

One Source to Rule Them All





We Want It All

We Want It All

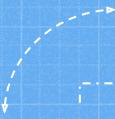
It's not uncommon that we - and the decision makers - want it all.

In the world of applications we want both a web presence and a native mobile presence.

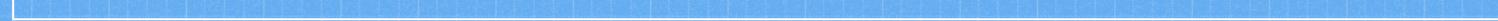
What we don't want is to maintain competing parallel codebases.

We Want It All

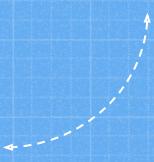
Tug-of-war between wanting to maintain only one source code versus our app having a home everywhere our users are found.

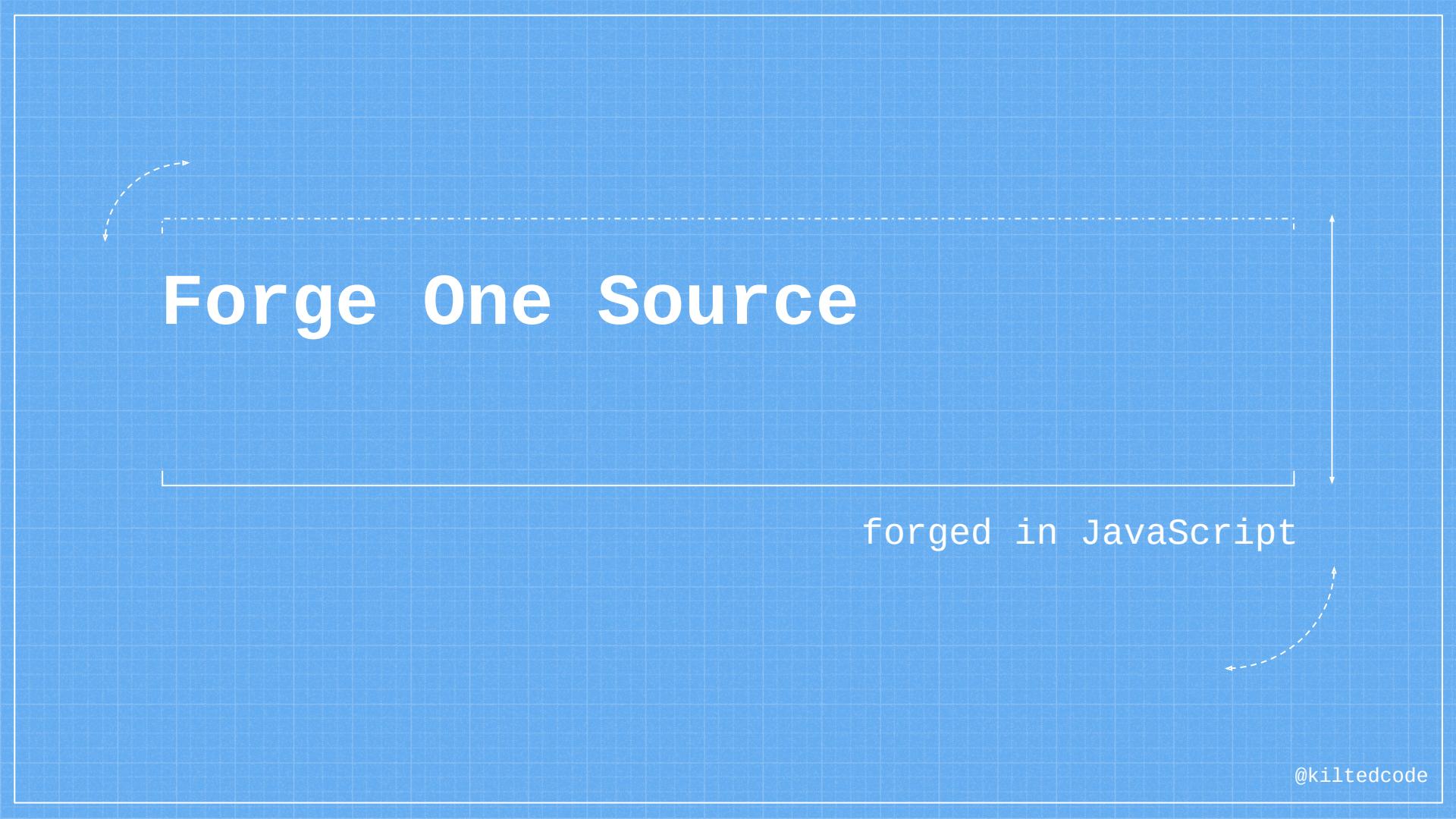


Who am I?



a kilt wearing coder





Forge One Source

forged in JavaScript

Forge One Source

- ★ What do we want?
- ★ What is NativeScript?
- ★ How do we share code?
- ★ xplat



What do we want?

Problem Solving

- ★ We *want* both a web and native application.
- ★ We *want* code reuse.
- ★ We *don't want* multiple separate projects.



What is NativeScript?

What is NativeScript?

Open source framework designed to empower
JavaScript with native APIs.

What is NativeScript?

Allows you to choose the flavor you love:

- ★ JavaScript
- ★ TypeScript
- ★ Angular
- ★ Ionic
- ★ Vue
- ★ React
- ★ Svelte

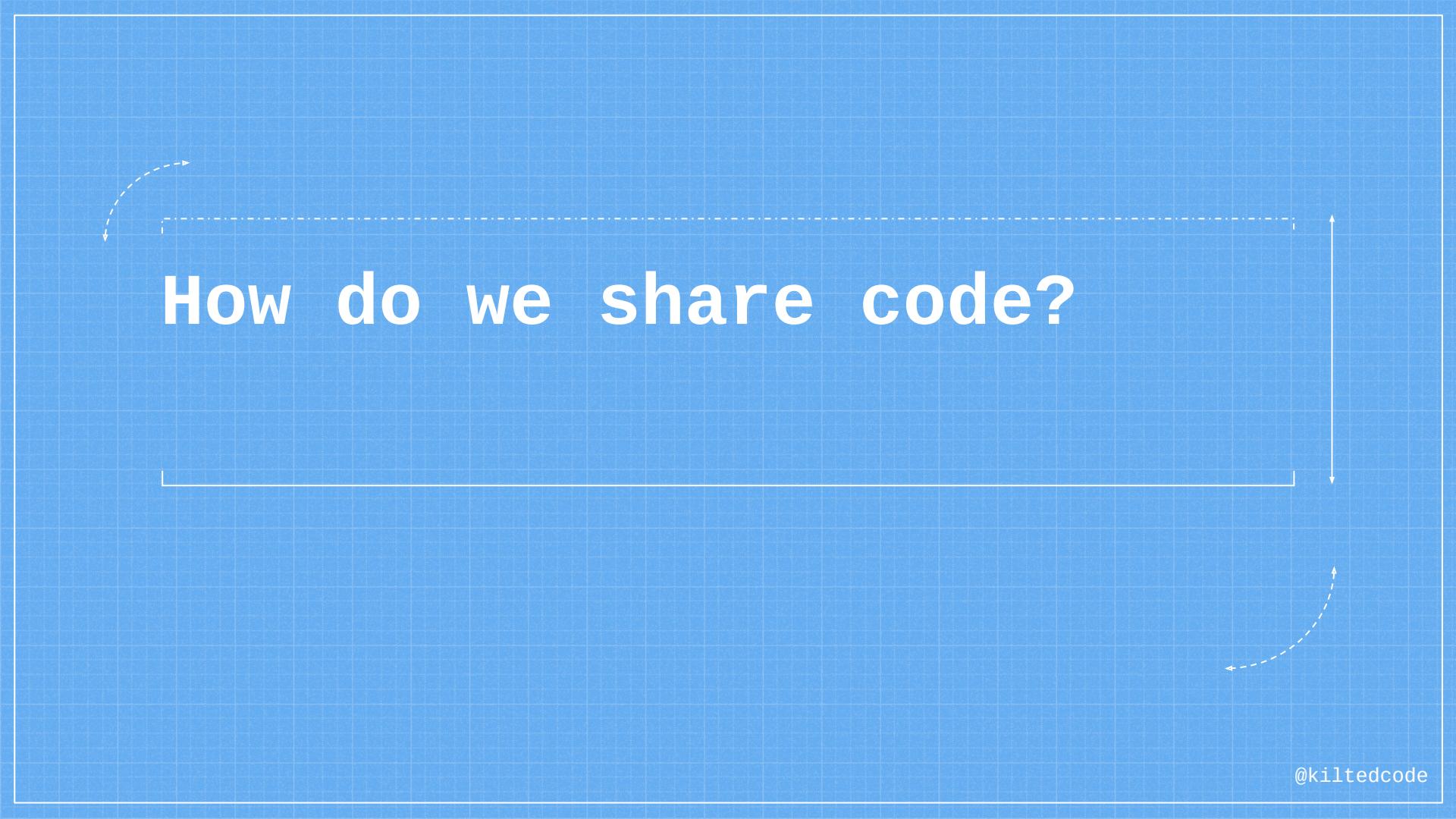
nativescript.org

NativeScript

- ★ Native performance
- ★ Extensible
- ★ Develop with what you know
- ★ Cross platform
- ★ Free and open source
- ★ Strong backing

NativeScript Layers

- ★ Write our app logic in Angular
- ★ UI with HTML-like syntax
- ★ Style with CSS



How do we share code?

How do we share code?

In Angular, a component has a TS class, an HTML template, and a style sheet.

How do we share code?

In NativeScript with Angular, a component has a TS class, an HTML-like template, and a style sheet.

How do we share code?

If all TypeScript code were the same, you can see how easily could just have two HTML and two style sheets.



Fundamental lessons about good code sharing

From NativeScript TSC

1

If you can share your code easily with other paradigms, other disciplines, other runtimes even, then you have a good code sharing approach that will continue to provide you and your team joy. No developer or team willingly gets into code sharing hoping to find themselves in a corner later. You always want to share now to more easily maintain and scale the code later.

2

A good code sharing approach should fit naturally in with well supported community standards and not require any extra build tooling just for the sharability to support itself. It should, in other words, stand firmly on it's own in scope of the language it is written in.

3

A good code sharing approach should not have to introduce new file extensions purely for sake of sharability (outside of those naturally supported by the framework) to deal with. All team's organize code by folders naturally and the same should be matched with good code sharing approaches avoiding new file extensions and concepts that go beyond general code organization.

4

A good code sharing approach should clearly identify deployment/distribution lines as well as distinct platform separation allowing various shared code segments to have clear designated deployment targets allowing teams to control their own sophisticated build pipelines as they desire. Further the shared code should live within a thoughtful organizational structure that supports the ability to scale and adapt to future needs aside from the deployment targets that use the shared code.

5

Within the specific scope of NativeScript's viewpoint, JavaScript is the universal language which provides the opportunity to share code effectively and responsibly. An approach that is based fundamentally on the strengths of JavaScript (and inherently TypeScript) is a good code sharing approach.



Recommendations and Solutions

from nativescript.org

Nrwl Nx DevTools with @nativescript/nx

- ★ Centered around JS/TS (lesson 1 and 5)
- ★ Uses standard build tooling (lesson 2)
- ★ No custom file extensions (lesson 3)
- ★ Nx splits “apps” and “libs” to identify targets that consume shared code (lesson 4)

Nrwl Nx DevTools with @nstudio/xplat

- ★ Centered around JS/TS (lesson 1 and 5)
- ★ Uses standard build tooling (lesson 2)
- ★ No custom file extensions (lesson 3)
- ★ Nx splits “apps” and “libs” to identify targets that consume shared code (lesson 4)
- ★ Builds on @nativescript/nx to scaled across more paradigms
- ★ Opinionated architecture is provided by xplat to help avoid common pitfalls from learned with cross platform development

Yarn workspaces

- ★ Centered around JS/TS (lesson 1 and 5)
- ★ Uses standard build tooling (lesson 2)
- ★ No custom file extensions (lesson 3)
- ★ Can link / share dependencies

Lerna

- ★ Centered around JS/TS (lesson 1 and 5)
- ★ Uses standard build tooling (lesson 2)
- ★ No custom file extensions (lesson 3)

xplat

Cross-platform (xplat)
tools for Nx workspaces

xplat

Added value pack for Nx to provide app generators and optional supporting architecture.

Supported platforms:

- ★ NativeScript
- ★ Web
- ★ Ionic
- ★ Electron

Up and running

Setup NativeScript
docs.nativescript.org

- ★ Node
- ★ NativeScript CLI
- ★ App environment
 - Android Studio + JDK
 - XCode, xcdeproj, cocoapods

Up and Running

Start with a Nx workspace

```
$ npx create-nx-workspace
```

Up and Running

CLIs and xplat

```
$ npm install -g @nrwl/cli  
$ npm install --save-dev @nstudio/xplat  
$ ng add @nstudio/xplat  
$ nx generate app
```

Robust Structure

Generate app

- ★ Name app
- ★ Select type of app to add
- ★ Generates the code

```
$ nx generate app
✓ What name would you like for this app? · hobbit-fitness-pal
? What type of app would like to create? ...
> electron      [Electron app]
  express        [Express app]
  ionic          [Ionic app]
  nativescript   [NativeScript app]
  nest           [Nest app]
  node           [Node app]
  react          [React app]
  web            [Web app]
```

Benefits of xplat

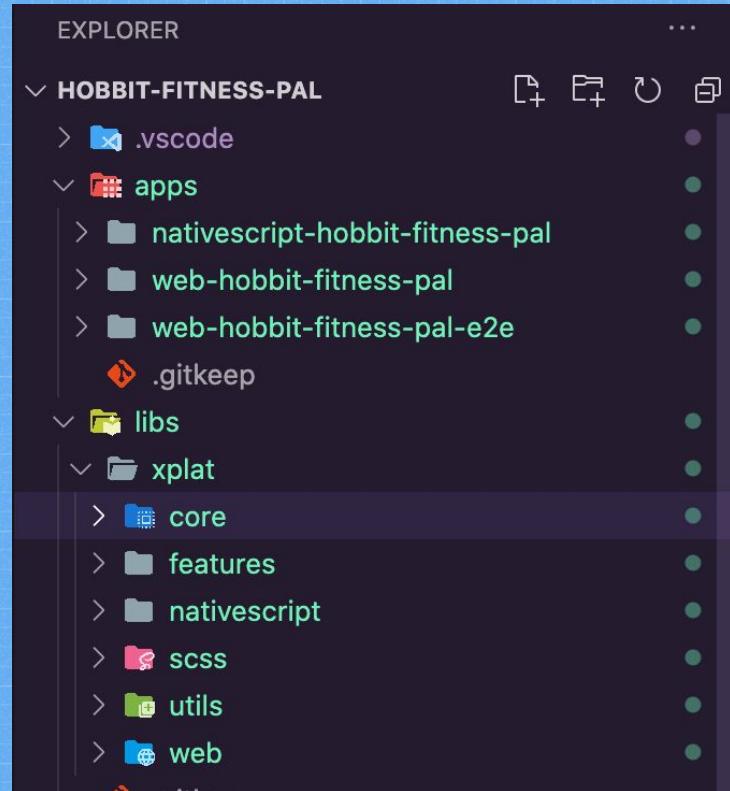
- ★ Robust structure for code sharing.
- ★ Easier to take advantage of deeper platform integration.
- ★ Built to support large complex codebases with sharing.

Robust Structure

Add multiple apps

End up with powerful code sharing structure

- ★ Individualized apps by type
- ★ Libs xplat for shared code



Generating Code with xplat

```
$ nx generate @nstudio/angular:service --help
nx generate @nstudio/angular:service [name] [options,...]
Options:
  --name          Service name
  --feature       Target feature
  --projects      Target projects
  --platforms     Target platforms
  --skipFormat    Skip formatting files
  --dryRun         Runs through and reports activity without writing to disk.
  --skip-nx-cache Skip the use of Nx cache.
  --help          Show available options for project target.
```

Generating Code with xplat

```
$ nx generate @nstudio/angular:service track --dryRun  
CREATE libs/xplat/core/src/lib/services/track.service.ts  
UPDATE libs/xplat/core/src/lib/services/index.ts
```

Track Overview Service

```
import { Injectable } from '@angular/core';
import { Observable, of } from 'rxjs';
import { TrackOption } from '../models';

@Injectable({
  providedIn: 'root',
})
export class TrackOverviewService {
  private tracking: TrackOption[] = [
    // mock data
  ];

  constructor() {}

  getTrackingOverview(): Observable<TrackOption[]> {
    return of(this.tracking);
  }
}
```

Generating Code with xplat

```
$ nx generate @nstudio/angular:component meal-list --platforms=web,nativescript  
--createBase=true --dryRun  
CREATE libs/xplat/features/src/lib/ui/base/meal-list.base-component.ts  
CREATE libs/xplat/web/features/src/lib/ui/components/meal-list/meal-list.component.html  
CREATE libs/xplat/web/features/src/lib/ui/components/meal-list/meal-list.component.ts  
CREATE  
libs/xplat/nativescript/features/src/lib/ui/components/meal-list/meal-list.component.html  
CREATE  
libs/xplat/nativescript/features/src/lib/ui/components/meal-list/meal-list.component.ts  
UPDATE libs/xplat/features/src/lib/ui/base/index.ts  
UPDATE libs/xplat/web/features/src/lib/ui/components/index.ts  
UPDATE libs/xplat/nativescript/features/src/lib/ui/components/index.ts
```

Generating Code with xplat

```
$ nx generate @nstudio/angular:component food-item  
--projects=web-hobbit-fitness-pal,nativescript-hobbit-fitness-pal --dryRun  
CREATE  
apps/web-hobbit-fitness-pal/src/app/features/shared/components/food-item/food-item.component.html  
CREATE  
apps/web-hobbit-fitness-pal/src/app/features/shared/components/food-item/food-item.component.ts  
CREATE apps/web-hobbit-fitness-pal/src/app/features/shared/components/index.ts  
CREATE  
apps/nativescript-hobbit-fitness-pal/src/features/shared/components/food-item/food-item.component.html  
CREATE  
apps/nativescript-hobbit-fitness-pal/src/features/shared/components/food-item/food-item.component.ts  
CREATE apps/nativescript-hobbit-fitness-pal/src/features/shared/components/index.ts
```

Home Base Component

```
import { Directive, OnInit } from '@angular/core';
import { BaseComponent, TrackOption, TrackOverviewService } from '@hobbit-fitness-pal/xplat/core';
import { Observable } from 'rxjs';

@Directive()
export abstract class HomeBaseComponent extends BaseComponent implements OnInit {
  trackingOverview$: Observable<TrackOption[]>;

  constructor(
    private trackOverviewService: TrackOverviewService
  ) {
    super();
  }

  ngOnInit() {
    this.getOverview();
  }

  getOverview(): void {
    this.trackingOverview$ = this.trackOverviewService.getTrackingOverview();
  }
}
```

Home Web Component

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

import { HomeBaseComponent } from '@hobbit-fitness-pal/xplat/features';

@Component({
  selector: 'hobbit-fitness-pal-home',
  templateUrl: 'home.component.html',
})
export class HomeComponent extends HomeBaseComponent {}
```

Home Web Component

```
<hobbit-fitness-pal-header
  title="HobbitFitnessPal"
></hobbit-fitness-pal-header>

<p class="headline">
  Ensuring you don't miss any important meals of the day!
</p>

<div class="overview-preview">
  <div *ngFor="let ov of trackingOverview$ | async"
    [ngClass]="{'tracked': ov.tracked}"
    class="overview-box"
    [title]="ov.meal">
    </div>
  </div>
  <a
    routerLink="/meal-add"
    class="add-link-btn"
    >
    +
  </a>
```

Home Nativescript Component

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

import { HomeBaseComponent } from '@hobbit-fitness-pal/xplat/features';

@Component({
  moduleId: module.id,
  selector: 'hobbit-fitness-pal-home',
  templateUrl: './home.component.html'
})
export class HomeComponent extends HomeBaseComponent {

}
```

Home Nativescript Component

```
<DockLayout>
    <GridLayout dock="bottom" rows="auto" columns="*">
        <Button text="+" nsRouterLink="/meal-add" class="add-link-btn" horizontalAlignment="right" height="45" width="45"></Button>
    </GridLayout>
    <StackLayout class="p-20 text-center" dock="top" stretchLastChild="true">
        <hobbit-fitness-pal-header title="HobbitFitnessPal"></hobbit-fitness-pal-header>
        <Label text="Ensuring you don't miss any important meals of the day!" class="h2 headline" textWrap="true"></Label>
        <WrapLayout orientation="horizontal" class="overview-preview">
            <Label *ngFor="let ov of trackingOverview$ | async" [ngClass]="{'tracked': ov.tracked}" class="overview-box" height="90" width="90"></Label>
        </WrapLayout>
    </StackLayout>
</DockLayout>
```

Up and Running

Serve the web app

```
$ nx serve web-hobbit-fitness-pal
```

Up and Running

Run the native app

```
$ nx run nativescript-hobbit-fitness-pal:ios  
$ nx run nativescript-hobbit-fitness-pal:android
```

Let's see it in action.

Meal Add Base Component

```
import { Directive } from '@angular/core';
import { Router } from '@angular/router';
import { BaseComponent, TrackService } from '@hobbit-fitness-pal/xplat/core';
import { Meal } from '@hobbit-fitness-pal/xplat/core';

@Directive()
export abstract class MealAddBaseComponent extends BaseComponent {

  mealName: string;

  constructor(
    protected router: Router,
    private trackService: TrackService
  ) {
    super();
  }

  trackMeal(meal: Meal): void {
    this.trackService.trackMeal(meal);
  }
}
```

Meal Add Web Component

```
import { Component, ElementRef, OnInit, ViewChild } from '@angular/core';
import { Meal } from '@hobbit-fitness-pal/xplat/core';
import { MealAddBaseComponent } from '@hobbit-fitness-pal/xplat/features';

@Component({
  selector: 'hobbit-fitness-pal-meal-add',
  templateUrl: 'meal-add.component.html',
})
export class MealAddComponent extends MealAddBaseComponent implements OnInit {

  @ViewChild('video')
  public video: ElementRef;

  @ViewChild('canvas')
  public canvas: ElementRef;

  showMessage = false;

  public ngOnInit() {}

  public ngAfterViewInit() {
    if(navigator.mediaDevices && navigator.mediaDevices.getUserMedia) {
      navigator.mediaDevices.getUserMedia({ video: true }).then(stream => {
        this.video.nativeElement.srcObject = stream;
        this.video.nativeElement.play();
      });
    }
  }

  captureImage(): void {
    this.canvas.nativeElement.getContext('2d').drawImage(this.video.nativeElement, 0, 0, 640, 480);
    const image = this.canvas.nativeElement.toDataURL('image/png');
    const imageMeal: Meal = {
      image,
      name: this.mealName
    };
    this.trackMeal(imageMeal);
    this.showMessage = true;
  }
}
```

Meal Add Nativescript Component

```
import { Component } from '@angular/core';
import { MealAddBaseComponent } from '@hobbit-fitness-pal/xplat/features';
import { Dialogs, ImageAsset } from '@nativescript/core';
import * as camera from "@nativescript/camera";
import { Meal } from '@hobbit-fitness-pal/xplat/core';

@Component({
    moduleId: module.id,
    selector: 'hobbit-fitness-pal-meal-add',
    templateUrl: './meal-add.component.html',
})
export class MealAddComponent extends MealAddBaseComponent {

    showForm = false;

    ngOnInit(): void {
        camera.requestPermissions().then(
            () => { this.showForm = true; },
            () => { /* permission request rejected */}
        );
    }

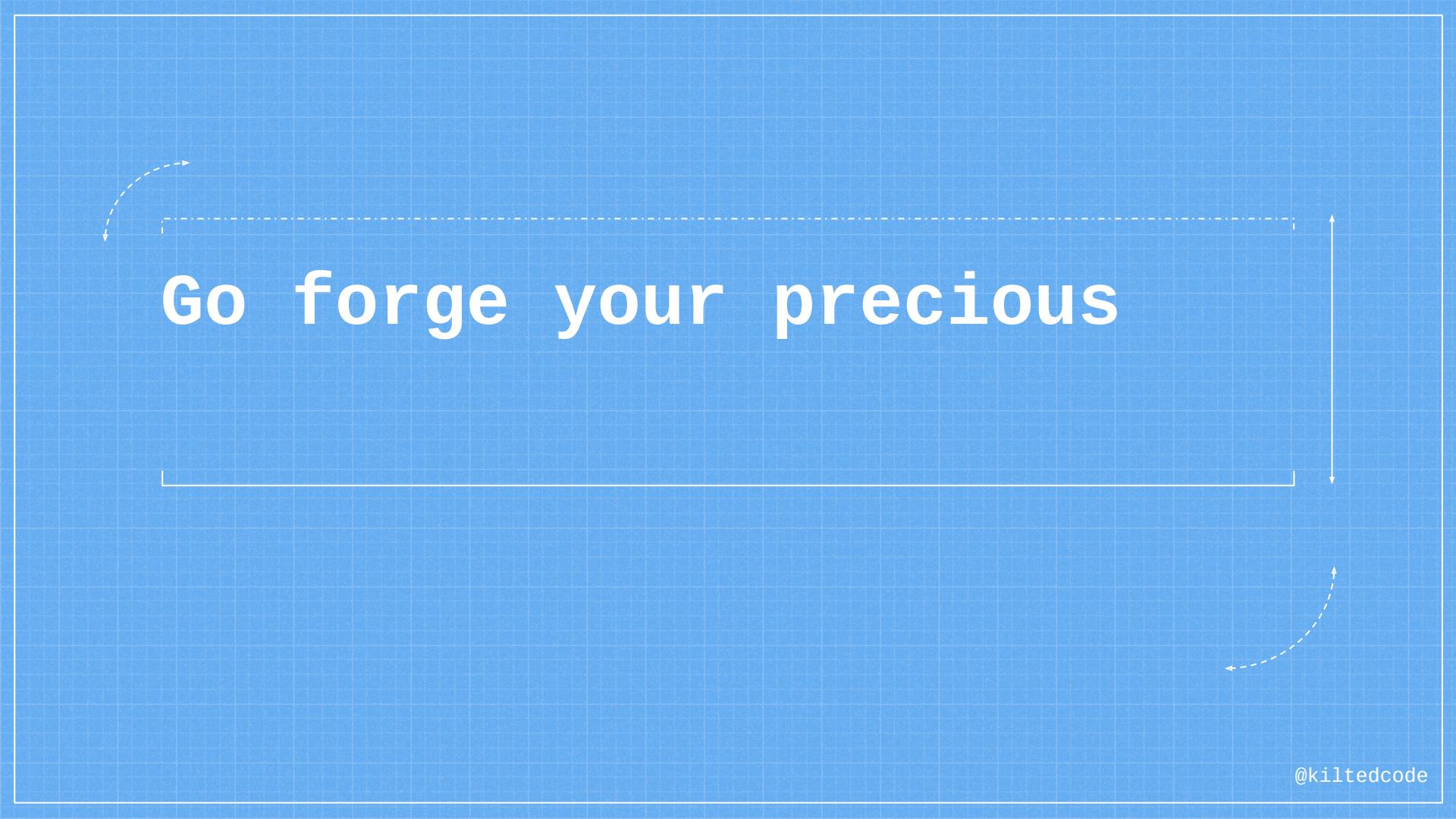
    captureImage(): void {
        camera.takePicture()
            .then((imageAsset: ImageAsset) => {
                const imageMeal: Meal = {
                    image: imageAsset.nativeImage,
                    name: this.mealName
                };
                this.trackMeal(imageMeal);

                const alertOptions = {
                    title: 'Meal Tracked',
                    message: 'Remember, stop before nightfall.',
                    okButtonText: 'Onward',
                    cancelable: false
                }

                Dialogs.alert(alertOptions).then(() => {
                    this.router.navigate(['/home']);
                })
            });
    }
}
```



**One source to rule them all,
One source to find them,
One source to bring them all
and in the JavaScript bind them.**



Go forge your precious



John Niedzwiecki
@kiltedcode

kiltedcode.github.io



Video background by Peter Gargiulo on [Unsplash](#)

@kiltedcode