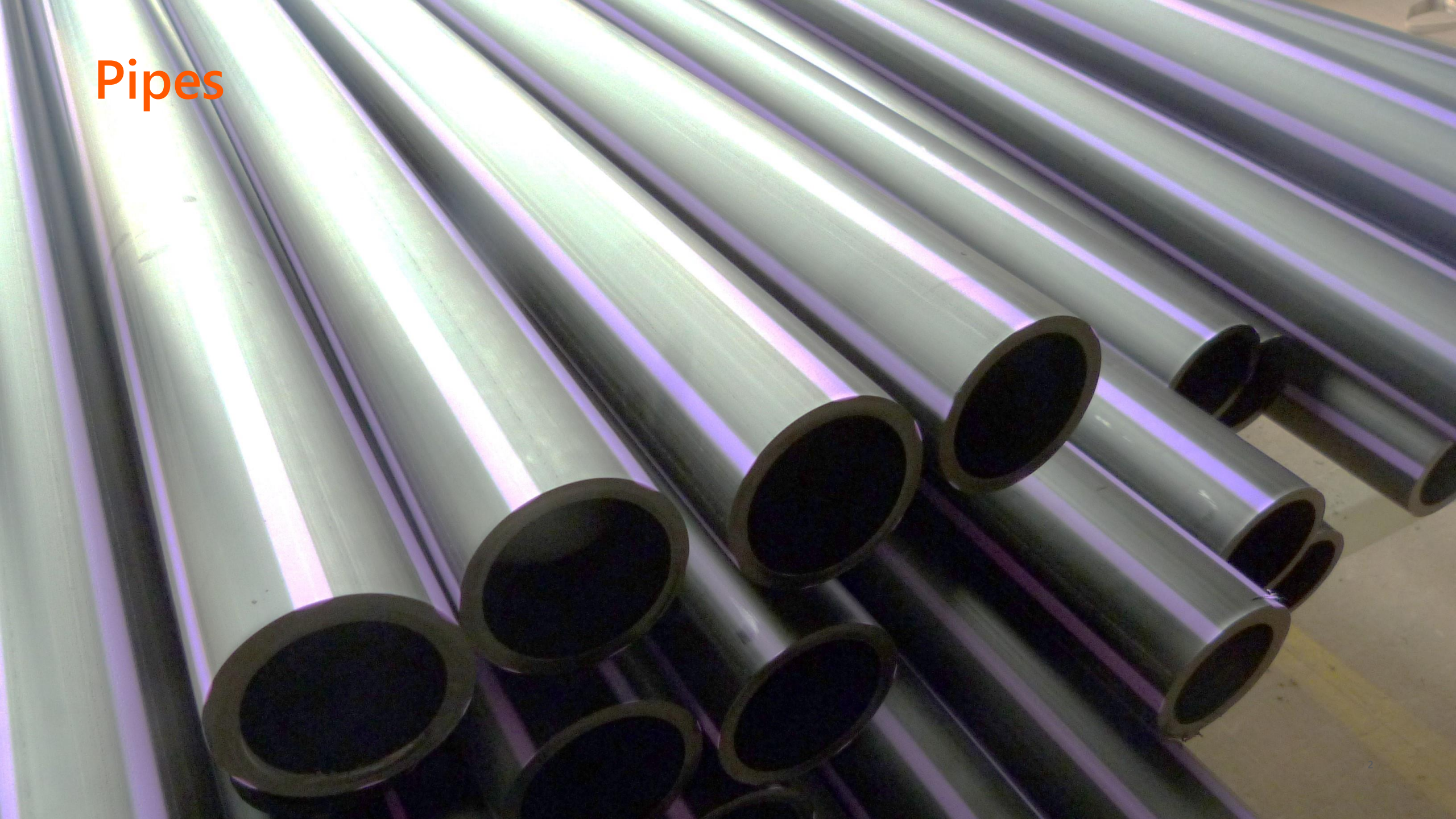
Angular

Workshops part 2

avanade

Pipes



Pipes

Pipes are helpers that transform your data in template based on some rules.

Built in pipes and custom pipes.

Usage

As simple as that:

```
<p>{{ today | date:'dd/MM/yyyy'}}</p>
```

Pipe is used by „|" sign. This particular „date" pipe transforms value in date format into readable date. We are passing format in quotes after ‚:' sign.

We could of course register our own pipes and use it across app/multiple apps.

EX: Use https://angular.io/api/common/DecimalPipe to format number 123.234234 and 0.1 to show exactly two numbers after dot.

Async Pipe

Probably most important pipe included in angular. It automaticaly subscribe and unsubscribe if needed to an observable.

```
<div *ngIf="customers$ | async as customers">
  <div *ngFor="let customer of customers">
    <span>{{ customer.firstName }}</span>
  </div>
</div>
```

Customers are Observable of customer array - consider it as some code for which we need to wait until it finish.

Directives

Directives are special „in template" information how to treat template blocks.

They allow us to add some common behaviour to our template.

For example we can easily add or remove some block of template (loader for example) by changing flag and use directive.

avanade

Directives

Structural Directives

These are responsible for changing DOM

```
<p *ngIf="true">
   stay in DOM
</p>
<p *ngIf="false">
   Off the DOM
</p>
```

Attribute Directives

These changing appearance or behavior

```
<p appHighlight>Highlight me!</p>
```

avanade

Exercise

Create a button which call a method toggleWarning.

Method will toggle visibility of paragraph with your custom warning message – realese creativity ☺

avanade

Exercise

Create a button which call a method toggleWarning.

Method will toggle visibility of paragraph with your custom warning message – realese creativity ☺

Tips:

```
<p *ngIf="true">
   stay in DOM
</p>
<p *ngIf="false">
   Off the DOM
</p>
```

(click)=„myFunc()”

avanade

Loop me baby one more time!

*ngFor is another usefull directive.

It alows us to duplicate content but not code.

Image that you have collection objects of the same type but with different values.

There is where *ngFor comes to an action.

```
<div *ngFor="let element of [1, 2, 3]”>
        {{ element }}
</div>
```

Loop me baby one more time!

```
*<div *ngFor="let hero of heroes; let i=index; let odd=odd; trackBy: trackByFunction"
[class.odd]="odd">
        {{ i }}. {{ hero.name }}
</div>
```

```
trackByFunction(index, item) {
    return index; // or item.id
}
```

Mock data

Add model for User to models.d.ts or wherever you want to put your models as interface.
Create place to store mock data and create const to store our array of objects.

In user-list component assign mock data to component property.

Display it in html using ngFor

But we created user component...

Yes we did!

Passing data between components could happen through multiple channels.

Comunicate between parent and direct children could be done using @Input and @Output.

To pass data from parent to children we need to do few things.

avanade

But we created user component...

1. Identify what we need to pass (is it collection or just one item?)
2. Use binding `[childUser]="user"`
3. Declare property childUser as `@Input() childUser: User;`
4. Move html into child
5. Enjoy :D

avanade

Some usefull resources and sources

https://angular.io/ there is literally everything (sometimes condensed)

https://material.angular.io/ great UI library working with hammer.js usefull when you dont feel comfortable with styles.

https://angular.io/api?type=pipe list of pipes included in angular

https://www.youtube.com/watch?v=jnp_ny4SOQE&feature=youtu.be&t=1320 ivy compiler which will be default in v9

avanade

Thank you

avanade