**Angular チュートリアル：Tour of Heroes**

**1. ヒーローエディター**

ライフサイクルハックについて(ngOnInitなど)

<https://qiita.com/ksh-fthr/items/ccd9861f919c4aa30ae8>

コンポーネントにおける初期処理は ngOnInit で統一する

Componentの初期化時に発火する

双方向データバインディング

・FormModuleをオプトインして、[(ngModel)]で双方向にデータを流す

・コンポーネントの値とテンプレート上の値とを双方向に同期させるためのバインド構文を双方向バインディングと言います。

コンポーネントの値（プロパティ）が変化したらテンプレートにも反映し、テンプレートの値を変えたら（例えばテキストボックスに値が入力されたら）コンポーネントのプロパティにも書き戻す、そんなバインド構文です。

疑問

・ライフサイクルハックとは？

・データバインディングとは？

・ディレクティブとは？

**2. リストの表示**

\*ngIf は、表示するメッセージが存在する場合のみメッセージエリアを表示します ← 存在しているときにのみ表示するため、定義していなくても動作するようになる

\*ngFor は、<div> 要素をくり返してメッセージリストを表示します

selectedHero?: Hero;

onSelect(hero: Hero): void {

this.selectedHero = hero;

}

Hero型のオブジェクトselectedHeroを定義して、それにheroを代入する

[class.selected]="hero === selectedHero"

現在の行のヒーローが selectedHero と同じ場合、Angularは selected のCSSクラスを追加します。2つのヒーローが異なる場合には、Angularはそのクラスを削除します。

**3. フィーチャーコンポーネントの作成**

@inputについて

<https://qiita.com/masaks/items/677195b78379e0877e24>

@Input()は 親コンポーネントから子コンポーネントに値を引き渡す役割を持ちます。

対して@Output()は 子コンポーネントから親コンポーネントにイベント(値)を引き渡す役割を持ちます。

**4. サービスの追加**

@injectable 及び providedIn: 'root

[https://qiita.com/ringtail003/items/d4329f7e77ce70ccc2d0 - インジェクターの指定](https://qiita.com/ringtail003/items/d4329f7e77ce70ccc2d0#インジェクターの指定)

このサービスクラスが他のサービスクラスやコンポーネントに注入可能である 事をマークアップしたものです。

そしてオブジェクトリテラルの providedIn: 'root' は アプリケーションのどこからでもこのサービスが利用可能な事 を示しています。

constructor

<https://www.it-mure.jp.net/ja/angular/コンストラクタとngoninitの違い/824013850/>

クラスがインスタンス化される時に実行され、クラスとそのサブクラス内のフィールドの適切な初期化を保証するクラスのデフォルトメソッドです。

クラスの初期化時に発火する。

コンストラクターではプロパティ定義などの簡単な初期化のみを行い、それ以外は 何もするべきではありません 。

**Observable HeroService**

getHeroes(): Observable<Hero[]>

{ const heroes = of(HEROES); return heroes; }

TypeScriptの文法理解が足りない。

ここの部分は再読。

ObservableとRxJSライブラリについて調べる。

なぜ、わざわざObservableとofを使うのか？ ← リクエストに対してレスポンスを受け取る行為は処理コストが高いため、任意のタイミングでサーバー側からデータを送信できるObservableが使われている。

Observableはpush型通信で送られてくるデータを流れ（ストリーム）として捉え、pushが行われたタイミングでそのデータに対し処理を行うことができます。

<https://qiita.com/Yamamoto0525/items/efc0e5617c7427fea86a>

<https://qiita.com/agajo/items/7942743a0130f7a0f30b> ← Subscribe関数についても記述されているので読むべし。

**HeroesComponentでのSubscribe**

再読。

コールバックとは？

**5. ナビゲーションの追加**

**AppRoutingModuleを追加する**

・Angularのベストプラクティスは、ルートのAppModuleからインポートされるルーティング専用のトップレベルモジュールで、 ルーターをロードして管理することです。←appmoduleからルートを管理するモジュールをインポートして、ルート管理を単一のファイル上で行うことで管理しやすくする

・routermoduleとroutesはアプリケーションにルーティング機能を持たせることができる

**ルート**

const routes: Routes = [

{ path: 'heroes', component: HeroesComponent }

];

ユーザーがクリックしてきた時、pathはurlを決めて、componentは表示するコンポーネントを決める

**RouterModule.forRoot()**

・全部分からない

・forRootメソッドについて調べる

**デフォルトルートを追加する**

{ path: '', redirectTo: '/dashboard', pathMatch: 'full' },

空のパス(最初のページ)の時は、パスがdashbordに一致するルートにリダイレクトする

## ルーティング可能な HeroDetailComponent

・全体的に何を言っているのか理解できない

**6. サーバーからのデータ取得**

genId(heroes: Hero[]): number { return heroes.length > 0 ? Math.max(...heroes.map(hero => hero.id)) + 1 : 11; }

この関数について詳しく調べる。

**ヒーローとHTTP**

・logメソッドでラップとは

**ヒーローを更新する**

httpOptions = {

headers: new HttpHeaders({ 'Content-Type': 'application/json' })

};

・httpリクエストの時、特別なヘッダーを期待とは？

AsyncPipe

Angularの[AsyncPipe](https://angular.jp/tutorial/api/common/AsyncPipe)が識別され、Observableが自動的にサブスクライブされるため、 コンポーネントクラスでこれを行う必要はありません。 ← ？？

・AsyncPipeとは？？

**RxJS subjectのsearchTerms**

・Subject？