"Once you stop learning, you start dying"

- Albert Einstein

RACHEL HEIMBACH

Developer

rheimbach@quintor.nl

@Rach_Nerd

HENK BAKKER

Developer

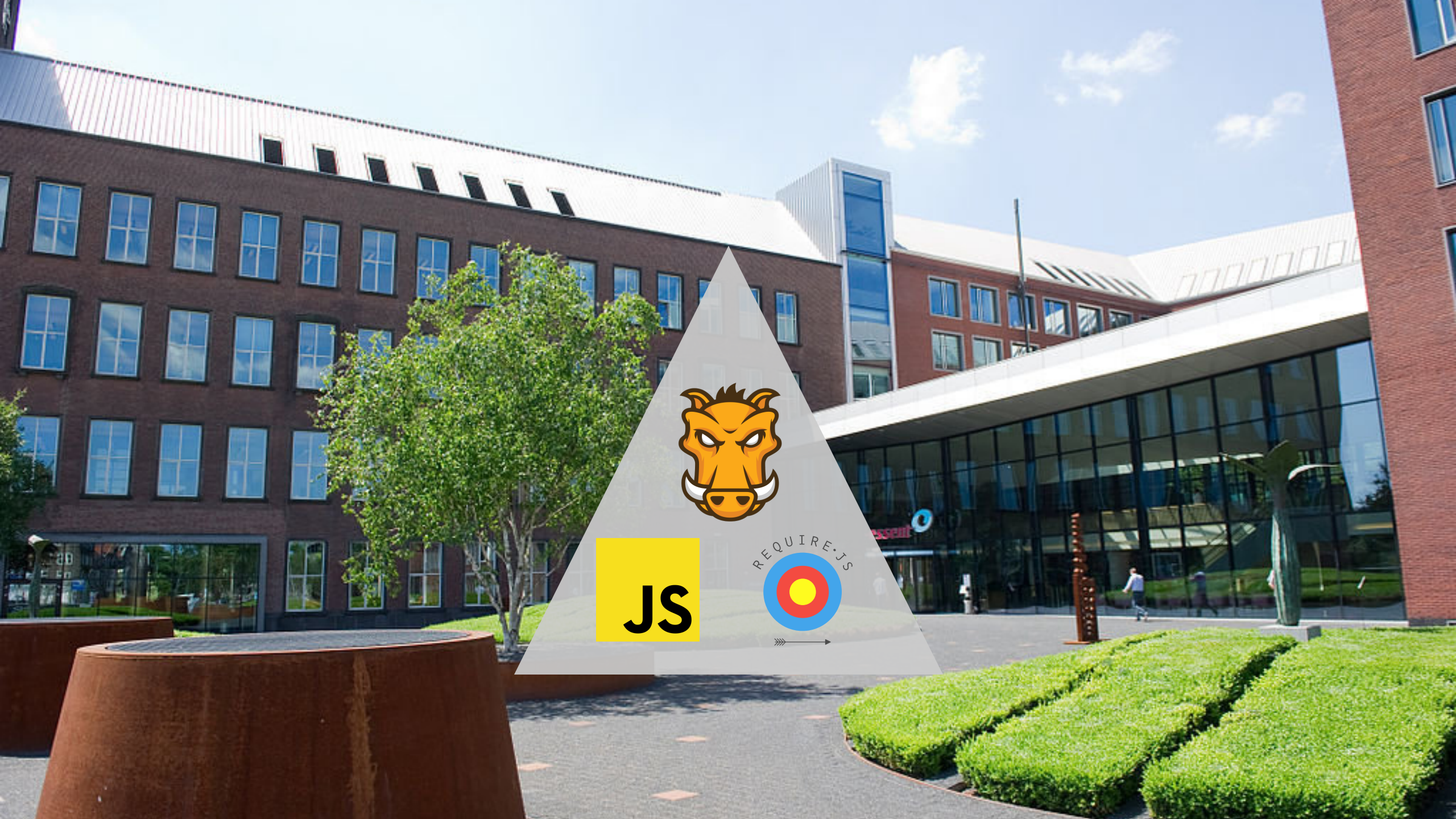
hbakker@quintor.nl

@spike1292

Quintor


essent


essent
Kim Janssen
uitloggen
Mijn verbruik
Mijn termijnbedrag
Uw termijnbedrag
Met uw huidige termijnbedrag komt uw rekening ongeveer op 0 uit.
Advies: Behoud uw termijnbedrag.
BespaarCoach
Beantwoord 5 vragen en ontdek waar de kansen liggen om het wooncomfort in uw huis
XML
energiedirect.nl
energiedirect.nl

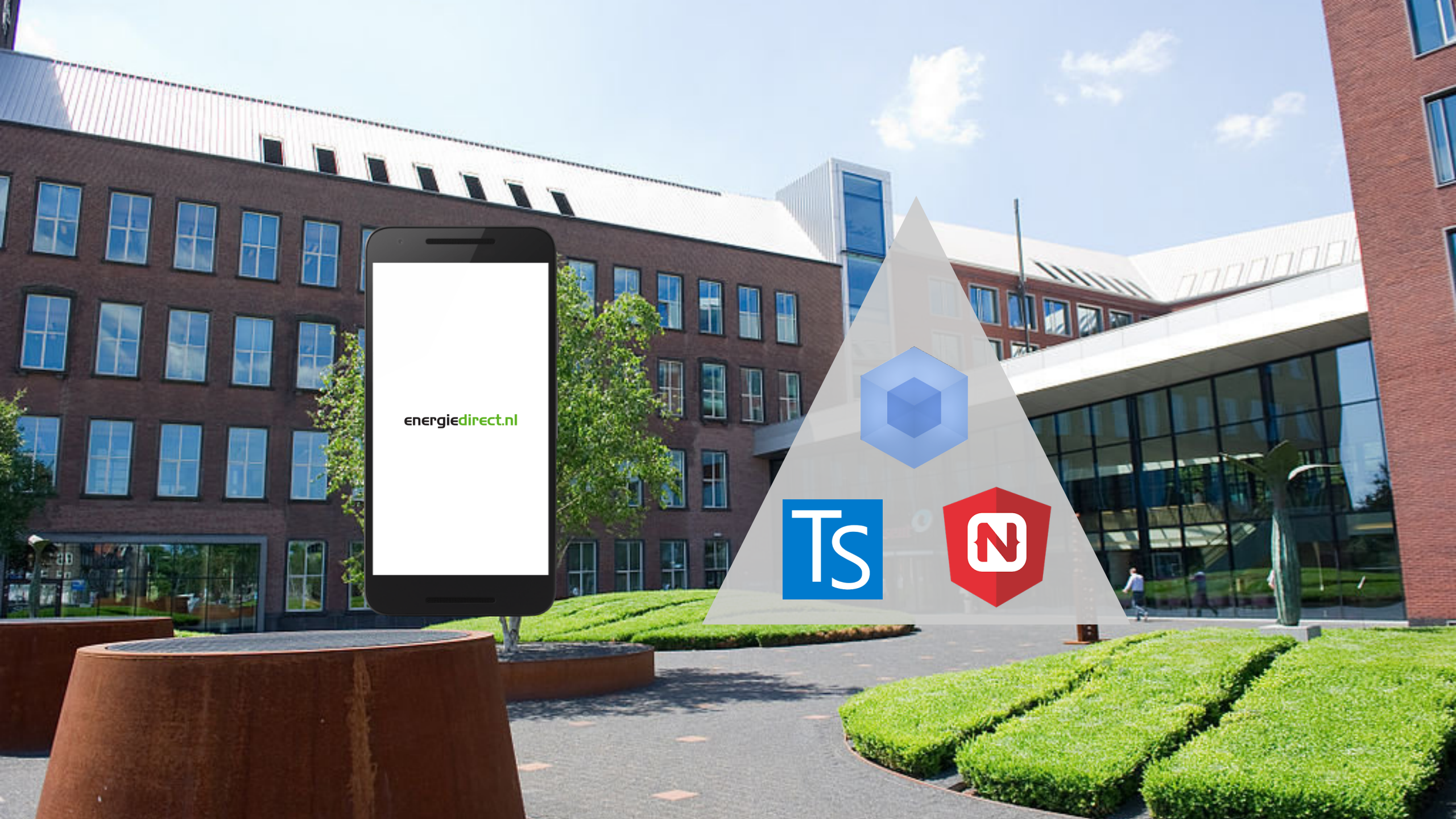

JS
REQUIRE.JS

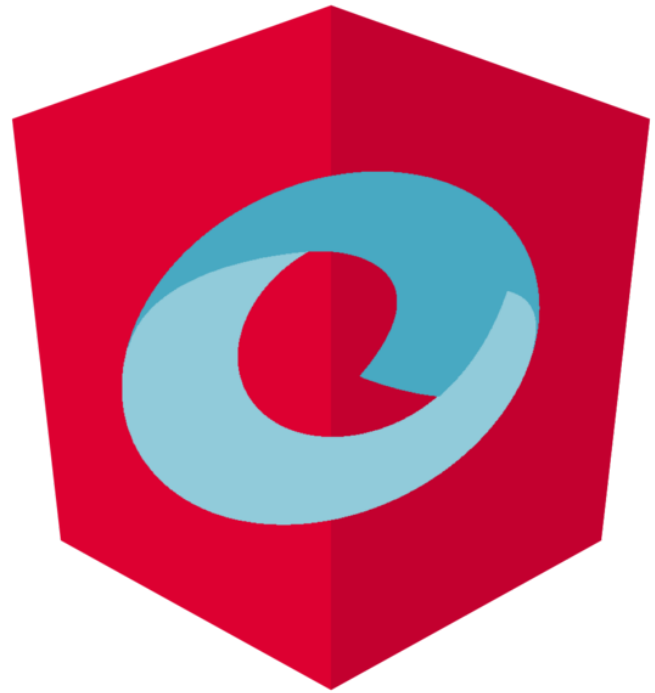

Angular 2
Big community
Popular framework
Scales well
Code share

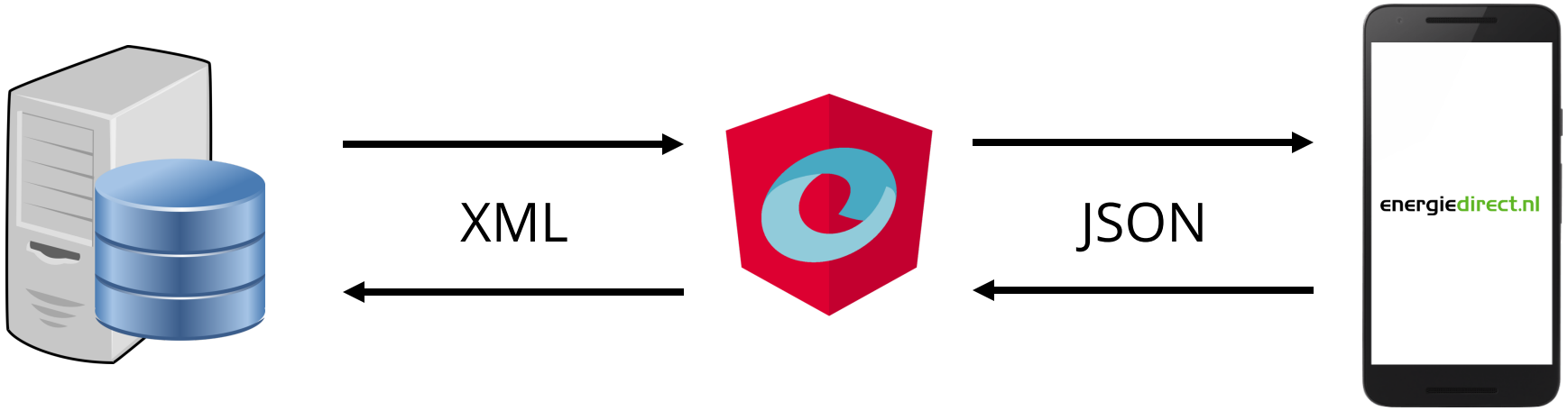

Ts
A

NativeScript

- iOS + Android app
- Native performance
- No webview
- Angular 2 integration


energiedirect.nl

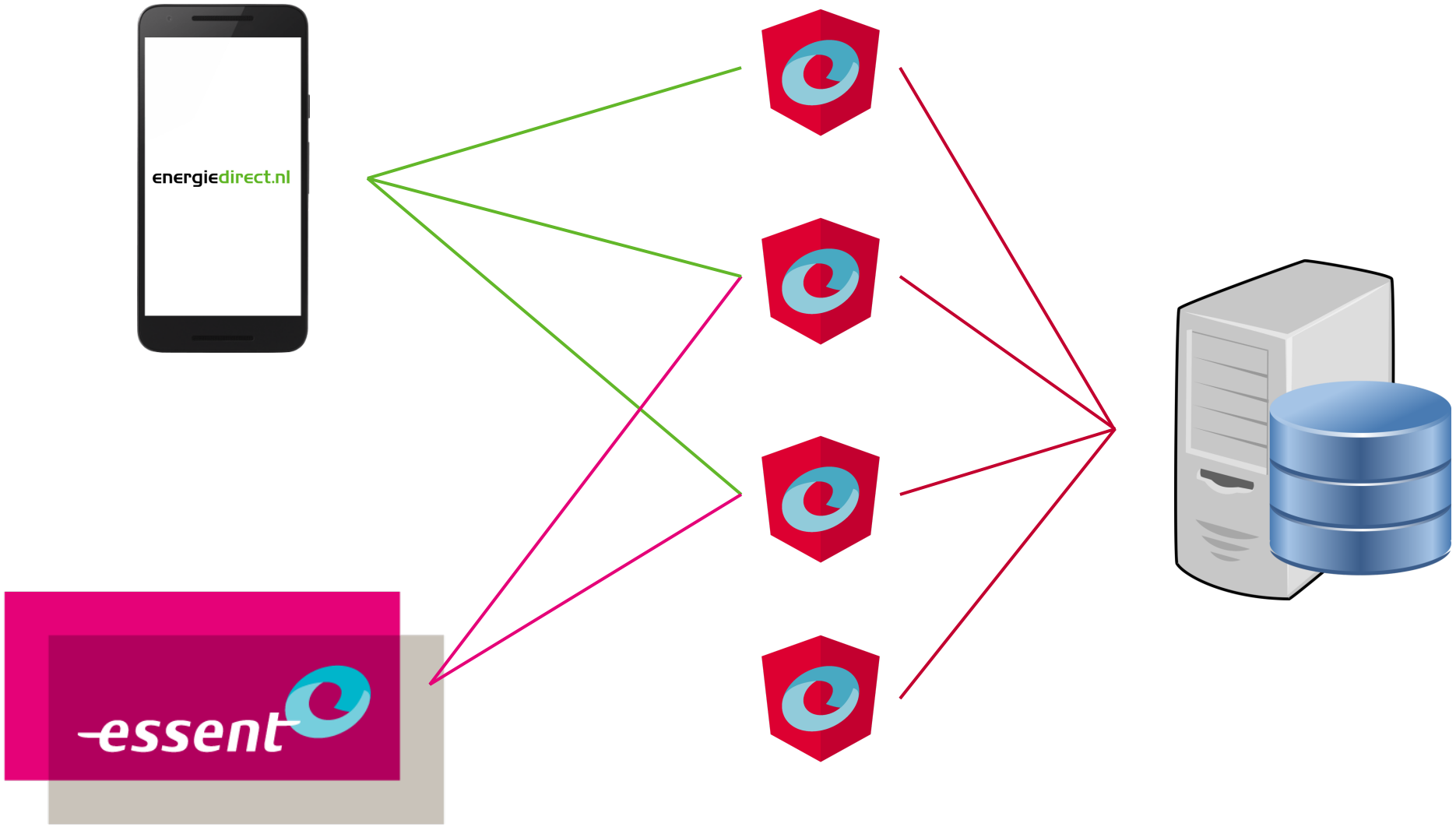



XML
JSON
energiedirect.nl

Application level

Essent modules

API Gateway

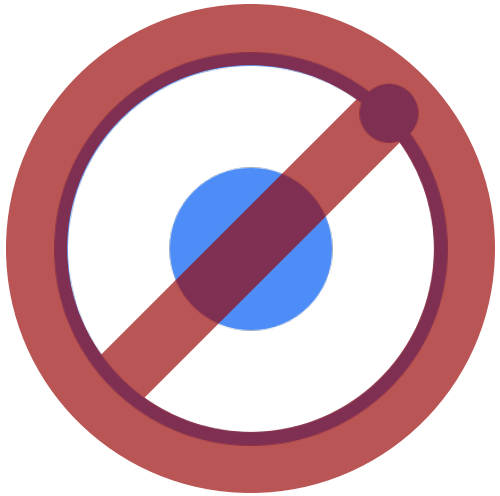


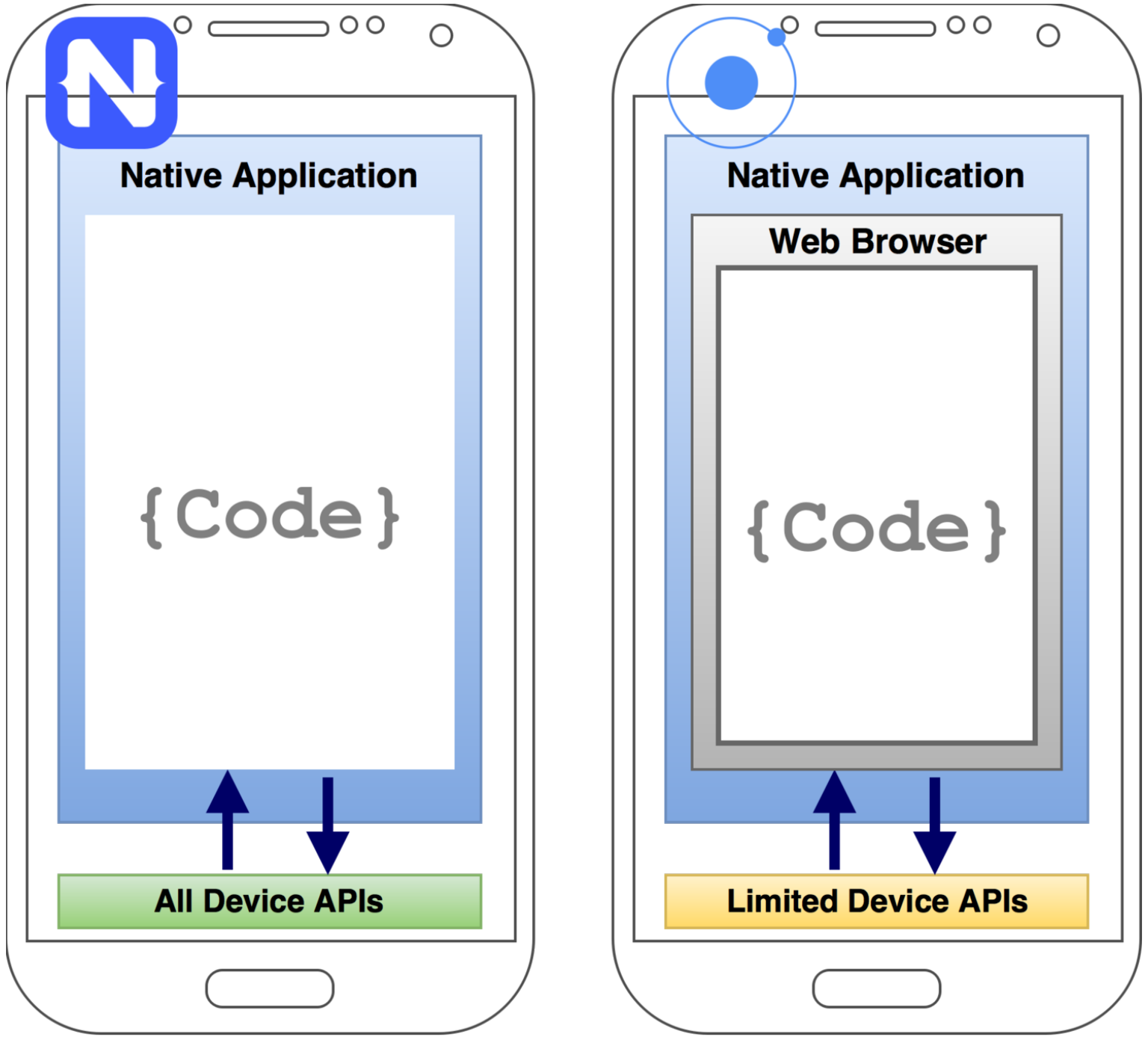
Code sharing

- Component-based
- Dependency injection
- Modules (Http, Forms, ...)
- Platform independent

This is **not** PhoneGap, Cordova or Ionic

- Real Native Components
- No DOM to manipulate
- Not HTML elements **styled** like native components




Native Application
{Code}
All Device APIs
Native Application
Web Browser
{Code}
Limited Device APIs

Use lots of prebuilt code

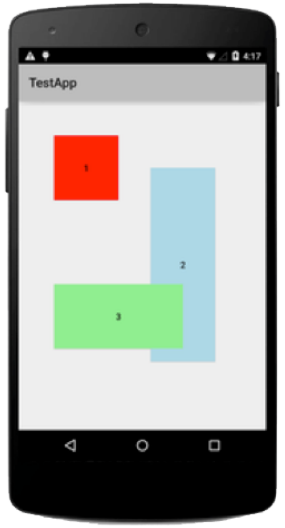
Native Layout Containers

Like ' ' - more concise though

Stack

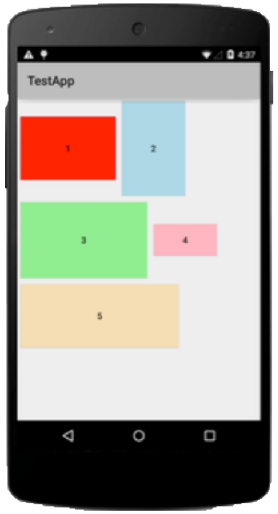
Grid

Absolute



Dock

Wrap



```
<Page xmlns="http://schemas.nativescript.org/tns.xsd">

  <StackLayout class="p-20">
    <Label text="Tap the button" class="h1 text-center"/>
    <Button text="TAP" tap="{{ onTap }}" class="btn btn-primary"/>
  </StackLayout>

</Page>

// DOM comparison...

<html>
<body>

  <div class="p-20">
    <h1 class="h1 text-center">Tap the button</h1>
    <button type="button" class="btn btn-primary"
        (tap)="onTap()">TAP</button>
  </div>

</body>
</html>
```

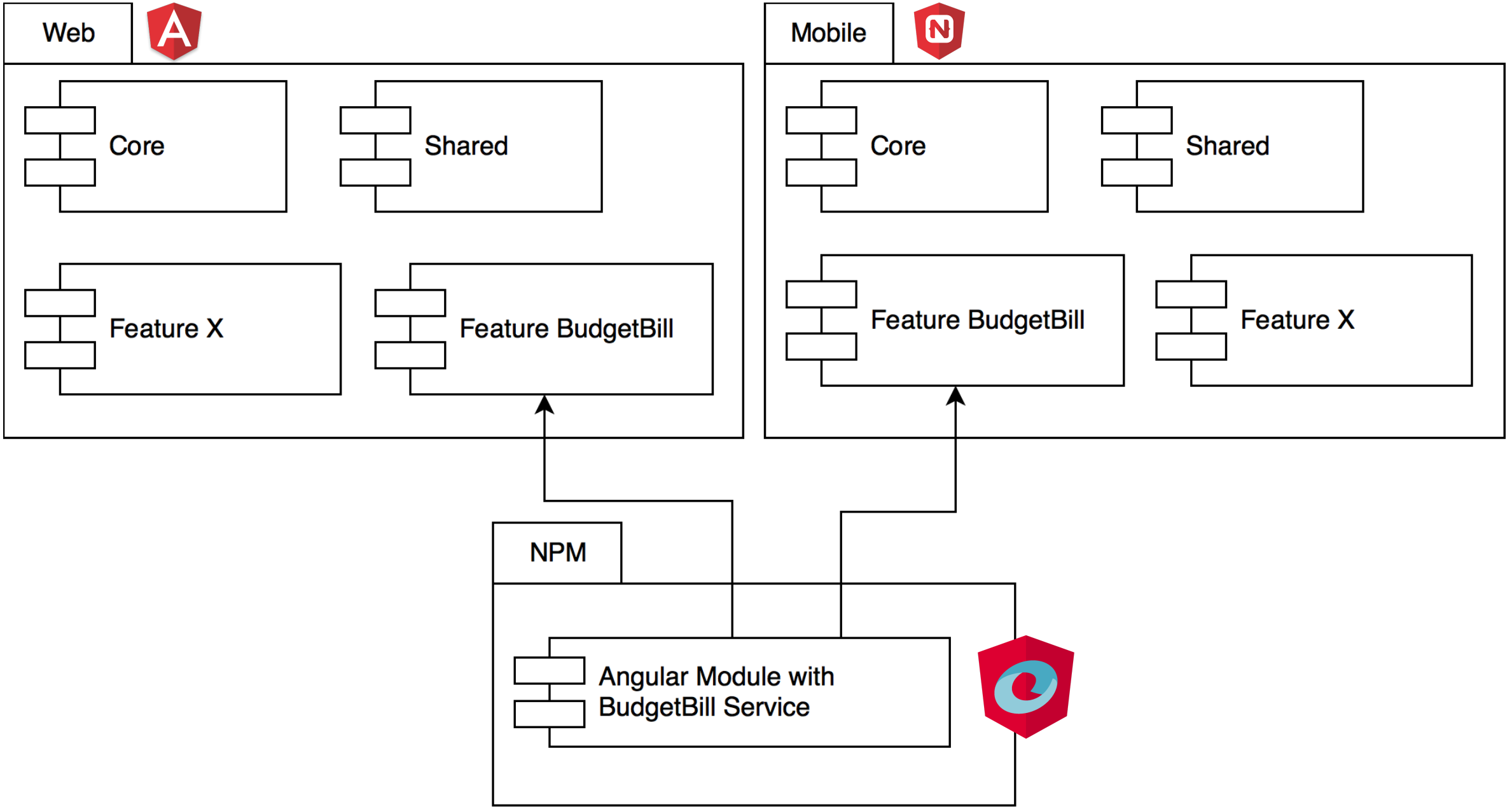
VM JS `const switch = new Switch();`

Nativescript modules — Switch module

Nativescript runtime




NativeScript Modules


Web
Core
Shared
Feature X
Feature BudgetBill
Mobile
Core
Shared
Feature BudgetBill
Feature X
NPM
Angular Module with
BudgetBill Service

State management!?

?

RxJS

"...is a set of libraries to compose asynchronous and event-based programs using observable collections and Array style composition in JavaScript" - RxJS

Observable

"The Observer and Observable interfaces provide a generalized mechanism for push-based notification, also known as the observer design pattern." -RxJS

```
const observable$ = Observable.of('foo');

observable$
    .subscribe(
        value => console.log(value),
        error => console.error(error),
        () => console.log('Completed')
    );
// LOG: foo
// LOG: Completed

observable$
    .map(text => text + ' bar')
    .subscribe(
        value => console.log(value),
        error => console.error(error),
        () => console.log('Completed')
    );
// LOG: foo bar
// LOG: Completed
```

Subject

"An RxJS Subject is a special type of Observable that allows values to be multicasted to many Observers" - RxJs

```
const subject = new Subject();

subject
    .subscribe(
        value => console.log(value),
        error => console.error(error),
        () => console.log('Completed')
    );

subject.next('foo'); // LOG: foo
subject.next('bar'); // LOG: bar
subject.complete(); // LOG: Completed
subject.next('foo');
```

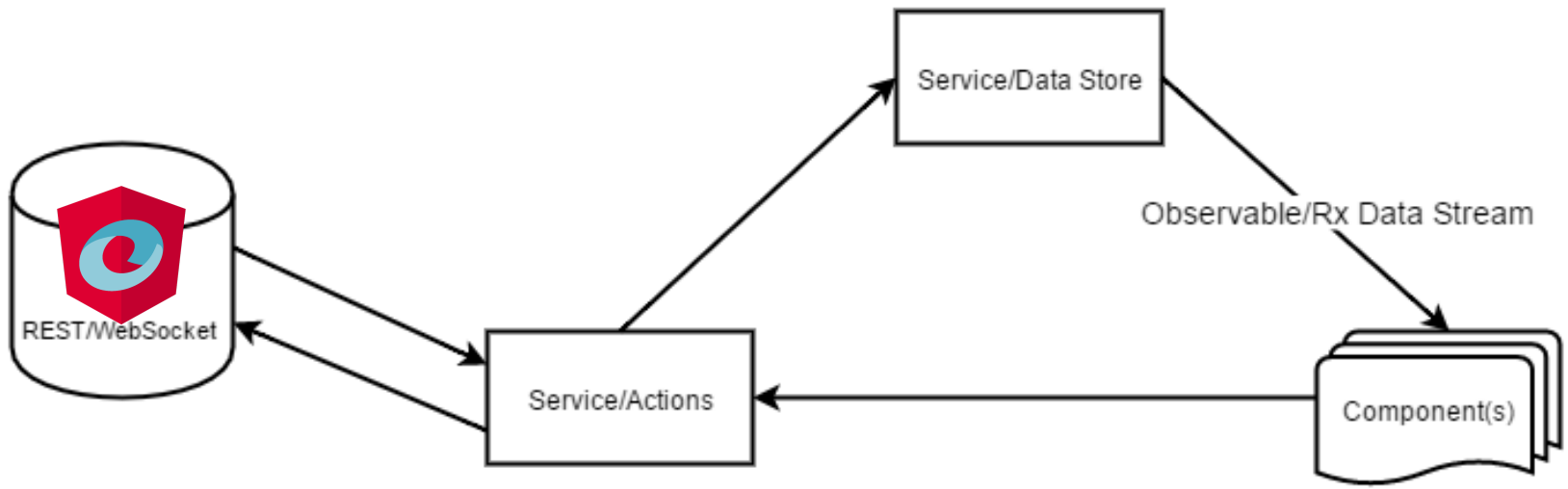
```
const subject = new Subject();
const observable$ = subject.asObservable();

observable$
    .subscribe(
        value => console.log(value),
        error => console.error(error),
        () => console.log('Completed')
    );


subject.next('foo'); // LOG: foo
subject.next('bar'); // LOG: bar

observable$.next('foo'); // Error
```

Angular 2 Observable Data Service



```
@Injectable()
export class BudgetBillService {
    amount$: Observable<number>;
    private budgetBillStateSubject: Subject<BudgetBillState>;

    constructor(private http: Http) {
        this.budgetBillStateSubject = new Subject<BudgetBillState>();
        this.amount$ = this.budgetBillStateSubject.asObservable()
            .map((bbState: BudgetBillState) => bbState.amount);
    }

    get(): void {
        this.http.get('assets/bbs.json')
            .map((res: Response) => res.json())
            .subscribe(
                (bbState: BudgetBillState) =>
                    this.budgetBillStateSubject.next(bbState),
                (error: Response) => console.error(error),
                () => console.log('Completed')
            );
    }
}
```

Streams are awesome!

Tips state management

- Keep state in streams
- Expose state as an observable
- Manipulate state by executing actions that push data through streams
- Never manually call error/complete on a stream you do not wish to close

Hot/Cold Observable

```
@Injectable()
export class TodosService {
    todos$: Observable<Todo[]>;
    private todosSubject: Subject<Todo[]>;

    constructor(private http: Http) {
        this.todosSubject = new Subject();
        this.todos$ = this.todosSubject.asObservable();
    }

    get(): void {
        this.http.get('assets/todos.json')
            .map((res: Response) => res.json())
            .subscribe(
                (todos: Todo[]) => this.todosSubject.next(todos),
                (error: Response) => console.error(error),
                () => console.log('Completed')
            );
    }
}
```

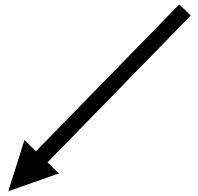
Hot

Cold

```
@Component({
    selector: 'q-todos',
    templateUrl: 'todos.component.html',
    styleUrls: ['todos.component.css']
})
export class TodosComponent implements OnInit {
    todos: Todo[] = [];

    constructor(private todosService: TodosService) {
    }

    ngOnInit() {
        this.todosService.todos$
            .subscribe(
                (todos: Todo[]) => this.todos = todos
            );
        this.todosService.get();
    }
}
```



Hot

Subscription

"A Subscription is an object that represents a disposable resource, usually the execution of an Observable." - RxJS

```
const subject = new Subject();

const observable$ = subject.asObservable();

const subscription = observable$
                        .subscribe(value =>
                            console.log(value)
                        );

subject.next('foo'); // LOG: foo
subscription.unsubscribe();
subject.next('bar');
```

```
@Component({
    selector: 'q-todos',
    templateUrl: 'todos.component.html',
    styleUrls: ['todos.component.css']
})
export class TodosComponent implements OnInit, OnDestroy {
    todos: Todo[] = [];
    private ngUnsubscribe = new Subject();

    constructor(private todosService: TodosService) {
    }

    ngOnInit() {
        this.todosService.todos$
            .takeUntil(this.ngUnsubscribe)
            .subscribe(
                (todos: Todo[]) => this.todos = todos
            );
        this.todosService.get();
    }

    ngOnDestroy() {
        this.ngUnsubscribe.next();
        this.ngUnsubscribe.complete();
    }
}
```

```
@Component({
    selector: 'q-todos',
    templateUrl: 'todos.component.html',
    styleUrls: ['todos.component.css']
})
export class TodosComponent implements OnInit {
    todos$: Observable<Todo[]>;

    constructor(private todosService: TodosService) {
    }

    ngOnInit() {
        this.todos$ = this.todosService.todos$
        this.todosService.get();
    }
}
```

```
<div *ngIf="todos$ | async as todos; else loading">
    <div *ngFor="let todo of todos">
        {{ todo.content }}
    </div>
</div>
<ng-template #loading>Loading...</ng-template>
```

Docs: https://angular.io/docs/ts/latest/api/common/index/NgIf-directive.html

Tips observables

- Identify hot observables
- Use async pipe to clean up hot subscriptions
- Unsubscribe remaining hot subscriptions onDestroy()
- Make use of operators like .take(1) to convert a hot streams to cold streams

Demo Time

Resources

- RxJS 5 Thinking Reactively Ben Lesh (Youtube)
- How to build Angular apps using Observable Data Services (Angular University)
- How NativeScript works (Telerik)
- Building Apps with NativeScript and Angular (NativeScript)