Blaise 5
Custom Blaise Web/MVC apps

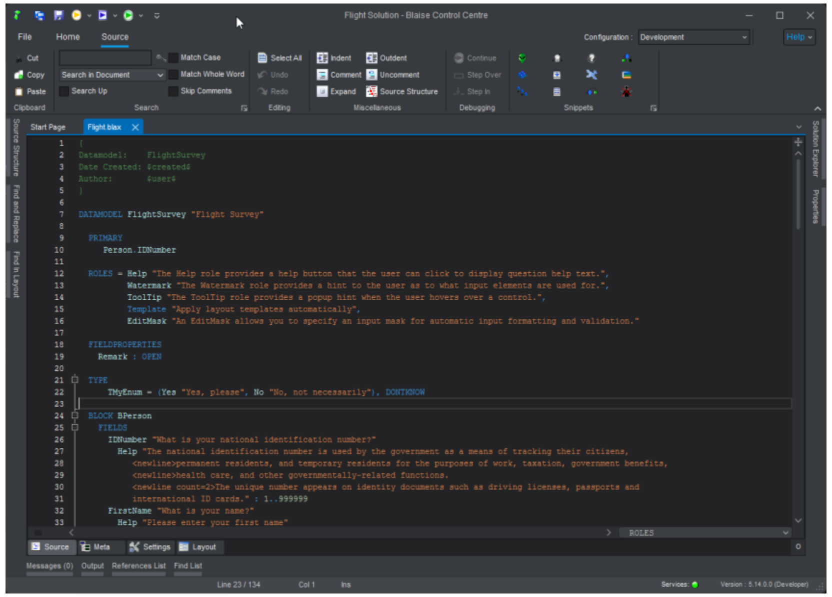
Based on Blaise 5.14, works the same on 5.13

Bas Huijts
shj.huijts@cbs.nl

Agenda

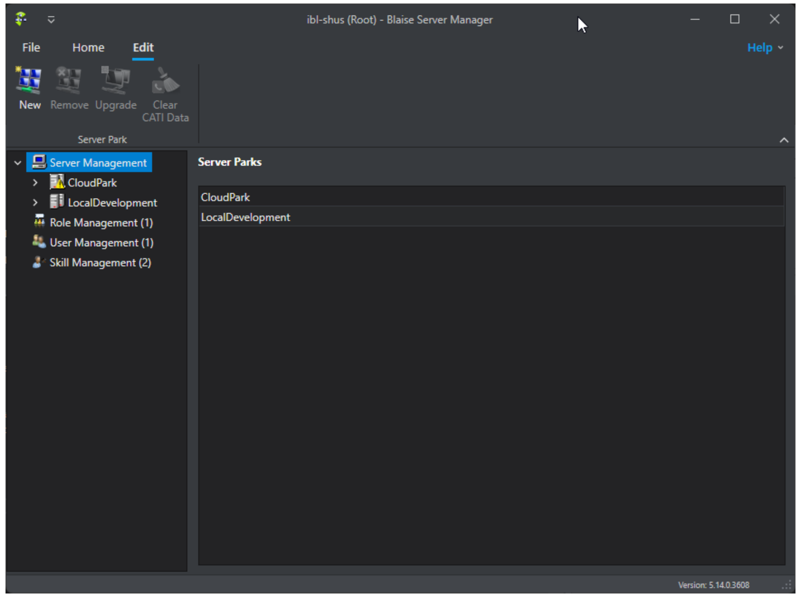
- Why do we need customization
- Angular basics
- Customization
- Q&A

First things first [1]

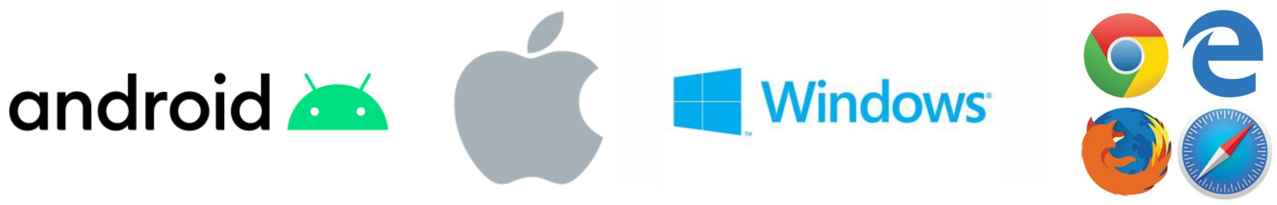


```
{
Datamodel:    FlightSurvey
Date Created: $created$
Author:       $user$
}

DATAMODEL FlightSurvey "Flight Survey"

  PRIMARY
     Person.IDNumber

  ROLES = Help "The Help role provides a help button that the user can click to display question help text.",
          Watermark "The Watermark role provides a hint to the user as to what input elements are used for.",
          ToolTip "The ToolTip role provides a popup hint when the user hovers over a control.",
          Template "Apply layout templates automatically",
          EditMask "An EditMask allows you to specify an input mask for automatic input formatting and validation."

  FIELDPROPERTIES
    Remark : OPEN

  TYPE
      TMyEnum = (Yes "Yes, please", No "No, not necessarily"), DONTKNOW

  BLOCK BPerson
    FIELDS
      IDNumber "What is your national identification number?"
        Help "The national identification number is used by the government as a means of tracking their citizens,
           <newline>permanent residents, and temporary residents for the purposes of work, taxation, government benefits,
           <newline>health care, and other governmentally-related functions.
           <newline count=2>The unique number appears on identity documents such as driving licenses, passports and
           international ID cards." : 1..999999
      FirstName "What is your name?"
        Help "Please enter your first name"
```

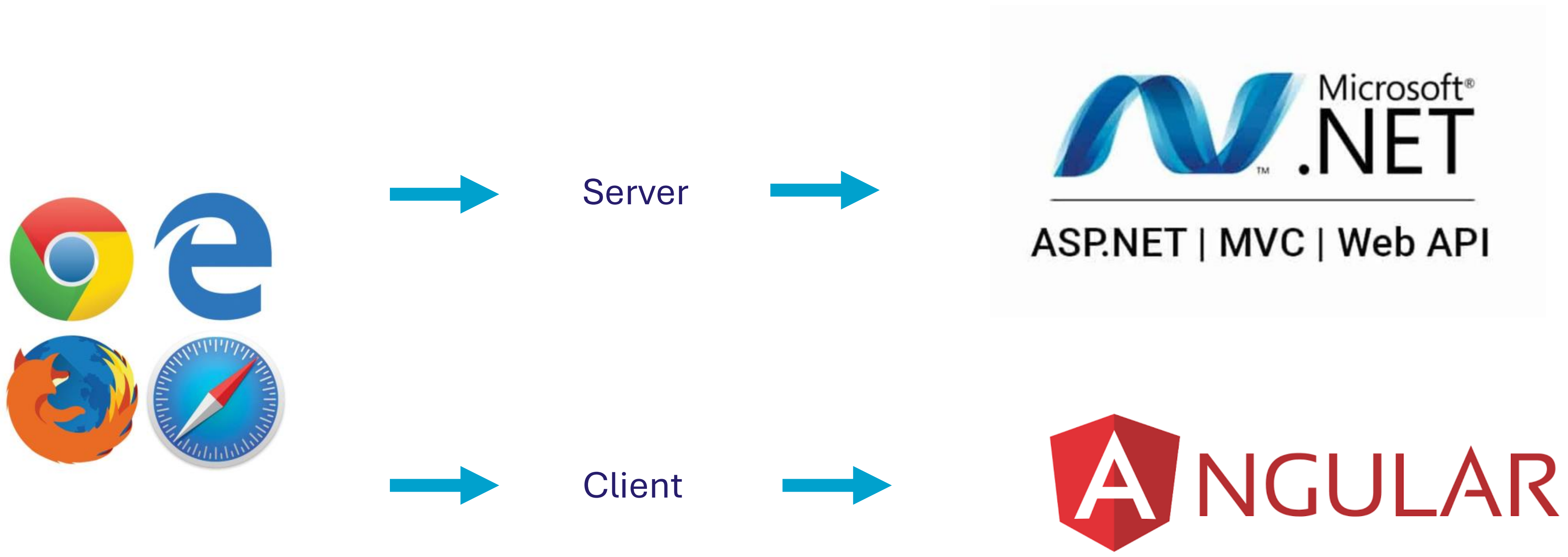
Blaise Control Centre



Blaise Server Manager

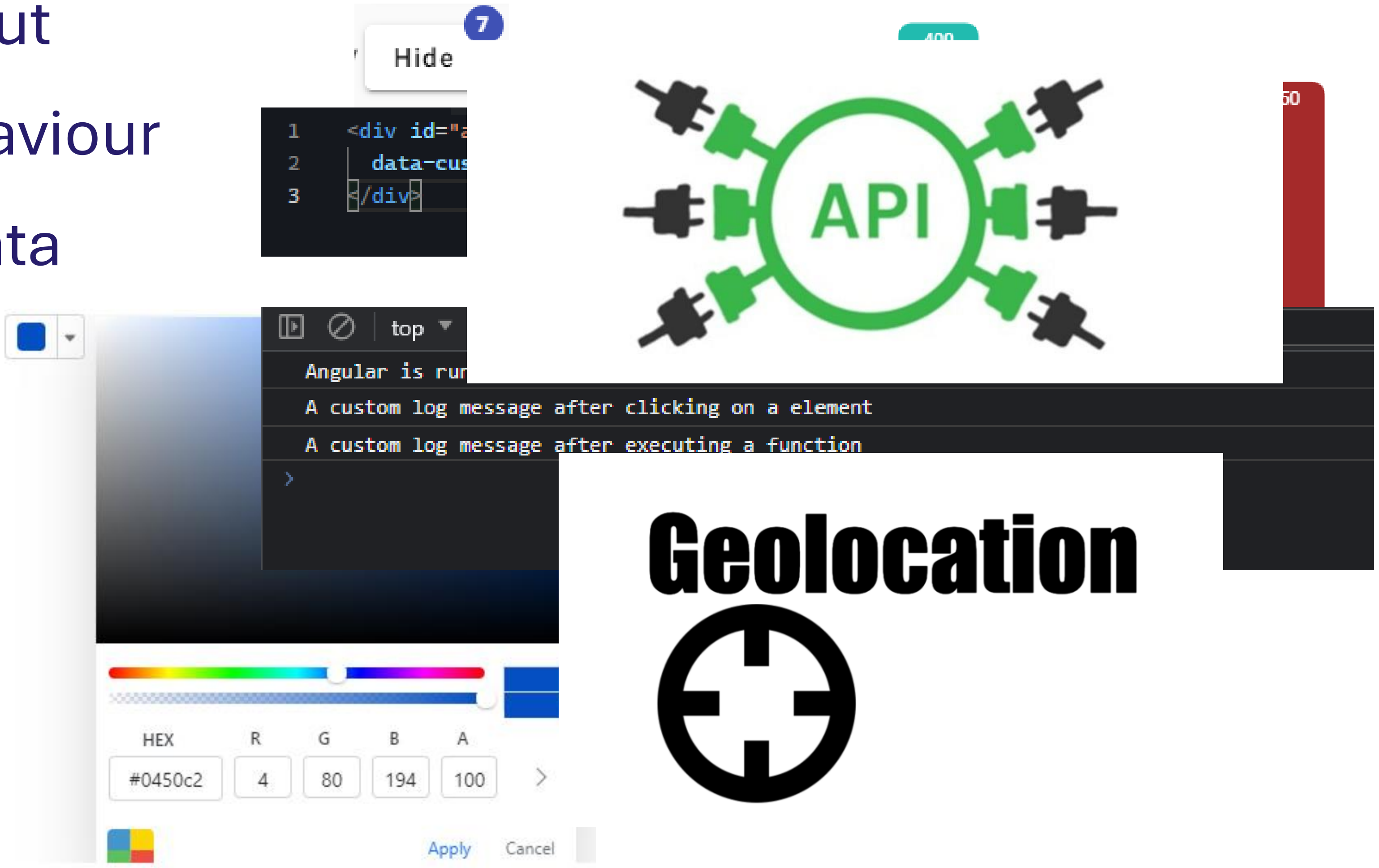


First things first [2]



Why customizing

- Need for custom layout
- Need for custom behaviour
- Accessing external data



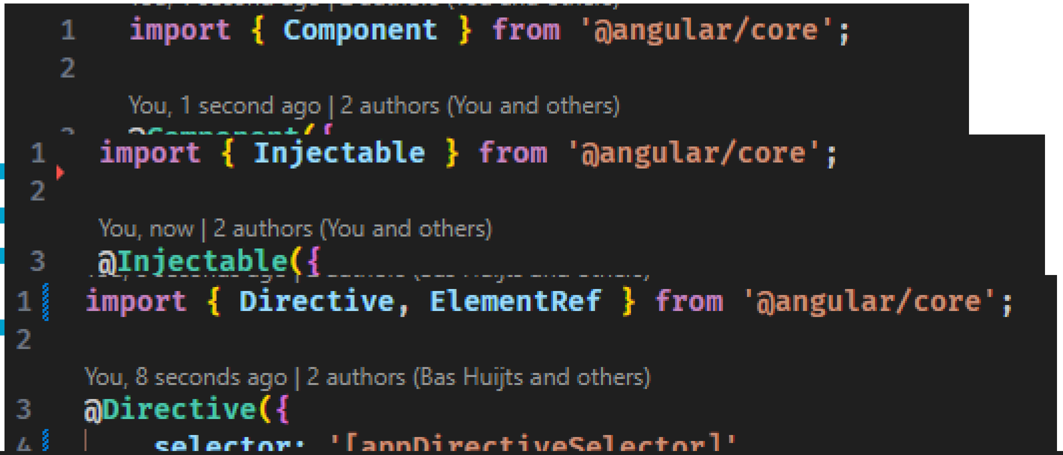
```
<div id="a
  data-cus
</div>
```

```
Angular is rur
A custom log message after clicking on a element
A custom log message after executing a function
```

HEX #0450c2, R 4, G 80, B 194, A 100

Angular basics

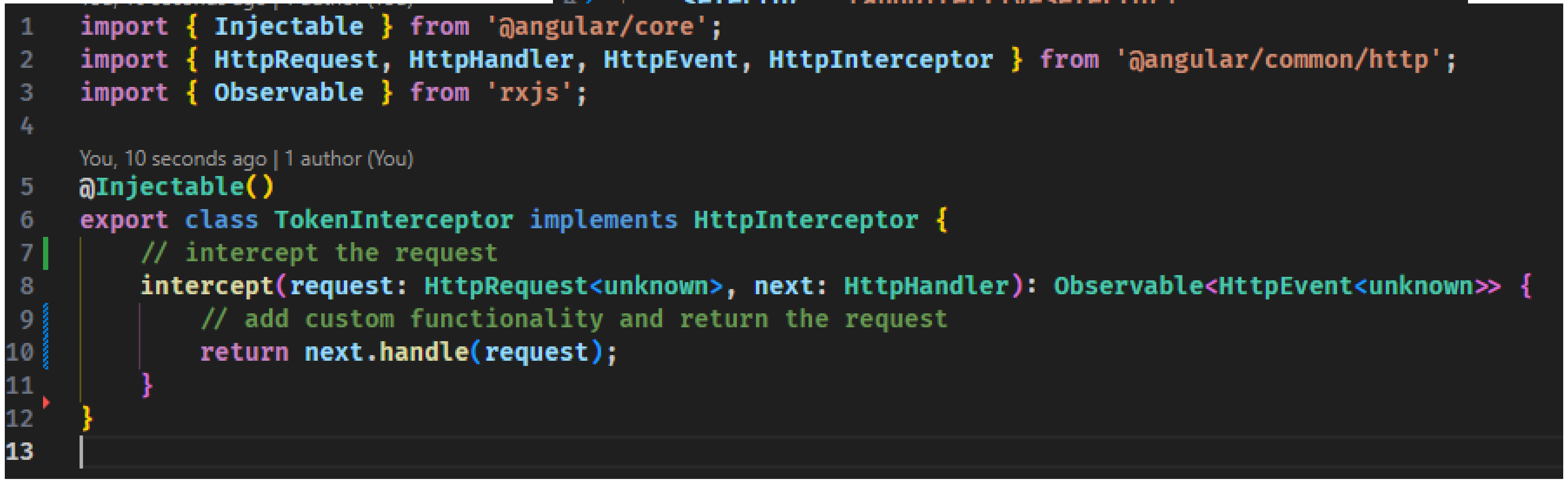
- Component:
  - HTML Template
  - Typescript class
  - Css Selector
  - Optionally css styles
- Service
- Directive
- Interceptor



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

You, 1 second ago | 2 authors (You and others)
@Component({
```

```
import { Injectable } from '@angular/core';

You, now | 2 authors (You and others)
@Injectable({
```

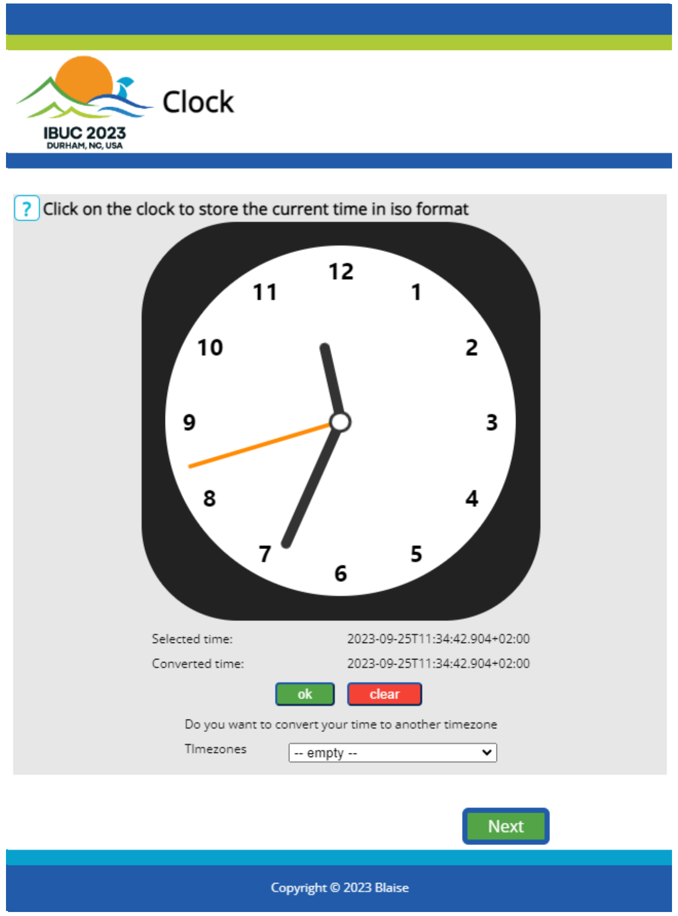
```
import { Directive, ElementRef } from '@angular/core';

You, 8 seconds ago | 2 authors (Bas Huijts and others)
@Directive({
  selector: '[appDirectiveSelector]'
```



```
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from '@angular/common/http';
import { Observable } from 'rxjs';

You, 10 seconds ago | 1 author (You)
@Injectable()
export class TokenInterceptor implements HttpInterceptor {
    // intercept the request
    intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {
        // add custom functionality and return the request
        return next.handle(request);
    }
}
```

Component

- Different layout / view for a certain input control, i.e. clock-view instead of stringtextbox
- Add missing functionality to a control, in my demo converting timezones using an external web api



Click on the clock to store the current time in iso format

Selected time: 2023-09-25T11:34:42.904+02:00

Converted time: 2023-09-25T11:34:42.904+02:00

Do you want to convert your time to another timezone

Timezones: -- empty --

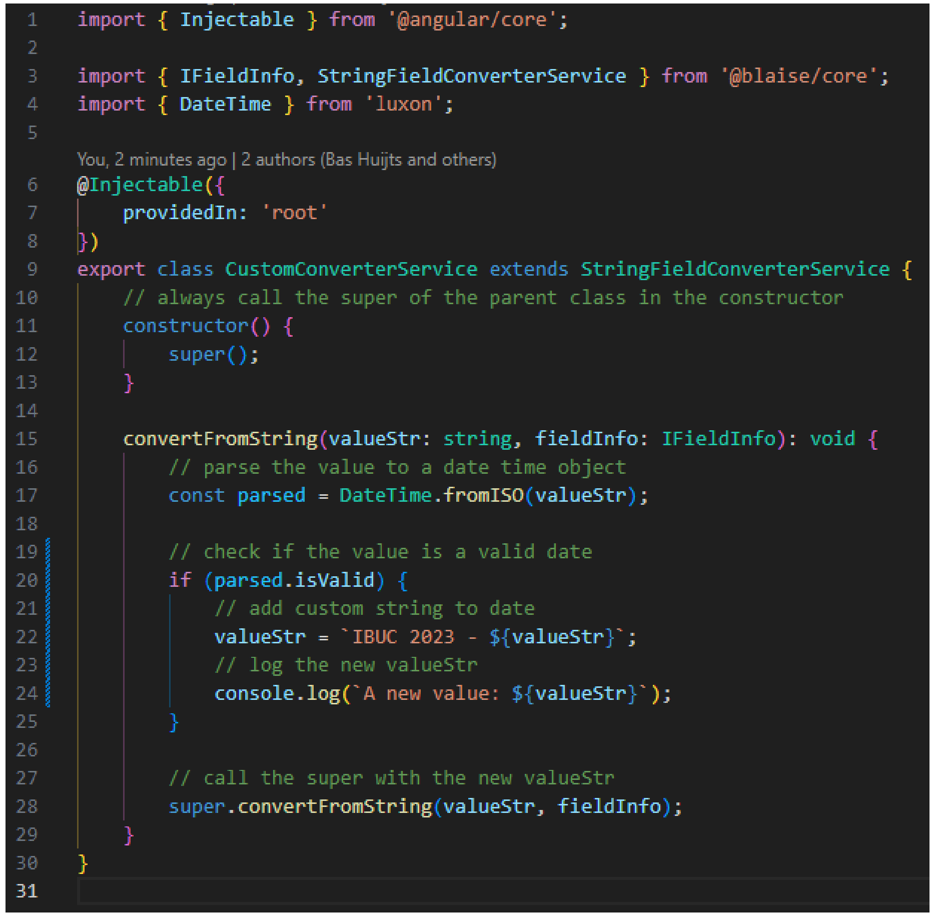
Copyright © 2023 Blaise

Demo Component customization

- Recap
  - Created an empty custom application
  - Added a custom property to make Angular aware of the customization
  - Created the custom clock component -> Blaise StringTextBox
  - Create a non Blaise component
  - Added a non Blaise Angular service

Service

- Client side rules
- Field Converters



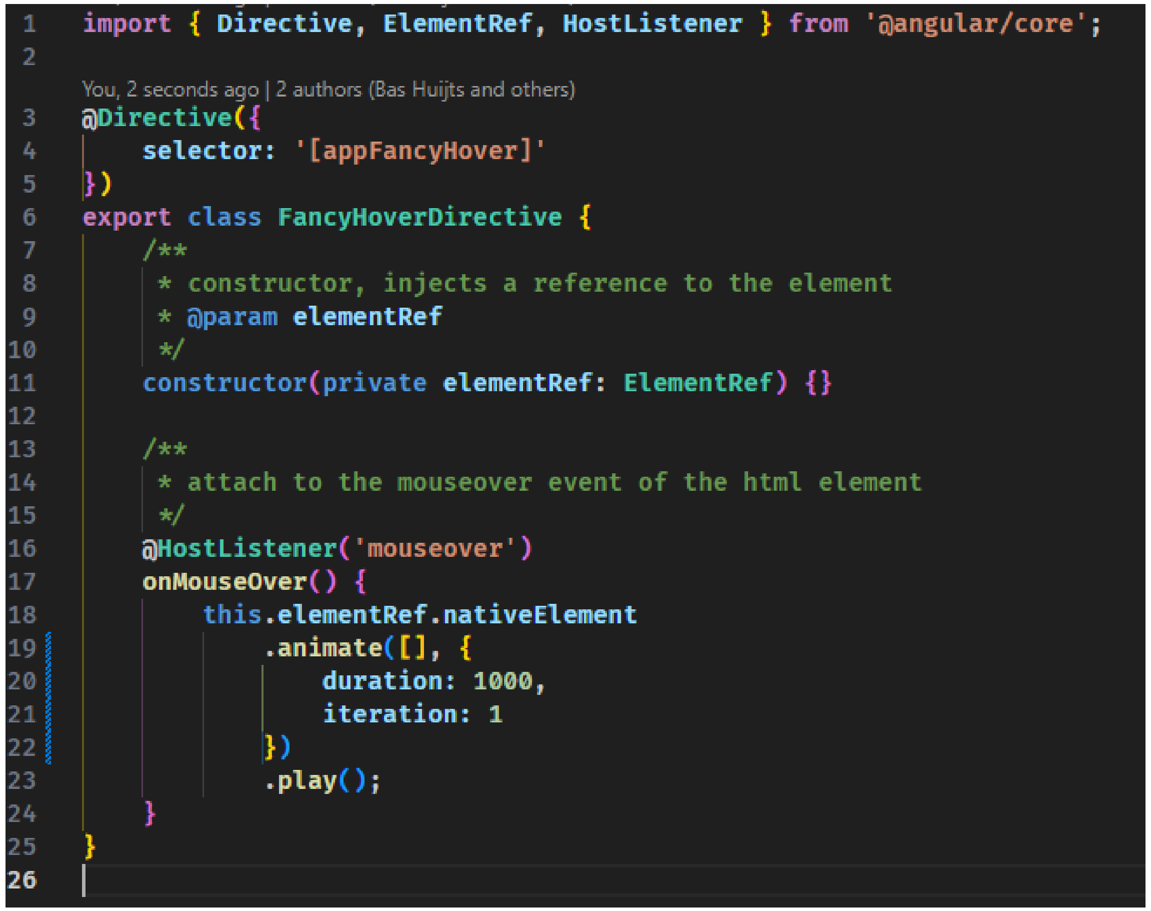
```
import { Injectable } from '@angular/core';

import { IFieldInfo, StringFieldConverterService } from '@blaise/core';
import { DateTime } from 'luxon';

You, 2 minutes ago | 2 authors (Bas Huijts and others)
@Injectable({
    providedIn: 'root'
})
export class CustomConverterService extends StringFieldConverterService {
    // always call the super of the parent class in the constructor
    constructor() {
        super();
    }

    convertFromString(valueStr: string, fieldInfo: IFieldInfo): void {
        // parse the value to a date time object
        const parsed = DateTime.fromISO(valueStr);

        // check if the value is a valid date
        if (parsed.isValid) {
            // add custom string to date
            valueStr = `IBUC 2023 - ${valueStr}`;
            // log the new valueStr
            console.log(`A new value: ${valueStr}`);
        }

        // call the super with the new valueStr
        super.convertFromString(valueStr, fieldInfo);
    }
}
```

Demo Service customization (Blaise)

- Recap
  - Added a custom converter based on the default StringFieldConverterService
  - Added a prefix to the value when needed
  - Added some logging

Directive

- Add fancy animations to controls



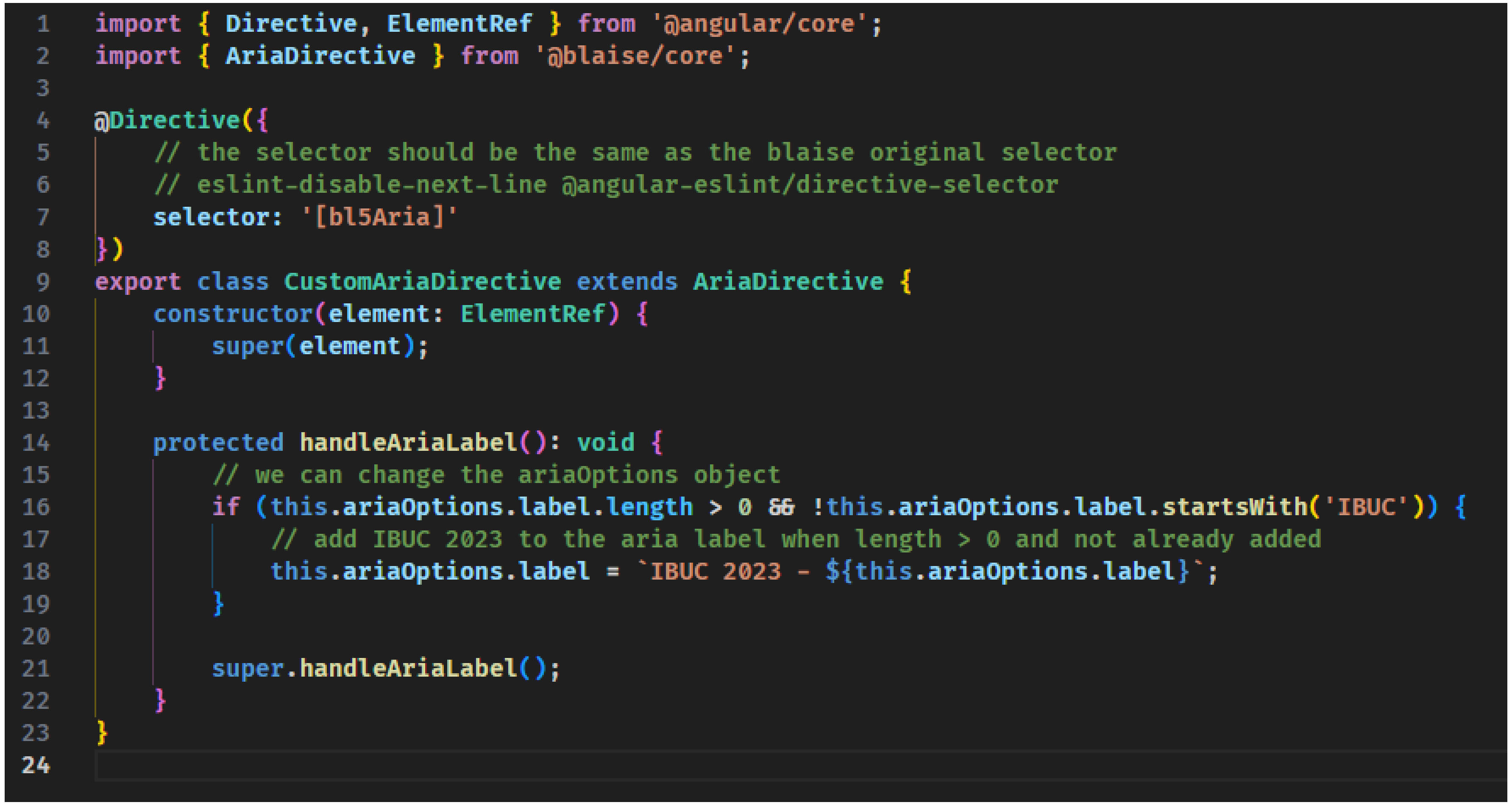
```
import { Directive, ElementRef, HostListener } from '@angular/core';

You, 2 seconds ago | 2 authors (Bas Huijts and others)
@Directive({
    selector: '[appFancyHover]'
})
export class FancyHoverDirective {
    /**
     * constructor, injects a reference to the element
     * @param elementRef
     */
    constructor(private elementRef: ElementRef) {}

    /**
     * attach to the mouseover event of the html element
     */
    @HostListener('mouseover')
    onMouseOver() {
        this.elementRef.nativeElement
            .animate([], {
                duration: 1000,
                iteration: 1
            })
            .play();
    }
}
```

Demo Directive customization (non Blaise)

- Recap
  - Create a non Blaise directive
  - Made Angular aware of the directive
  - Applied it to elements

Directive

- Override Blaise AriaDirective



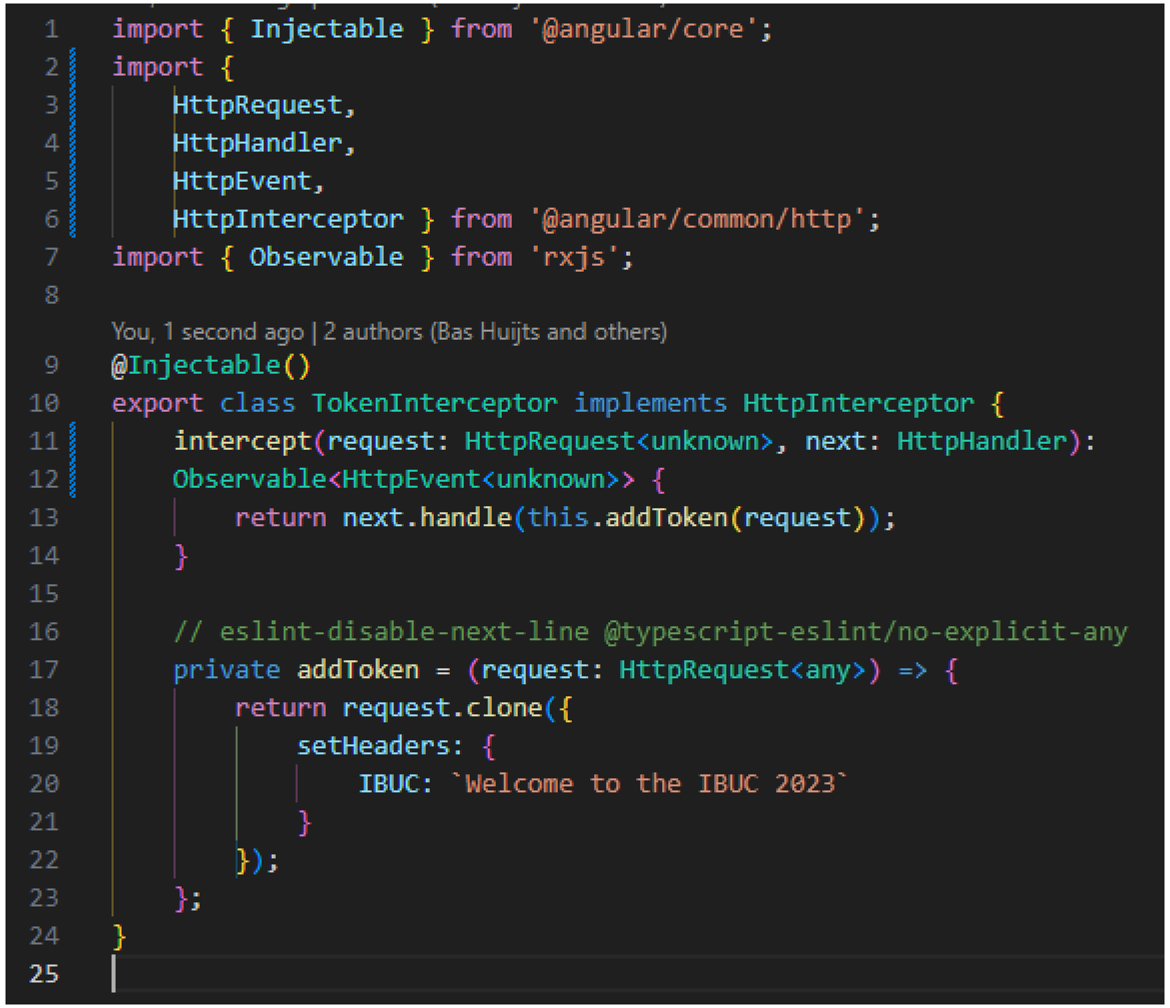
```
import { Directive, ElementRef } from '@angular/core';
import { AriaDirective } from '@blaise/core';

@Directive({
    // the selector should be the same as the blaise original selector
    // eslint-disable-next-line @angular-eslint/directive-selector
    selector: '[bl5Aria]'
})
export class CustomAriaDirective extends AriaDirective {
    constructor(element: ElementRef) {
        super(element);
    }

    protected handleAriaLabel(): void {
        // we can change the ariaOptions object
        if (this.ariaOptions.label.length > 0 && !this.ariaOptions.label.startsWith('IBUC')) {
            // add IBUC 2023 to the aria label when length > 0 and not already added
            this.ariaOptions.label = `IBUC 2023 - ${this.ariaOptions.label}`;
        }

        super.handleAriaLabel();
    }
}

```

Demo Directive customization (Blaise)

- Recap
  - Created an custom Blaise AriaDirective
  - Make Angular aware of the directive
  - Apply it to elements

Interceptor

- Front-end <-> back-end
- Authenticate
- Authorize
- Logging
- Add custom header

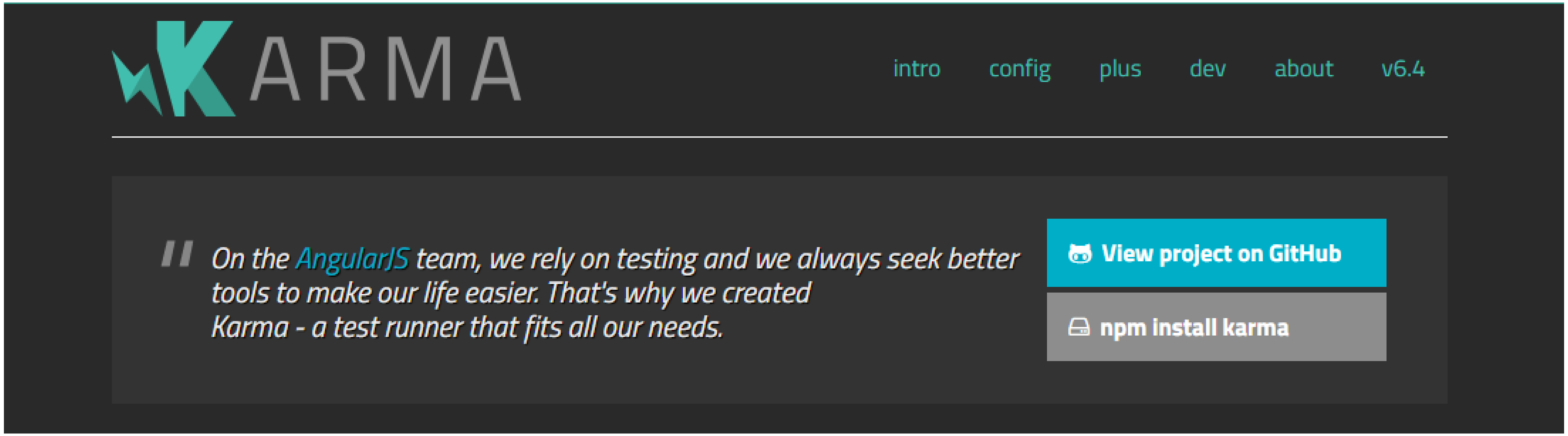


```
import { Injectable } from '@angular/core';
import {
    HttpRequest,
    HttpHandler,
    HttpEvent,
    HttpInterceptor } from '@angular/common/http';
import { Observable } from 'rxjs';

You, 1 second ago | 2 authors (Bas Huijts and others)
@Injectable()
export class TokenInterceptor implements HttpInterceptor {
    intercept(request: HttpRequest<unknown>, next: HttpHandler):
    Observable<HttpEvent<unknown>> {
        return next.handle(this.addToken(request));
    }

    // eslint-disable-next-line @typescript-eslint/no-explicit-any
    private addToken = (request: HttpRequest<any>) => {
        return request.clone({
            setHeaders: {
                IBUC: `Welcome to the IBUC 2023`
            }
        });
    };
}
```

Demo Interceptor customization (non Blaise)

- Recap
  - Created an interceptor to add a custom header
  - Make Angular aware of it

Testing



On the AngularJS team, we rely on testing and we always seek better tools to make our life easier. That's why we created Karma - a test runner that fits all our needs.

- Karma as testing framework
  - Unit tests
  - Code coverage

Demo Unit testing with Karma

- Recap
  - Add Karma to our application
  - Added unit tests to CustomAriaDirective
  - Created code coverage report

Recap

- Changed a component to use another view
- Used an standalone service to add functionality
- Overrided an Blaise service to change the entered value
- Added an directive to apply an hover animation
- Overrided an Blaise directive to prefix an label
- Implemented an interceptor to add an token to reqeusts
- Added unit testing to our application
- Generated code coverage of our custom aria directive

Q&A

Thank you for attention