ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

```
C:\ng new_
```

Angulars Renaissance

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Hi, I'm...

Kobi Hari

- Freelancer
- Developer, Instructor and Consultant
- Angular, Async development, .NET core

hari@applicolors.com

https://github.com/kobi2294

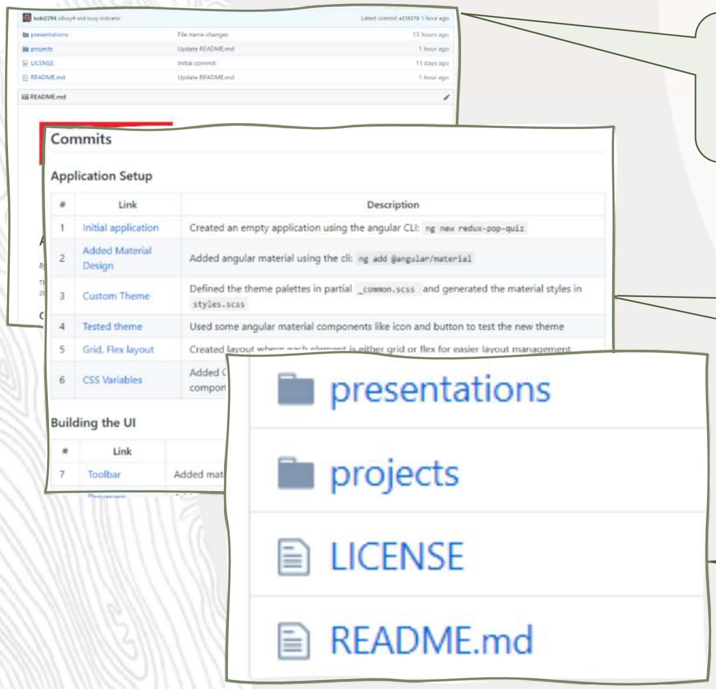
https://www.youtube.com/@kobihari-applicolors1464

ORACLE TECH DAYS

ORACLE | JOHN BRYCE תלמדו הייטק. זה עובד! a matrix company

We have a GitHub Repository!

kobi-hari-courses/2312-oracle-tech-days-ngnew



presentations	File name changes	13 hours ago
projects	Update README.md	1 hour ago
LICENSE	Initial commit	11 days ago
README.md	Update README.md	1 hour ago

README.md

Commits

Application Setup

#	Link	Description
1	Initial application	Created an empty application using the angular CLI: `ng new redux-pop-quiz`
2	Added Material Design	Added angular material using the cli: `ng add @angular/material`
3	Custom Theme	Defined the theme palettes in partial `_common.scss` and generated the material styles in `styles.scss`
4	Tested theme	Used some angular material components like icon and button to test the new theme
5	Grid, Flex layout	Created layout where each element is either grid or flex for easier layout management
6	CSS Variables	Added [illegible]

Building the UI

#	Link	
7	Toolbar	Added mat[illegible]

presentations

projects

LICENSE

README.md

Session Summary

Project Code with Commits Summary

This presentation and other Useful Links

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Our Agenda

- inject() – The new Dependency Injection
- Standalone – Apps without modules
- Routing - Redirected
- Signals – The new Reactivity

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

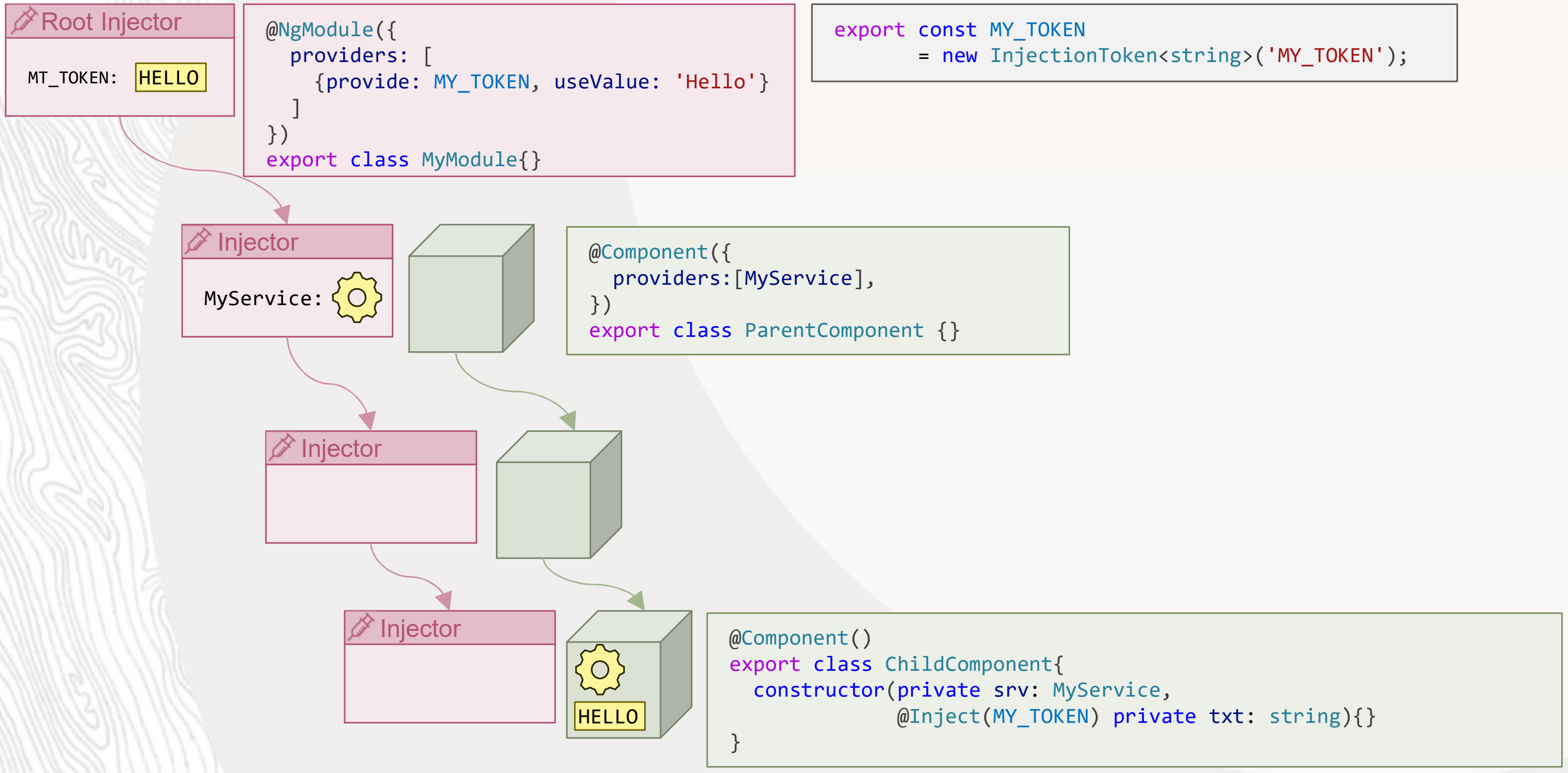
inject()

The new Dependency Injection

ORACLE TECH DAYS

ORACLE JOHN BRYCE תלמדו הייטק. זה עובד! a matrix company

Dependency Injection - Recap



```
@NgModule({
  providers: [
    {provide: MY_TOKEN, useValue: 'Hello'}
  ]
})
export class MyModule{}
```

```
export const MY_TOKEN
       = new InjectionToken<string>('MY_TOKEN');
```

```
@Component({
  providers:[MyService],
})
export class ParentComponent {}
```

```
@Component()
export class ChildComponent{
  constructor(private srv: MyService,
              @Inject(MY_TOKEN) private txt: string){}
}
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

The Injector

```
@Component({
  providers: [{provide: MY_NUMBER, useValue: 42}]
})
export class AppComponent {

  constructor(private injector1: Injector) {
    const injector2 = Injector.create({
      providers: [{provide: MY_TOKEN, useValue: 'WORLD'}],
      parent: injector1
    });

    const val = injector2.get(MY_TOKEN);  // val === 'WORLD'
    const num = injector2.get(MY_NUMBER); // num === 42
  }
}
```

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

inject()

ng old (<14)

```
@Component()
export class ChildComponent{
  constructor(private srv: MyService,
              @Inject(MY_TOKEN) private txt: string){}
}
```

ng new (14)

```
@Component()
export class ChildComponent{
  private srv = inject(MyService);
  private txt = inject(MY_TOKEN);
}
```

ORACLE
TECH DAYS

Injection Context

- The `inject()` function only works in injection context:
  - Inside the constructor
  - Field initializers
  - Factory function of provider

```
@Component()
export class ChildComponent{
  private str = inject(MY_TOKEN); // will work

  constructor() {
    const txt = inject(MY_TOKEN); // will work
  }
}
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
!תלמדו הייטק. זה עובד
a matrix company

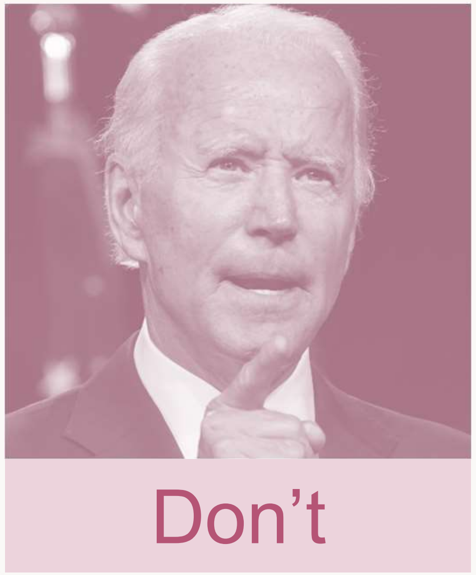
Can you use `inject` in these cases?

```
 // in property setters
set myProp(val: string) {
    inject(Service).refresh();
}

constructor() {
    // in subscribe body
    of(true).subscribe(val => {
      const txt2 = inject(MY_TOKEN);  // error
    });

    // in promise continuation
    Promise.resolve(true).then(val => {
      const txt3 = inject(MY_TOKEN); // error
    })
}

   // in lifecycle hooks
ngOnInit() {
    const txt = inject(MY_TOKEN); // error
}
```



run-In-Injection-Context()

- You can run functions that use inject() in other places by:
  - Storing the `Injector`
  - Using the `runInInjectionContext` function

```
@Component()
export class ChildComponent{
  private injector = inject(Injector);

  ngOnInit() {
    runInInjectionContext(this.injector, () => {
      const num = inject(MY_TOKEN);
    })
  }
}
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Demo – `inject()` function

- ✓ Using inject() in components
- ✓ Creating self-injecting utility functions
- ✓ Creating our own injectors
- ✓ Using run-in-injection-context()
- ✓ The async-await trap

Destroy Ref

- You can now inject “DestroyRef” to components, services, directives, pipes…
  - This replaces the need for “OnDestroy”
- If you want to auto-complete observables – you can use the takeUntilDestroyed operator
  - Note that it uses inject() to inject the DestroyRef
  - So you may only use this operator in injection context
  - Alternatively, you may use it anywhere and pass the DestroyRef as parameter

```
export class SomeComponent {
  constructor(private destroyRef: DestroyRef) {
    destroyRef.onDestroy(() =>
      console.log('Destroyed'));

    interval().pipe(
      takeUntilDestroyed()
    ).subscribe(val => console.log(val));
  }

  notInjectionContext() {
    interval().pipe(
      takeUntilDestroyed(this.destroyRef)
    ).subscribe(val => console.log(val))
  }
}
```

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

standalone

Apps without modules

ORACLE TECH DAYS

ORACLE JOHN BRYCE תלמדו הייטק. זה עובד! a matrix company

Standalone Components

ng old (<14)

```
@NgModule({
    declarations: [StandaloneComponent],
    exports: [StandaloneComponent]
})
export class SomeModule {}

@Component({
  selector: 'app-standalone',
  templateUrl: './standalone.component.html',
})
export class StandaloneComponent {}
```

ng new (14)

```
@Component({
  standalone: true,
  selector: 'app-standalone',
  templateUrl: './standalone.component.html',
})
export class StandaloneComponent {}
```

ORACLE TECH DAYS

ORACLE JOHN BRYCE תלמדו הייטק. זה עובד! a matrix company

Imports

- You can import into standalone components
  - Modules
  - Other standalones
- Modules can import standalones

- Standalone components are like a component and module in the same object
- Directives and Pipes can also be standalone

```
@Component({
  standalone: true,
  selector: 'app-standalone',
  templateUrl: './standalone.component.html',
  imports: [CommonModule, OtherStandaloneComponent]
})
export class StandaloneComponent {}
```

```
@NgModule({
    imports: [
        StandaloneComponent,
        OtherStandaloneComponent
    ]
})
export class SomeModule {}
```

ORACLE TECH DAYS

ORACLE

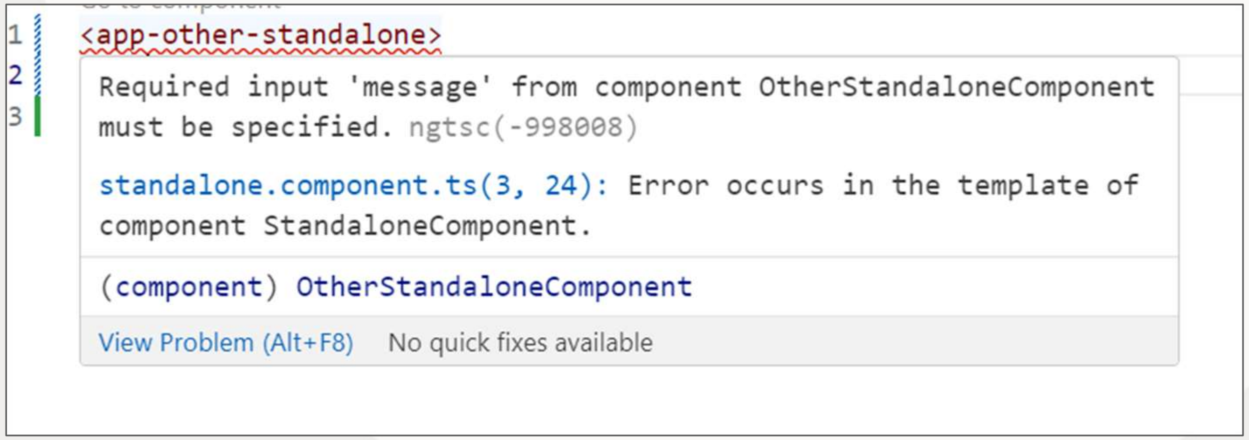
JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Required Inputs

- Angular 16 Also allow to define input as mandatory.

```
export class OtherStandaloneComponent {
  @Input({alias: 'message', required: true})
  txtMessage!: string;
}
```

- The angular compiler will show an error message if the input is not specified



```
<app-other-standalone>
Required input 'message' from component OtherStandaloneComponent
must be specified. ngtsc(-998008)
standalone.component.ts(3, 24): Error occurs in the template of
component StandaloneComponent.
(component) OtherStandaloneComponent
View Problem (Alt+F8)    No quick fixes available
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Applications without modules

- The roles of the module
  - Declare, Import and export angular template “objects”
    - Components
    - Directives
    - Pipes
  - Define Dependency Injection Providers
- Standalone components replace the first role
- But what replaces the second one?

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Bootstrap

ng old (<15)

```
// main.ts
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));

// app.module.ts
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  bootstrap: [AppComponent],
  providers: [
    {provide: MY_TOKEN, useValue: 42},
    {provide: MyService, useClass: MyOtherService}
  ]
})
export class AppModule{}
```

ng new (15)

```
// main.ts
bootstrapApplication(AppComponent, appConfig)
  .catch((err) => console.error(err));

// app.config.ts
export const appConfig: ApplicationConfig = {
  providers: [
    {provide: MY_TOKEN, useValue: 42},
    {provide: MyService, useClass: MyOtherService}
  ]
};
```

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Importing providers

ng old (<15)

```
// app.module.ts
  @NgModule({
  imports: [
    HttpClientModule,
    BrowserAnimations,
    RouterModule.forRoot(routes),
    ModuleWithProviders
  ],
})
```

ng new (15)

```
export const appConfig: ApplicationConfig = {
  providers: [
    provideHttpClient(),
    provideRouter(routes),
    provideAnimations(),
    importProvidersFrom(ModuleWithProviders)
  ]
};
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Demo – Standalone apps

- ✓ ng new --standalone
- ✓ Bootstrapping an application
- ✓ Defining root providers
- ✓ Providing Http, Animations
- ✓ Providing the router
- ✓ Providing other modules

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Routing

Redirected

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

No routing module

ng old (<15)

```
// app-routing.module.ts
const routes: Routes = [
  {path: 'home', component: HomeComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

ng new (15)

```
// app.routes.ts
export const routes: Routes = [
    { path: 'home', component: HomeComponent }
];
```

ORACLE TECH DAYS

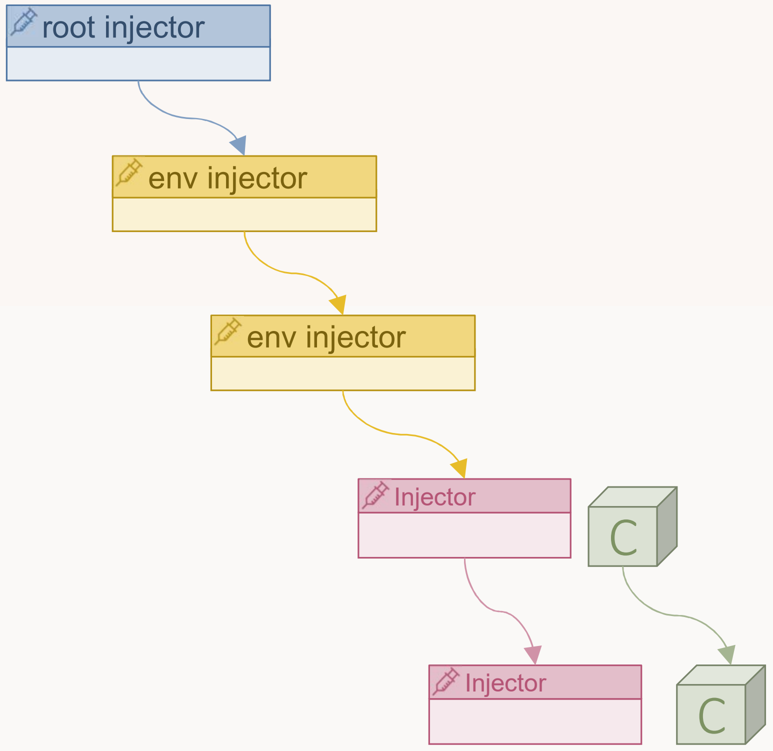
ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Environment Injectors

- You can define providers that will be applicable to subset of routes
- This creates an additional Hierarchy of injectors
- Once the component Injectors are exhausted, The environment injector Hierarchy is used

```
// app.routes.ts
export const routes: Routes = [
    {
        path: 'home', component: HomeComponent,
        providers: [{
            provide: MY_TOKEN, useValue: 100
        }],
        children: [
            {path: 'page-a', component: PageAComponent},
            {path: 'page-b', component: PageBComponent}
        ]
}];
```



ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Lazy Loading

- You can load one standalone component

```
// app.routes.ts
export const routes: Routes = [
    {
        path: 'courses',
        loadComponent: () => import('./courses/courses.component')
                             .then(m => m.CoursesComponent)
    }
];
```

- Or you can load a set of routes

```
// app.routes.ts
export const routes: Routes = [
    {
        path: 'courses',
        loadChildren: () => import('./courses/courses.routes')
                            .then(m => m.coursesRoutes)
    }
];

// courses.routes
export const coursesRoutes: Route[] = [
    { path: 'list', component: CourseListComponent},
    { path: 'top', component: TopCourseComponent}
]
```

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Control Flow - @if

ng old (<17)

```
<ng-container *ngIf="store.isDone(); else notDone">
    The job is completed
</ng-container>
<ng-template #notDone>
    The job is still going
</ng-template>
```

ng new (17)

```
@if (store.isDone()) {
    The job is completed
} @else {
    The job is still going
}
```

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Control Flow - @for

ng old (<17)

```
// app.component.ts
trackById: TrackByFunction<Item> = (index, item) => item.id;


// app.component.html
    <ul>
        <li *ngFor="let item in items; let index = index; trackBy trackById">
            {{index}}. {{item.name}}
        </li>
    </ul>
```

ng new (17)

```
    <ul>
        @for (item of items; track item.id) {
            <li>{{$index}} {{item.name}}</li>
        }
    </ul>
```

ORACLE
TECH DAYS

Deferrable Views

- In Angular 17, you can lazy-load parts of the template
- This is done using the `@defer` control flow keyword
- You can then set a trigger that will cause the section to be loaded
- Possible Triggers:
    - `viewport`: Will trigger when the placeholder is in view
    - `Interaction`: Will trigger when the user interacts with the placeholder through mouse or keyboard events
    - `timer(time)`: Will trigger after a certain amount of time
    - `(when condition)`: Will trigger when the condition becomes true
    - And you can combine triggers

```
    @defer (on viewport; on timer(5s)) {
        <h1>this will be lazy loaded</h1>
    } @placeholder {
        <h1>when this is scrolled to,
            it will cause the lazy loading to
start
        </h1>
    }
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Environment Injector Initialization

- In “Module” applications, initialization is done using `APP_INITIALIZER`
  - Occurs only during application initialization
  - Does not occur when a module is lazy loaded
- In “standalone” applications, you can initialize environment injector when it is created
  - Use the `ENVIRONMENT_INITIALIZER` token for that
- You can create your own environment injectors using the `createEnvironmentInjector()` function

```
export const routes: Routes = [
    {
        path: 'courses',
        loadChildren: () => import('./courses/courses.routes')
                            .then(m => m.coursesRoutes),
        providers: [ENVIRONMENT_INITIALIZER, () =>
                    console.log('called when the courses route is loaded')]
    }
];
```

Demo – Standalone Routing

- ✓ Define Router
- ✓ Lazy load components
- ✓ Lazy load routes set
- ✓ Use Environment Injectors
- ✓ Initialize Environment Injector

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Other Goodies

- Directive Composition
- Router Inputs
- Functional Guards
- Functional Resolver

ORACLE TECH DAYS

Directive Composition

- Directives and components may have `hostDirectives`
- These are applied to the component or directive itself

```
@Directive({
  selector: '[appMy]',
  standalone: true,
  hostDirectives: [HostedDirective]
})
export class MyDirective {}
```

- You can bind inputs and outputs of the hosted directives to the inputs and outputs of the host directive

```
@Directive({
  selector: '[appMy]',
  standalone: true,
  hostDirectives: [{
      directive: HostedDirective,
      inputs: ['hostedInput: myInput'],
      outputs:['onHostedEvent: myEvent']
    }]
})
export class MyDirective {}
```

ORACLE TECH DAYS

ORACLE JOHN BRYCE תלמדו הייטק. זה עובד! a matrix company

Functional Guards and Resolvers

- Guards and Resolvers are classes that wrap a single function
- The only reason we used a class was… for Dependency Injection
- But now, with the inject() function, we can inject into functions.
- Use the `ResolveFn<T>` and `CanActivateFn` Typescript types to help you create such functions

```
// app.routes.ts
export const routes: Routes = [
    {
        path: 'admin',
        canActivate: () => inject(AuthService).isAdmin()
    }
];
```

```
// app.routes.ts
export const routes: Routes = [
    {
        path: 'admin',
        resolve: {user: () => inject(DateService).getUser()}
    }
];
```

ORACLE TECH DAYS

Router Inputs

- Components may receive inputs from the router
  - Router parameters
  - Query parameters
  - Route data from the data property
  - Route data from the resolvers
- You can respond to value changes by using:
  - Property Setters
  - OnChanges hook

```
// app.routes.ts
export const routes: Routes = [
    {
        path: 'course/:id',
        data: { courseType: 'online' },
        resolve: {topic: () => inject(DateService).getTopic()},
        component: CourseComponent
    }
];

// course.component.ts
export class CourseComponent {
    @Input() id?: string;
    @Input() courseType?: string;
    @Input() topic?: string;
}
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Demo – Standalone Routing

- ✓ Define functional guard
- ✓ Define functional resolver
- ✓ Feed component inputs
- ✓ Use Directive Composition

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Signals

The new reactivity

ORACLE TECH DAYS

ORACLE JOHN BRYCE תלמדו הייטק. זה עובד! a matrix company

Signals

- Normal properties do not maintain their relationships with each other

```
let width = 10;
let height = 20;
let area = width * height; // 200

width = 15;
// area is still 200
```

- With signals, atomic values “signal” when they change, so computed values remain correct

```
let width = signal(10);
let height = signal(20);
let area = computed(() => width() * height()); // 200

width.set(15);
console.log(area()); // area is 300
```

- Yes… It’s kinda like `BehaviorSubject`

The signal primitives

- You create a signal with the `signal()` function
- You create a computed signal with the computed() function
  - Do not cause side effects inside it
  - Do not create new signals inside it
  - This should be a pure function that depends on other signals – that's it
- You can read from a signal by simply calling it like a function
  - It is synchronous
  - It will return a value instantly
  - No need to subscribe – key difference from observables

```
let firstName = signal('Kobi');
let lastName = signal('Hari');


let fullName = computed(() =>
     firstName() + ' ' + lastName());

console.log(fullName());
```

ORACLE TECH DAYS

ORACLE

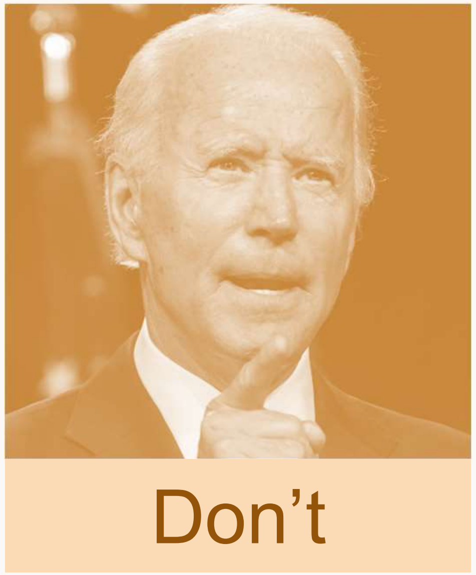
JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Inside computed, can you: ?

```
 // Create a new signal?
  let fullName = computed(() =>
        signal('Mr')
            + firstName() + ' '
            + lastName());

// Cause side effect?
  let fullName = computed(() =>
        console.log('HEY!!!'));

// do asynchronous stuff
  let fullName = computed(() =>
        await userService.getName());
```



ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Updating signals

- You can only update atomic (writeable) signals
- Use set() when you want to set a new value that does not depend on the previous
- Use update() when you want to set a new value that **does** depend on previous value
- Use mutate() when you want to modify the original value (mutate array or object inside signal).
    - Don't...

Do

```
firstName.set('Yakov');
lastName.update(val => val.toLowerCase());
```

Don't

```
firstName.set('Mr ' + firstName());
```

Instead, do

```
let titledName = computed(() => 'Mr ' + firstName());

// or

firstName.update(val => 'Mr ' + val);
```

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

When updating signals, should you: ?

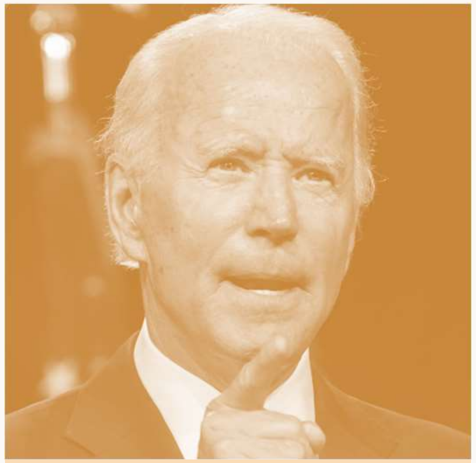
```
 // use the previous value the signal?
 firstname.set('Mr' + firstName());

 // instead use update(state => newState)


// use the value of another signal?
firstname.set('Mr' + favoriteName());

// instead use computed


// in general, use mutate?
myArray.mutate(val => val.push('new value'))
```



Don't

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

effect

- Use effect to respond to changes in signals
  - The effect will run once when defined
  - It will run again every time one the signals it depends on change

- You may not change other signals inside effect
  - It will throw an error if you try…
  - But you Can modify signals inside effect after `await`

```
effect(() => {
    console.log('full name: ', fullName());
});
```

Don’t

```
let salutation = signal('');
effect(() => {
    salutation.set('Hello' + fullName());
});
```

Instead, do

```
let salutation = computed(() => 'Hello' + fullName());
```

This is, actually, allowed

```
effect(async () => {
    if (fullName() === 'Kobi') {
        let res = await (someService.someMethod());
        resultSignal.set(res);
    }
});
```

ORACLE
TECH DAYS

Limitations

1. Signals are synchronous – so they must have initial value when created

2. signal(), computed() and effect() must be called in Injection Context
   - Because they rely on `DestroyRef` to complete and unsubscribe

3. You can not create signals inside `effects` or `computed` (See rule number 2)

Is RxJs dead?

- Signals are great for
  - Component binding
  - Alternative to `map` and `combineLatest`
  - They are the future of angular
- But…
  - You cannot replace higher order observables
  - You cannot do anything asynchronous with them
  - You cannot filter them…
  - They rely on injection context so you cannot use them anywhere else
  - ~~They are still for review…~~
  - ~~Angular 17 will have further improvements~~

ORACLE TECH DAYS

ORACLE JOHN BRYCE תלמדו הייטק. זה עובד! a matrix company

Signals - Verdict

- On its own – problematic.
- Too much Magic
  - Relies too heavily on `InjectionContext`
  - Has a complex algorithm for change flow, that may cause bugs
  - Effects can't really call any service, create new signals, or even change them.
- You should currently only use it in components
  - And only if they are totally synchronous by nature
  - Otherwise – requires complementary technology
- Recommended: Use it with NgRx Signal Store

ORACLE TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Demo – Signals

- ✓ Create Writeable signals
- ✓ Create computed signals
- ✓ Present Signals in components
- ✓ Modify Signals
- ✓ Create side effects
- ✓ See the limitations of signals

ORACLE
TECH DAYS

ORACLE

JOHN BRYCE
תלמדו הייטק. זה עובד!
a matrix company

Thank You