS_TOKEN = new InjectionToken<CartOptions>("CART_OPTIONS");
10
11 @Injectable({ providedIn: 'platform' })
12 export class CartService {
13   private cartItems = signal<Product[]>([]);
14
15   constructor(@Inject(CART_OPTIONS_TOKEN) private cartOptions: CartOptions) {
16     if (this.cartOptions && this.cartOptions.persistenceType === 'local') {
17       const cartString = localStorage.getItem(this.cartOptions.persistenceKey);
18       const cart: Product[] = cartString ? JSON.parse(cartString) as Product[] : [];
19       this.cartItems.set(cart);
20     }
21   }
22
23   get cart() {
24     return this.cartItems.asReadonly();
25   }
26
27   add(product: Product) {
28     this.cartItems.update((oldCart) => [...oldCart, product]);
```

Configuring Injectors

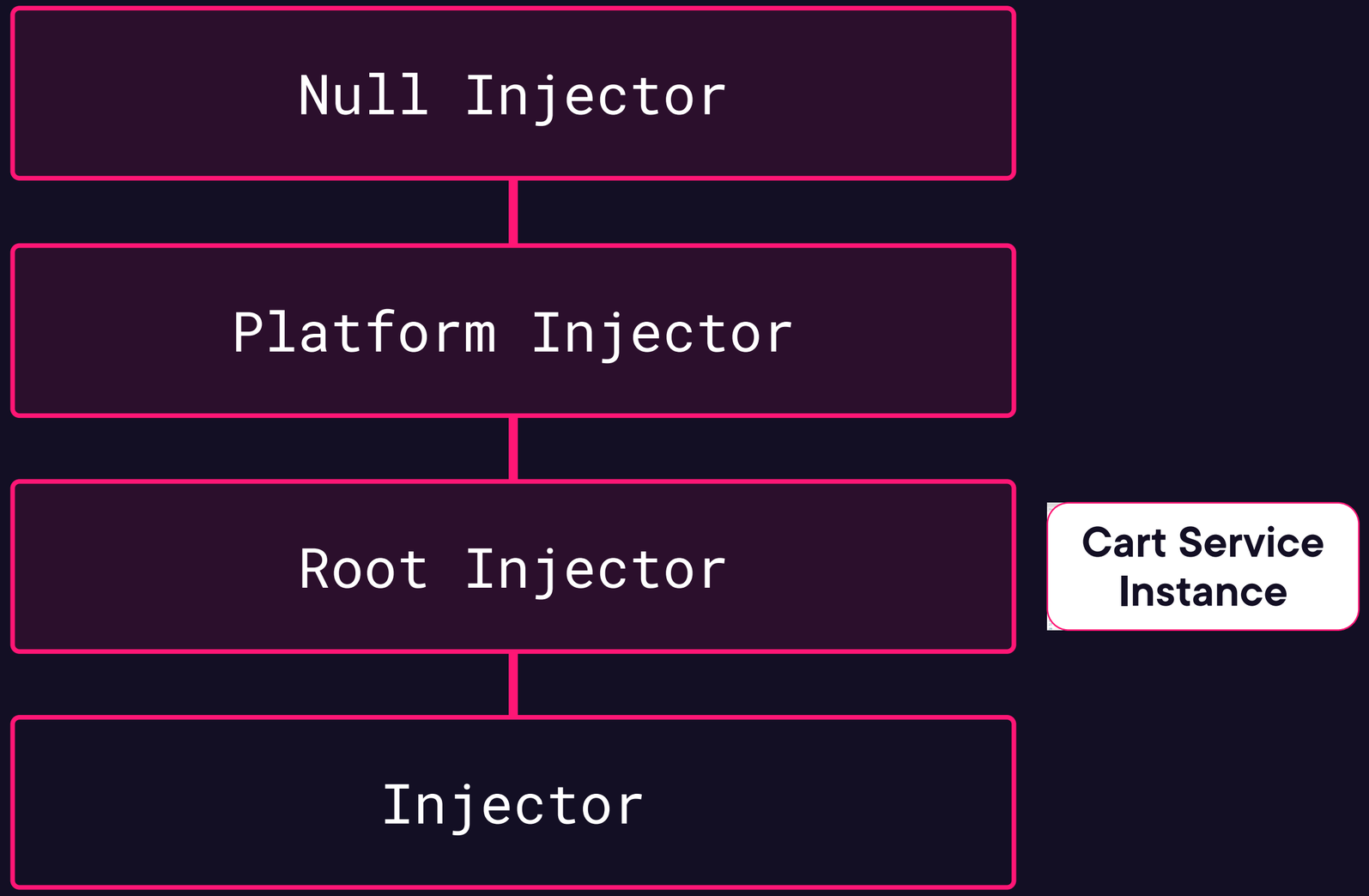


cart.service.ts - joes-robot-shop - Visual Studio Code

squad.module.ts | main.ts | cart.service.ts | app.module.ts

```
1  import { Inject, Injectable, InjectionToken, computed, signal } from "@angular/core";
2  import { Product } from "@shared/product.model";
3
4  export type CartOptions = {
5    persistenceType: string,
6    persistenceKey: string,
7  };
8
9  export const CART_OPTIONS_TOKEN = new InjectionToken<CartOptions>("CART_OPTIONS");
10
11 @Injectable({ providedIn: 'platform' })
12 export class CartService {
13   private cartItems = signal<Product[]>([]);
14
15   constructor(@Inject(CART_OPTIONS_TOKEN) private cartOptions: CartOptions) {
16     if (this.cartOptions && this.cartOptions.persistenceType === 'local') {
17       const cartString = localStorage.getItem(this.cartOptions.persistenceKey);
18       const cart: Product[] = cartString ? JSON.parse(cartString) as Product[] : [];
19       this.cartItems.set(cart);
20     }
21   }
22
23   get cart() {
24     return this.cartItems.asReadonly();
25   }
26
27   add(product: Product) {
28     this.cartItems.update((oldCart) => [...oldCart, product]);
```

Configuring Injectors

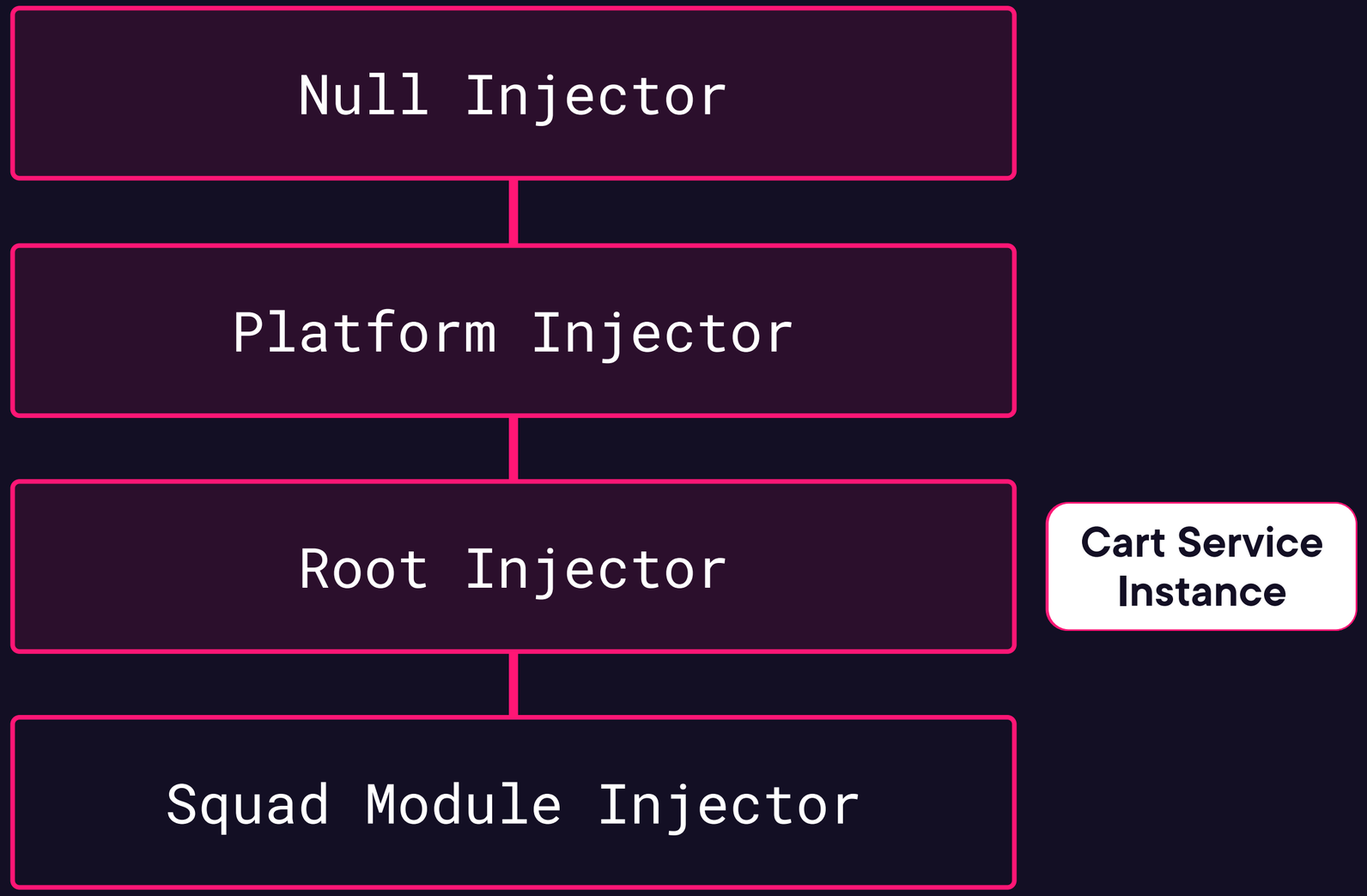


squad.module.ts - joes-robot-shop - Visual Studio Code

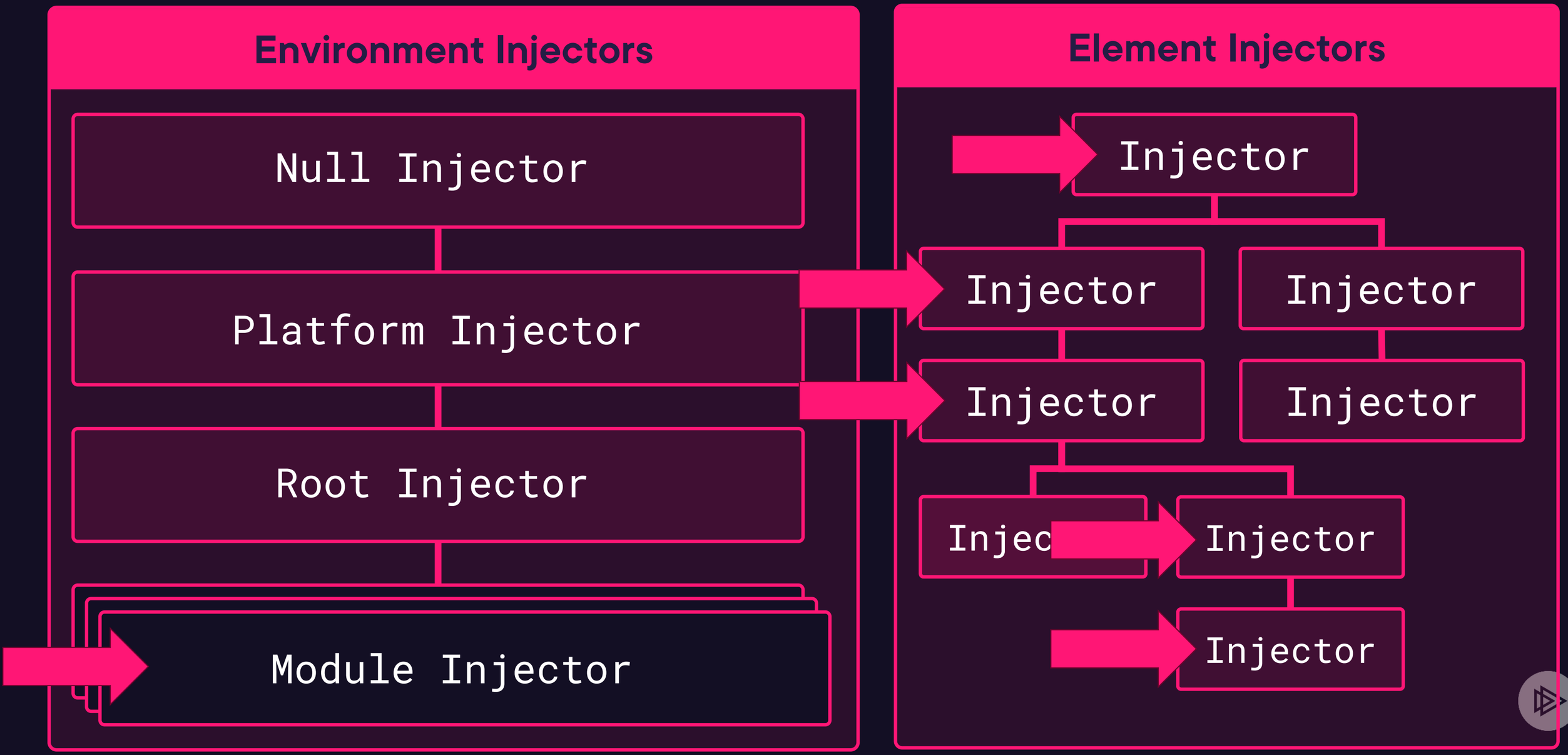
TS squad.module.ts × | TS main.ts | TS cart.service.ts | TS app.module.ts

```
5  import { CART_OPTIONS_TOKEN, CartOptions, CartService } from '@catalog/cart.service';
6
7  @NgModule({
8    declarations: [SquadCatalogComponent],
9    imports: [SharedModule, SquadRoutingModule],
10   providers: [
11     {
12       provide: CartService,
13       useFactory: (cartOptions: any) => { return new CartService(cartOptions); },
14       deps: [CART_OPTIONS_TOKEN],
15       multi: false,
16     },
17     {
18       provide: CART_OPTIONS_TOKEN,
19       useValue: { persistenceType: 'local', persistenceKey: 'squad-cart' },
20       multi: false,
21     },
22   ],
23 })
24 export class SquadModule { }
25
```

Configuring Injectors



Configuring Injectors

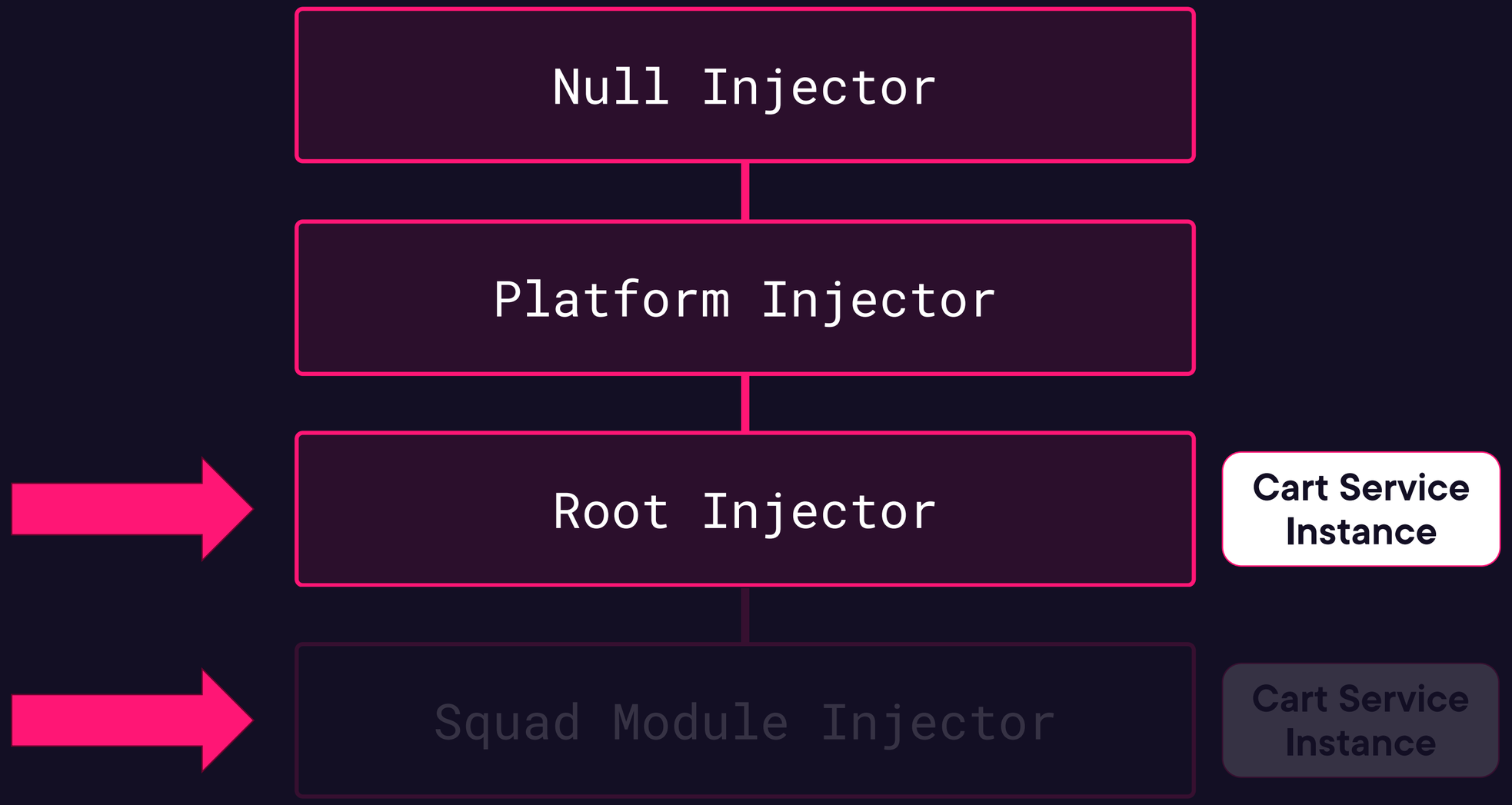


squad.module.ts - joes-robot-shop - Visual Studio Code

TS squad.module.ts × | TS main.ts | TS cart.service.ts | TS app.module.ts

```
5  import { CART_OPTIONS_TOKEN, CartOptions, CartService } from '@catalog/cart.service';
6
7  @NgModule({
8    declarations: [SquadCatalogComponent],
9    imports: [SharedModule, SquadRoutingModule],
10   providers: [
11     {
12       provide: CartService,
13       useFactory: (cartOptions: any) => { return new CartService(cartOptions); },
14       deps: [CART_OPTIONS_TOKEN],
15       multi: false,
16     },
17     {
18       provide: CART_OPTIONS_TOKEN,
19       useValue: { persistenceType: 'local', persistenceKey: 'squad-cart' },
20       multi: false,
21     },
22   ],
23 })
24 export class SquadModule { }
25
```

Configuring Injectors



cart.service.ts - joes-robot-shop - Visual Studio Code

squad.module.ts | main.ts | cart.service.ts | app.module.ts

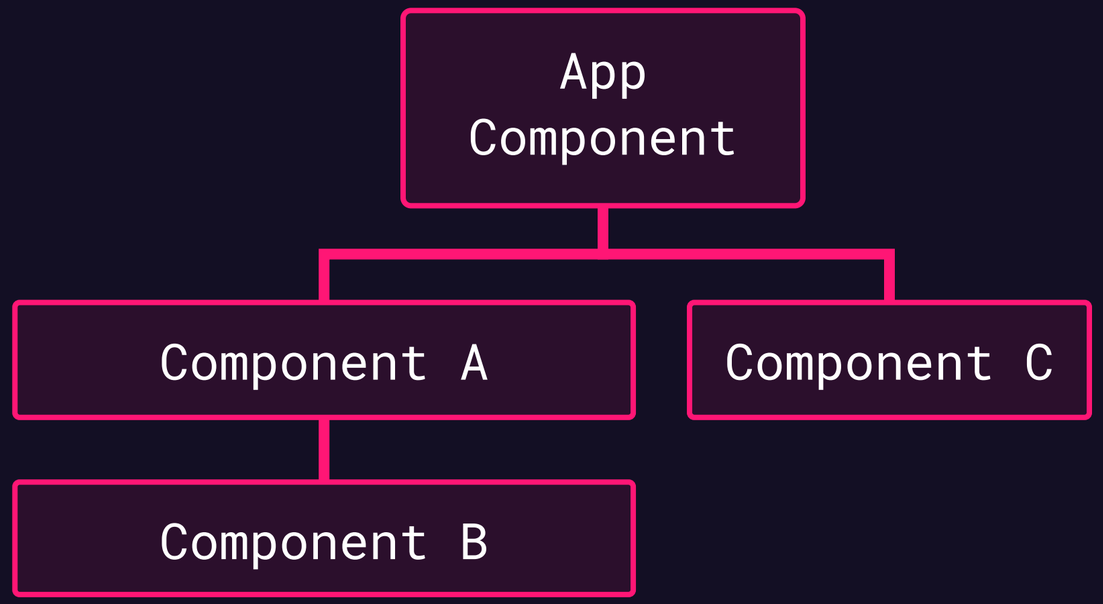
```
1  import { Inject, Injectable, InjectionToken, computed, signal } from "@angular/core";
2  import { Product } from "@shared/product.model";
3
4  export type CartOptions = {
5    persistenceType: string,
6    persistenceKey: string,
7  };
8
9  export const CART_OPTIONS_TOKEN = new InjectionToken<CartOptions>("CART_OPTIONS");
10
11 @Injectable({ providedIn: 'root' })
12 export class CartService {
13   private cartItems = signal<Product[]>([]);
14
15   constructor(@Inject(CART_OPTIONS_TOKEN) private cartOptions: CartOptions) {
16     if (this.cartOptions && this.cartOptions.persistenceType === 'local') {
17       const cartString = localStorage.getItem(this.cartOptions.persistenceKey);
18       const cart: Product[] = cartString ? JSON.parse(cartString) as Product[] : [];
19       this.cartItems.set(cart);
20     }
21   }
22
23   get cart() {
24     return this.cartItems.asReadonly();
25   }
26
27   add(product: Product) {
28     this.cartItems.update((oldCart) => [...oldCart, product]);
```

squad.module.ts - joes-robot-shop - Visual Studio Code

TS squad.module.ts × | TS main.ts | TS cart.service.ts | TS app.module.ts

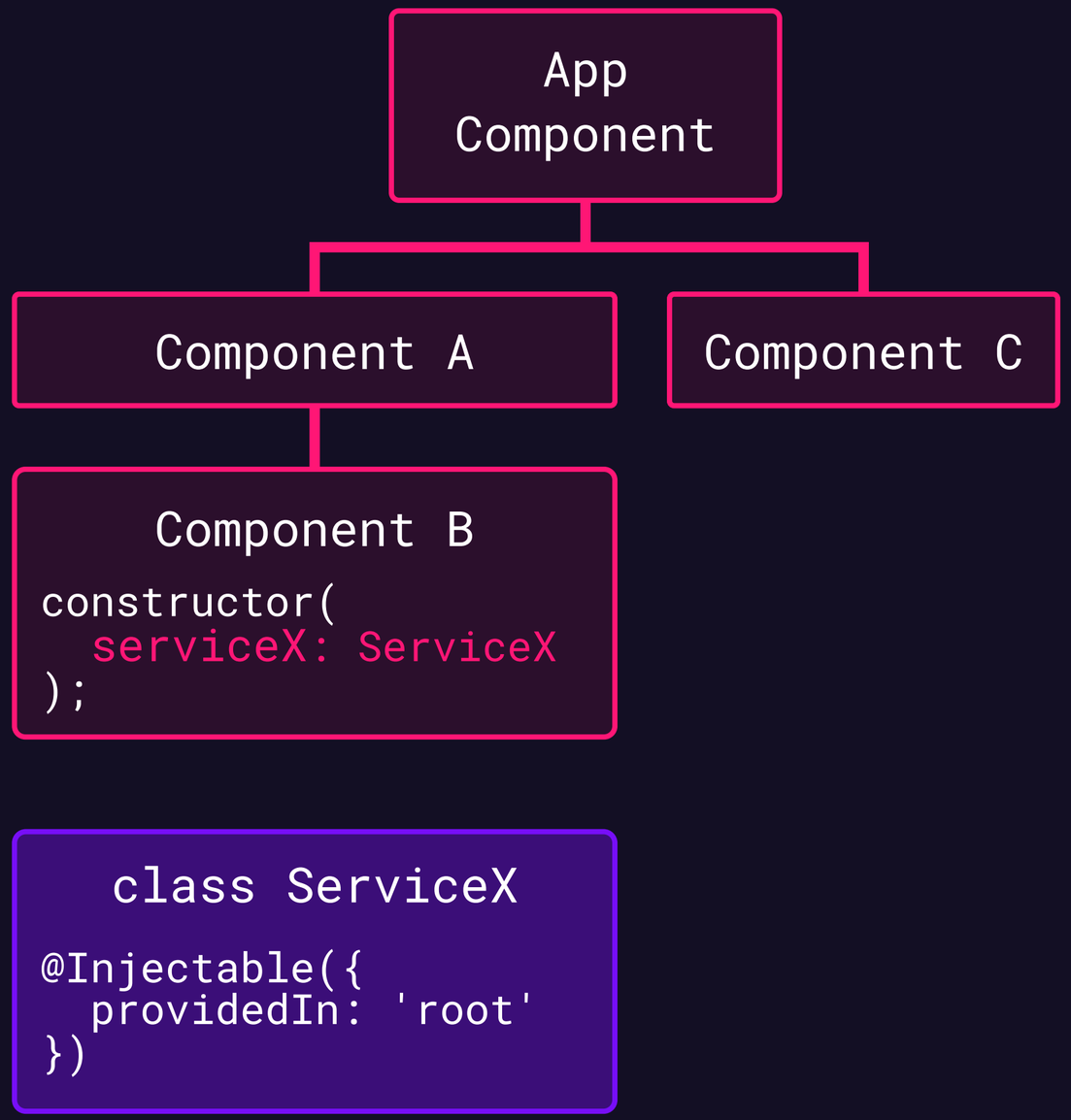
```
5  import { CART_OPTIONS_TOKEN, CartOptions, CartService } from '@catalog/cart.service';
6
7  @NgModule({
8    declarations: [SquadCatalogComponent],
9    imports: [SharedModule, SquadRoutingModule],
10   providers: [
11     {
12       provide: CartService,
13       useFactory: (cartOptions: any) => { return new CartService(cartOptions); },
14       deps: [CART_OPTIONS_TOKEN],
15       multi: false,
16     },
17     {
18       provide: CART_OPTIONS_TOKEN,
19       useValue: { persistenceType: 'local', persistenceKey: 'squad-cart' },
20       multi: false,
21     },
22   ],
23 })
24 export class SquadModule { }
25
```

A Graphical Look at Injectors

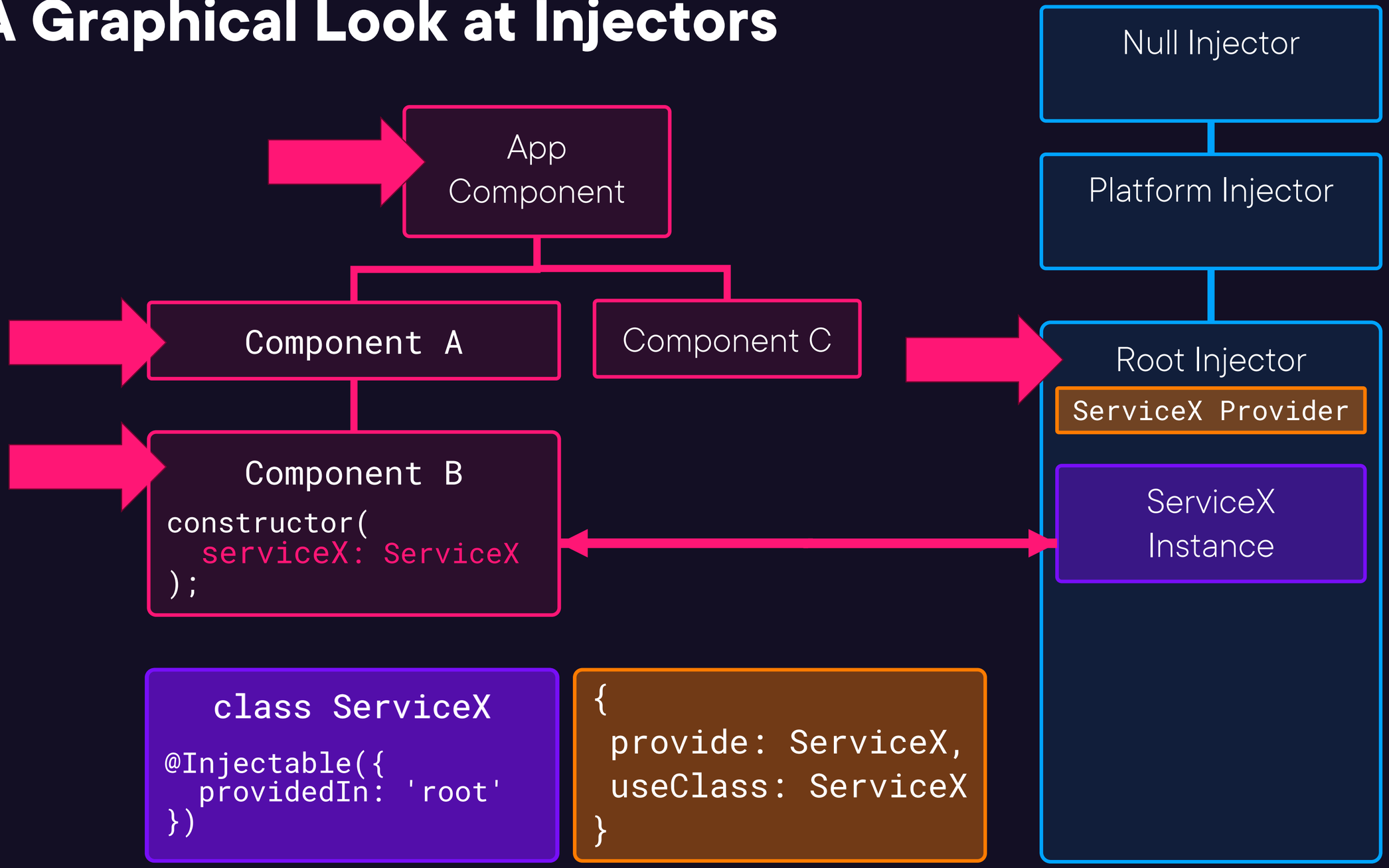



```
class ServiceX

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

A Graphical Look at Injectors



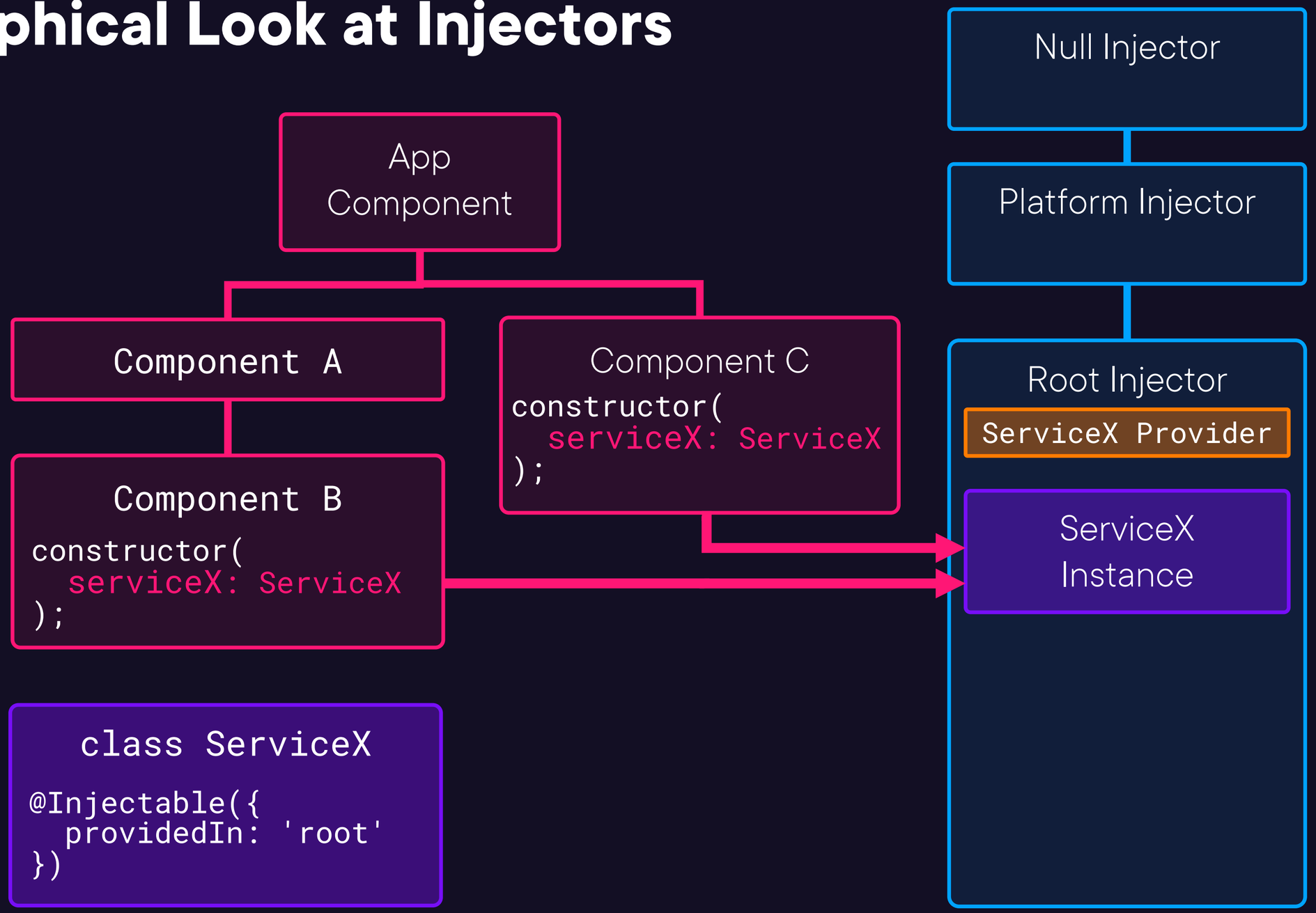
A Graphical Look at Injectors



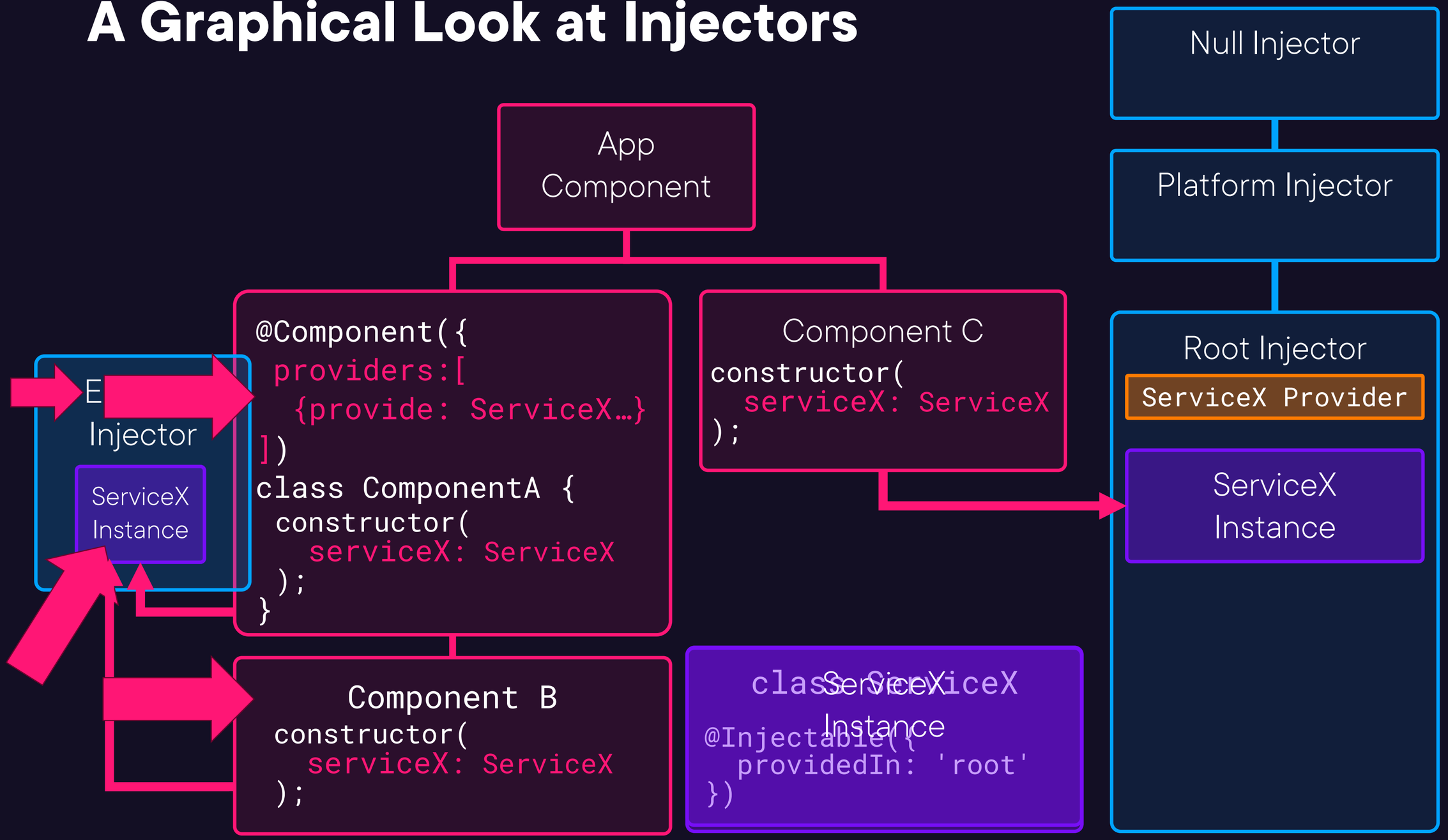

```
class ServiceX

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

```
{
 provide: ServiceX,
 useClass: ServiceX
}
```

A Graphical Look at Injectors



A Graphical Look at Injectors



A Graphical Look at Injectors

App Component

ComponentA

```
constructor(
  serviceX: ServiceX
);
```

Component C

```
constructor(
  serviceX: ServiceX
);
```

Component B

Null Injector

Platform Injector

Root Injector

ServiceX Provider

ServiceX Instance

```
class ServiceX

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

A Graphical Look at Injectors

App Component

ComponentA

```
constructor(
  serviceX: ServiceX
);
```

Component B

```
class ServiceX

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

Eagerly-loaded Feature Module

```
providers:[
  {provide: ServiceX…}
]
```

Component C

```
constructor(
  serviceX: ServiceX
);
```

Null Injector

Platform Injector

Root Injector

ServiceX Provider

ServiceX Instance

A Graphical Look at Injectors

App Component

ComponentA

```
constructor(
  serviceX: ServiceX
);
```

Component B

```
class ServiceX

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

Lazily-loaded Feature Module

```
providers:[
  {provide: ServiceX…}
]
```

Component C

```
constructor(
  serviceX: ServiceX
);
```

Null Injector

Platform Injector

Root Injector

ServiceX Provider

ServiceX Instance

A Graphical Look at Injectors

App Component

ComponentA

```
constructor(
  serviceX: ServiceX
);
```

Component B

ServiceX Instance

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

Lazily-loaded Feature Module

```
providers:[
  {provide: ServiceX…}
]
```

Component C

```
constructor(
  serviceX: ServiceX
);
```

Null Injector

Platform Injector

Root Injector

ServiceX Provider

ServiceX Instance

Module Injector

ServiceX Instance

@Host Modifier

```
<ComponentA>
  <ComponentB />
</ComponentA>
```

```
@Component(...)
class ComponentB {
  constructor(
    private serviceX:ServiceX
  ) { }
}
```

```
<ComponentA myDirective>

</ComponentA>
```

```
@Directive(...)
class MyDirective {
  constructor(
    private serviceX:ServiceX
  ) { }
}
```

@Host Modifier

```
<ComponentA>
  <ComponentB />
</ComponentA>
```

```
@Component(...)
class ComponentB {
  constructor(
    @Host() private serviceX:ServiceX
  ) { }
}
```

```
<ComponentA myDirective>

</ComponentA>
```

```
@Directive(...)
class MyDirective {
  constructor(
    @Host() private serviceX:ServiceX
  ) { }
}
```

A Graphical Look at Injectors

App Component

ComponentA

```
constructor(
  serviceX: ServiceX
);
```

Component B

ServiceX Instance

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

Lazily-loaded Feature Module

```
providers:[
  {provide: ServiceX…}
]
```

Component C

```
constructor(
  serviceX: ServiceX
);
```

Null Injector

Platform Injector

Root Injector

ServiceX Provider

ServiceX Instance

Module Injector

ServiceX Instance