Angular Developer 4

example code

Exam ?

Angular Interview Questions & Answers

Which questions?

5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 21, 22, 24, 25, 26, 27, 28, 29, 30, 60, 62, 79

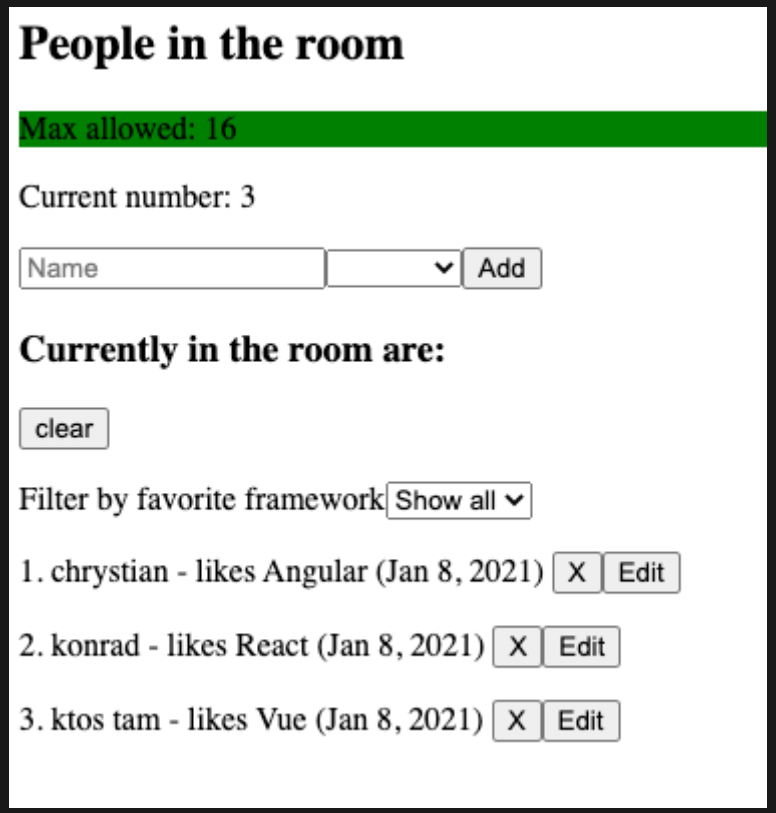
TASKS

'People in the room'

COMPONENTIZE

- Form component
- List component

LIST FILTERING



I want to be able to filter by favorite framework

Project

Two screens

- Intro page with intro text and player form
- Game page

Intro page

1. some quick introductory text
2. two inputs
    - player name
    - player email
3. start game button
4. upon clicking 'start' we check name and email and notify player whats wrong
5. if name and email are fine then store this data and move to game page

Game page

1. there should be a button 'exit game' which will move player to intro page
2. there should be nice, personalized welcome message (with player name)
3. integrate ngx-tetris
4. big indication of the game status (ready, started, paused...)
5. we need points counting mechanism (each cleared line counts)
6. display current amount of points
7. display time spent wile playing
8. there should be 'gameplay history' with all actions and each entry should have
    - timestamp
    - action name (player started the game, paused, line cleared...)
9. gameplay history should be
    - filterable by event type (ie. show only 'line cleared' events)
    - sortable by timestamp (latest first or oldest first)

Component driven

We do a lot in components

they can get long and ugly

One of the homeworks

```
I KNOW, BAD JOKE :P
```

ok, its fake :P

Solution?

More components

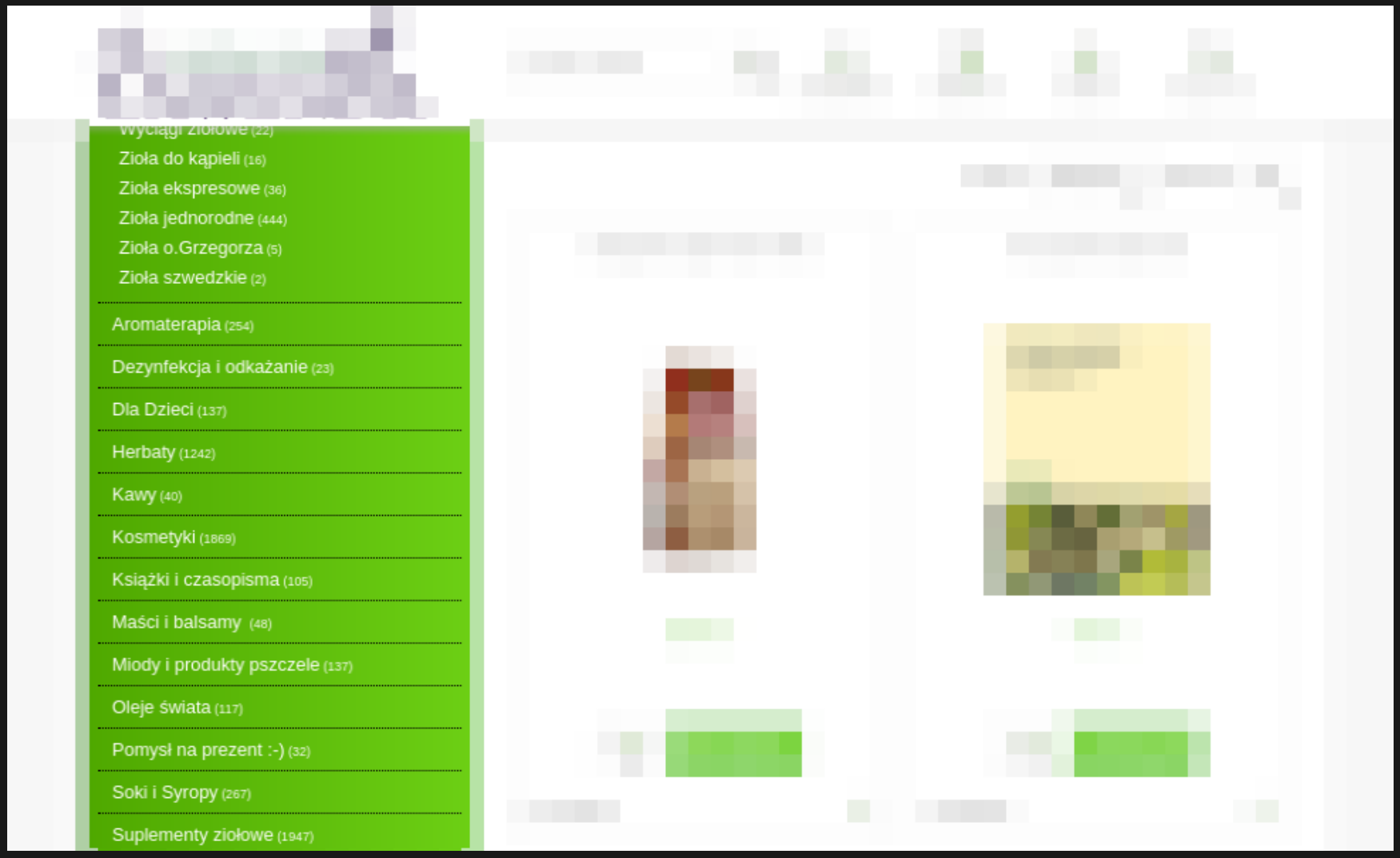
but smaller

LETS COMPONENTIZE

Creating components

```
$ ng generate component cart
```

```
CREATE src/app/cart/cart.component.css (0 bytes)
CREATE src/app/cart/cart.component.html (19 bytes)
CREATE src/app/cart/cart.component.spec.ts (612 bytes)
CREATE src/app/cart/cart.component.ts (267 bytes)
UPDATE src/app/app.module.ts (505 bytes)
```


Wyciągi ziołowe (22)
Zioła do kąpieli (16)
Zioła ekspresowe (36)
Zioła jednorodne (444)
Zioła o.Grzegorza (5)
Zioła szwedzkie (2)
Aromaterapia (254)
Dezynfekcja i odkażanie (23)
Dla Dzieci (137)
Herbaty (1242)
Kawy (40)
Kosmetyki (1869)
Książki i czasopisma (105)
Maści i balsamy (48)
Miody i produkty pszczele (137)
Oleje świata (117)
Pomysł na prezent :-) (32)
Soki i Syropy (267)
Suplementy ziołowe (1947)

Categories list/selector

```
// category-selector.component.ts
export class CategorySelectorComponent {
  public categories = [ ... ];
  public selected;

  onCategoryClicked(category): void {
    this.selected = category;
  }
}
```

```
<!-- category-selector.component.html -->
<p *ngFor="let category of categories">
  <button (click)="onCategoryClicked(category)">
    {{ category.name }} ({{ category.products.length }})
  </button>
</p>
```

sortuj po: domyślnie pokazuj po: 30

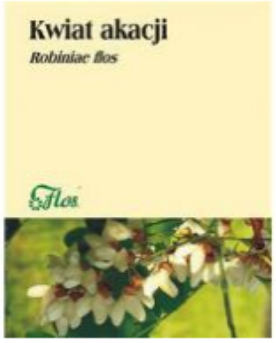
Adaptomix 50g Natura Wita

15.90 zł

do koszyka

porównaj

Akacja kwiat 50g Flos



5.65 zł

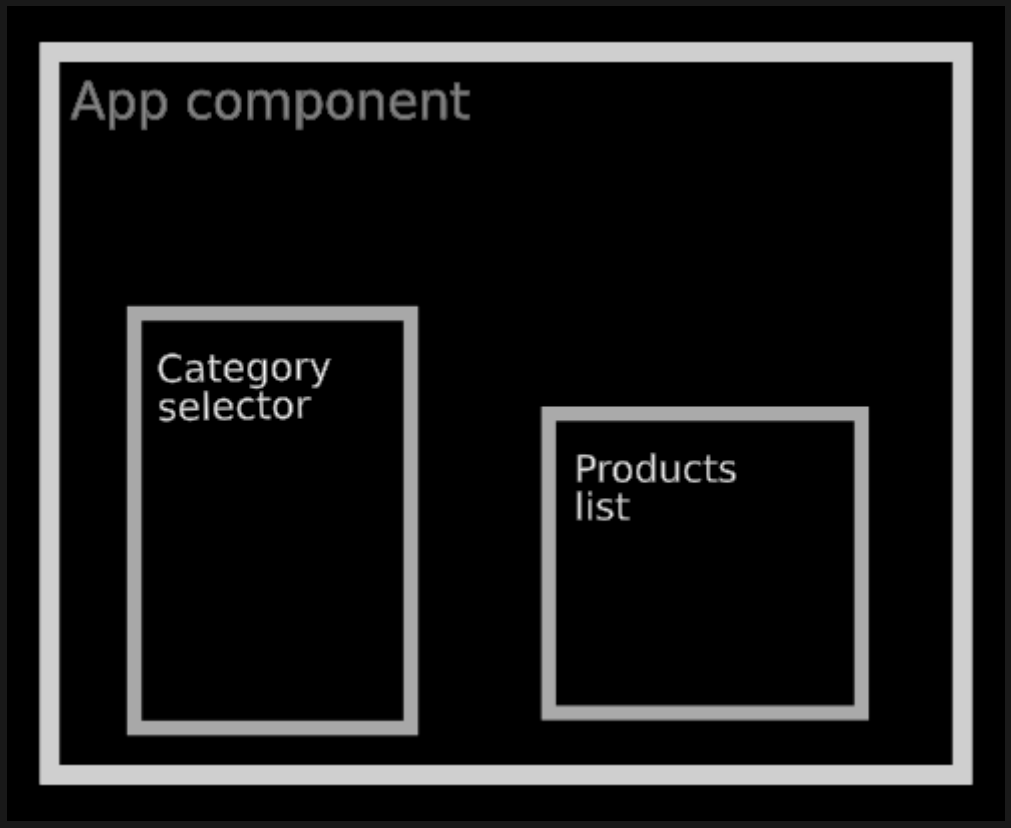
do koszyka

porównaj

Products list

```
// products-list.component.ts
export class ProductsListComponent {
  public products = [ ... ];
  public cart = [];

  addToCart(product): void {
    this.cart.push(product);
  }
}
```

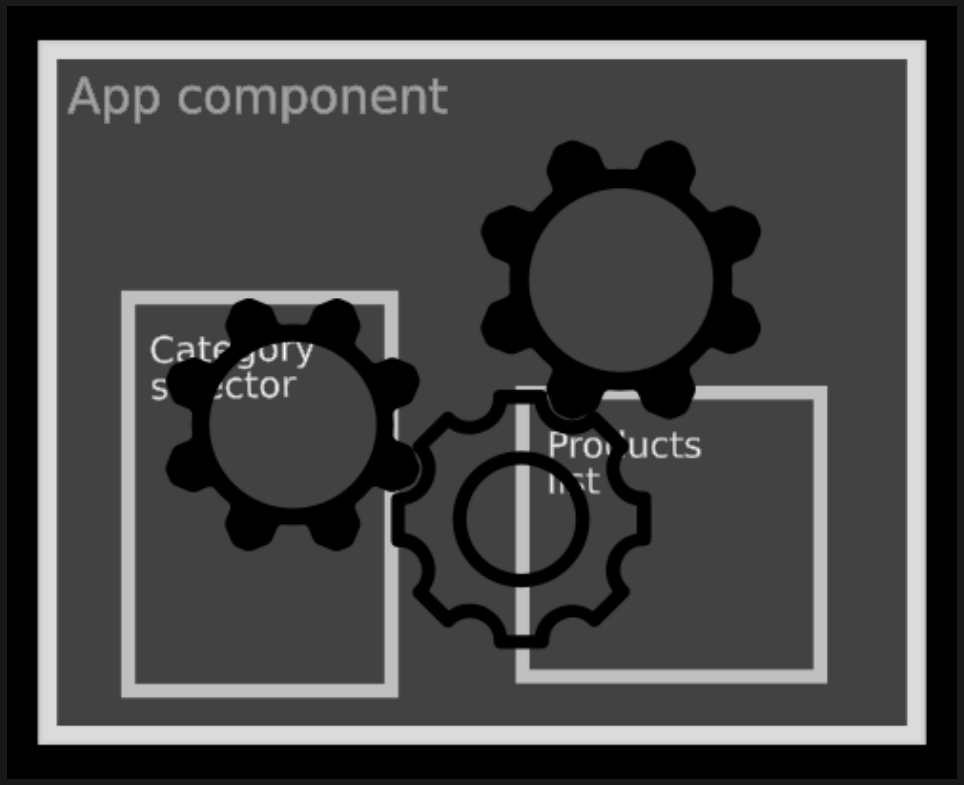
```
<!-- products-list.component.html -->
<div *ngFor="let product of products">
  <p>{{ product.name }}</p>
  <img [src]="product.imgUrl"/>
  <p>{{ product.price }}</p>
  <button (click)="addToCart(product)">
    Add to cart
  </button>
</div>
```


App component
Category selector
Products list

```
<!-- app.component.html -->
<div class="column-3">
  <!-- left column -->
</div>
<div class="column-9">
  <!-- right column -->
</div>
```

```
<!-- app.component.html -->
<div class="column-3">
  <app-category-selector>
    </app-category-selector>
</div>
<div class="column-9">
  <app-products-list>
    </app-products-list>
</div>
```

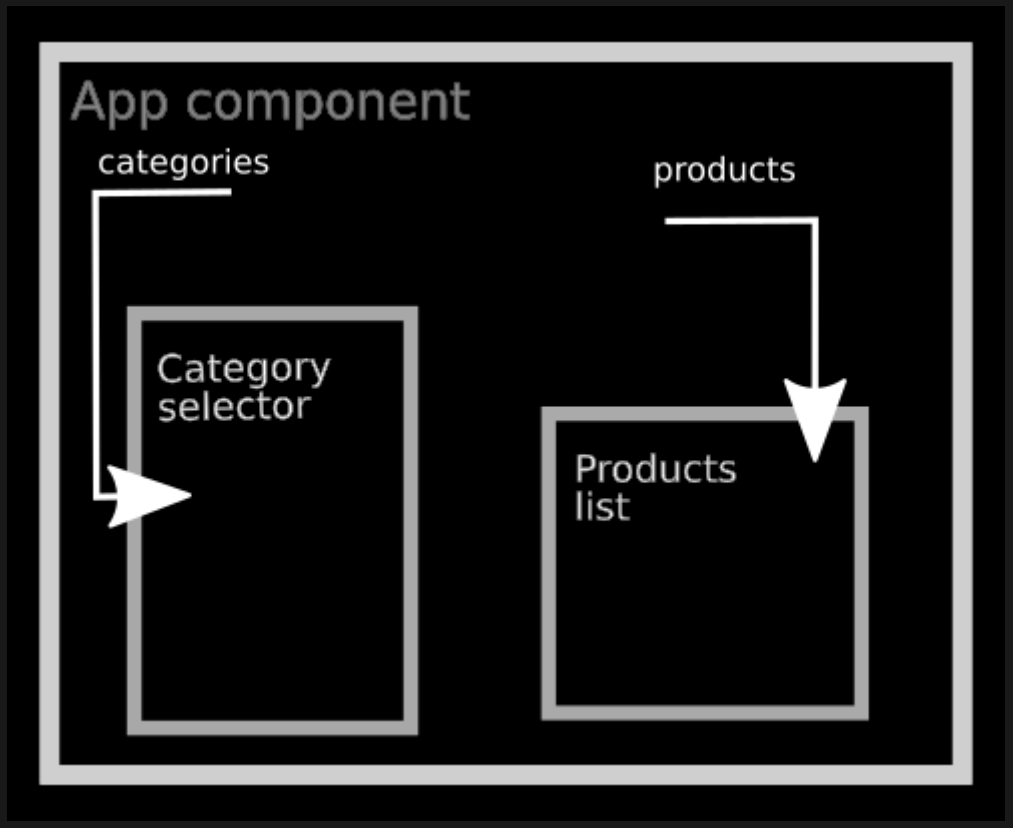
```ts
// definitions.ts
export interface Product {
  name: string;
  price: number;
}

export interface Category {
  name: string;
  products: Array<Product>;
}
```


App component
Category selector
Products list

Components communication

Passing data to components

(down)


App component
categories
products
Category selector
Products list

DATA SOURCE

App component

(parent)

```
// app.component.ts
export class AppComponent {
  public categories: Array<Category>;
  public products: Array<Product>;
}
```

DATA RECEIVERS

Category selector & Products list

(children)

How to pass the data?

INPUTS

@Input() decorator

Products list

```
1  // products-list.component.ts
2  import {Component, Input, OnInit} from '@angular/core';
3
4  export class ProductsListComponent {
5    @Input() public products: Array<Product>;
6    public cart = [];
7
8    addToCart(product): void {
9      this.cart.push(product);
10   }
11 }
```

Category selector

```
1  // category-selector.component.ts
2  import {Component, Input, OnInit} from '@angular/core';
3
4  export class CategorySelectorComponent {
5    @Input() public categories: Array<Category>;
6    public selected;
7
8    onCategoryClicked(category): void {
9      this.selected = category;
10   }
11 }
```

Wiring everything together

```
// app.component.ts
export class AppComponent {
  public allCategories: Array<Category>;
  public productsToDisplay: Array<Product>;
}
```

```
<!-- app.component.html -->
<div class="column-3">
  <app-category-selector [categories]="allCategories">
    </app-category-selector>
</div>
<div class="column-9">
  <app-products-list [products]="productsToDisplay">
    </app-products-list>
</div>
```

From parent to child

1. @Input()

Configure child component (data receiver) by decorating property with @Input() decorator

```
export class CartComponent {
  @Input() products: any;
}
```

2. [ ]

In parent component connect the data to child component using property binding

```
// shop.component.ts
export class ShopComponent {
   selectedProducts = [...];
}
```

```
<!-- shop.component.html -->
<div class="navbar">
  <cart [products]="selectedProducts">
    </cart>
</div>
```

PARENT

CHILD

CHILD

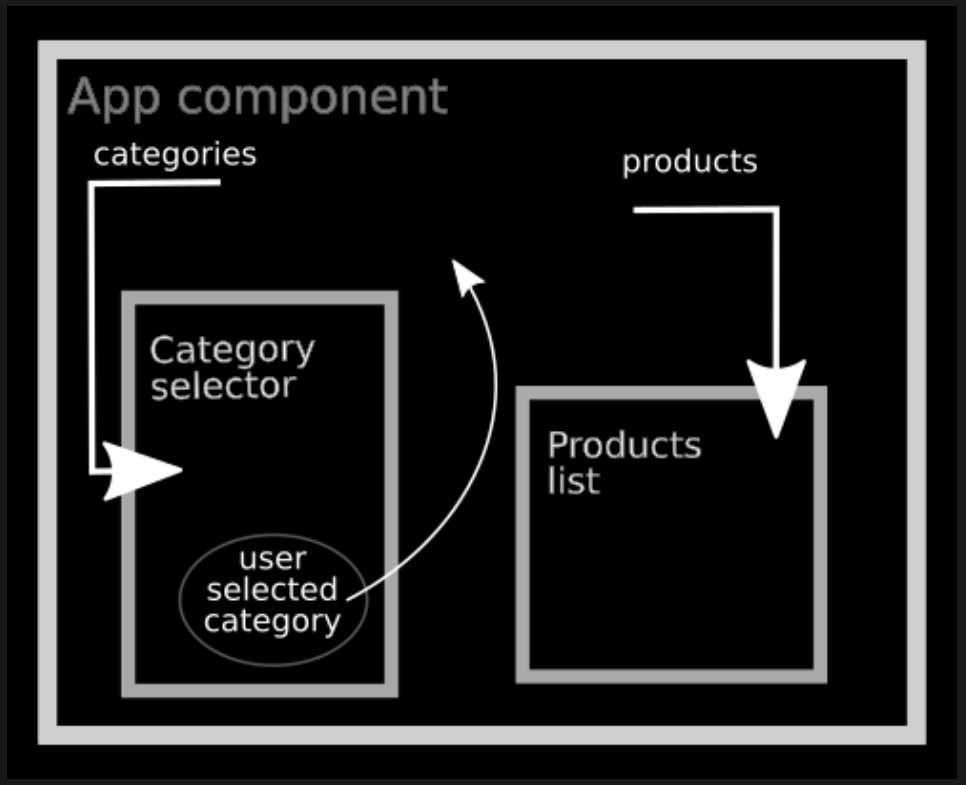
↓

PARENT

But how?

Sending data to parent

(up)


App component
categories
products
Category selector
Products list
user selected category

DATA SOURCE

Category selector

(child)

DATA RECEIVER

App component

(parent)

How to pass the data?

OUTPUTS

@Output() decorator

Emitting selected category

```
1  // category-selector.component.ts
2  import {Component, Output, EventEmitter, OnInit}
3                                          from '@angular/core';
4  export class CategorySelectorComponent {
5    public categories: Array<Category>;
6    @Output() selected = new EventEmitter<Category>();
7
8    onCategoryClicked(category): void {
9      this.selected.emit(category);
10   }
11 }
```

emitter? emitting?

Connecting child output to parent

```
<!-- app.component.html -->
<div class="column-3">
  <app-category-selector
    [categories]="allCategories"></app-category-selector>
</div>
...
```

Connecting child output to parent

```
<!-- app.component.html -->
<div class="column-3">
  <app-category-selector
    (selected)="onCategorySelected($event)"
    [categories]="allCategories"></app-category-selector>
</div>
...
```

```ts
// app.component.ts
export class AppComponent {
  public allCategories: Array<Category>;
  public productsToDisplay: Array<Product>;

  public onCategorySelected(category: Category) {
    this.productsToDisplay = category.products;
  }
}
```

From child to parent

1. @Output()

   Configure child component (event emitter) by decorating property with @Output() and add logic emitting the event

```
export class ProductsListComponent {
  @Output() buy = new EventEmitter<Product>();
  addToCart(product): void {
    this.buy.emit(product);
  }
}
```

2. ( )

   In parent component connect the data to child component using event binding

```
<app-products-list
  (buy)="addToCart($event)"
  [products]="productsToDisplay"></app-products-list>
```

```
export class AppComponent {
  public productsInCart: Array<Product> = [];
  public addToCart(product: Product) {
    this.productsInCart.push(product);
  }
}
```

```
@Output() buy = new EventEmitter<Product>();
```

Whats this 'Product' between < and > ?

TypeScripts GENERICS

```
export class ProductsListComponent {
  @Output() buy = new EventEmitter<Product>();

  addToCart(product): void {
    this.buy.emit(product);
  }
}
```

```
interface EventEmitter<T> {
  emit(value?: T): void;
}
```

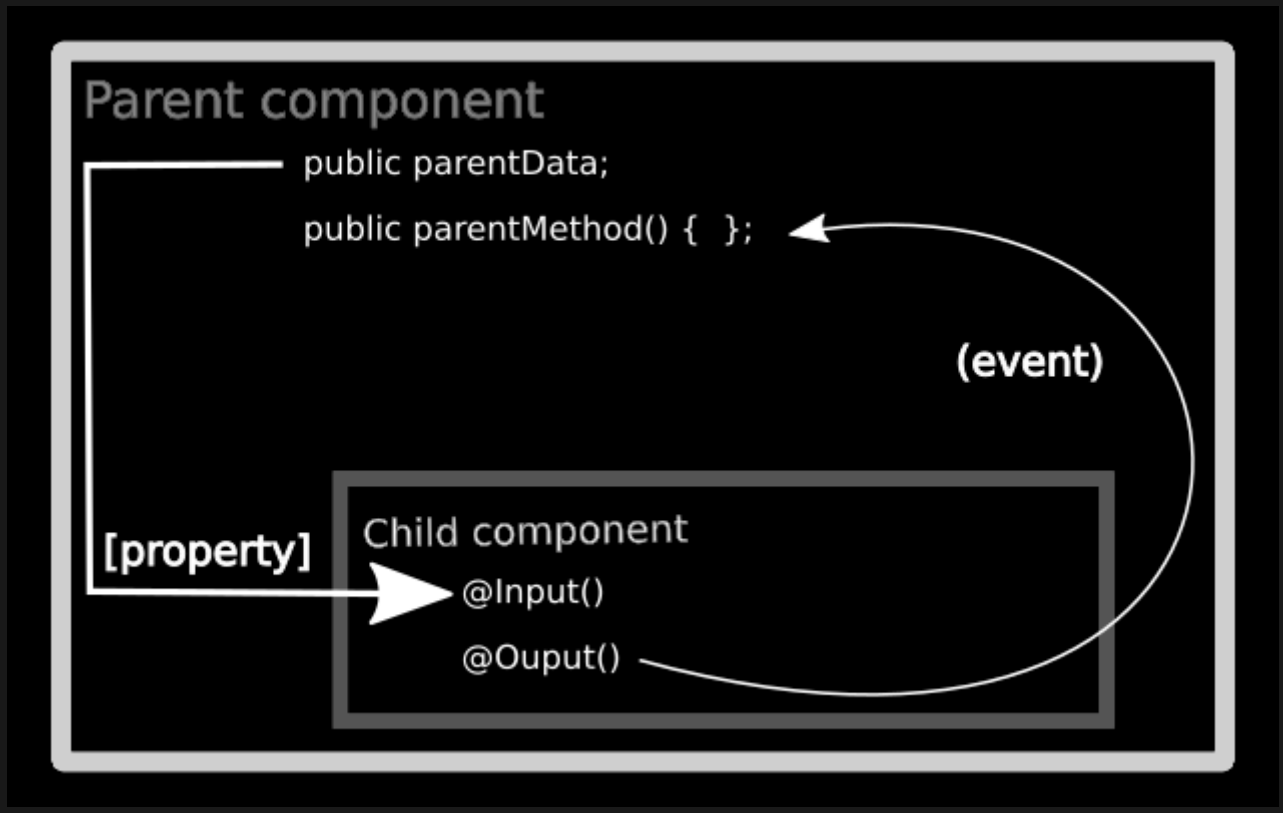
GENERICS

are

configurable types

COMPONENTS COMMUNICATION

Summary


Parent component
public parentData;
public parentMethod() { };
(event)
[property]
Child component
@Input()
@Ouput()

Input()

PARENT

```
export class ParentComponent {
  public rgb = ['red', 'blue', 'green'];
}
```

```
<child [colors]="rgb"</child>
```

CHILD

```
export class ChildComponent {
   @Input() colors: Array<string>;
}
```

```
<p *ngFor="let color of colors">
  {{ color }}
</p>
```

Output()

```
export class ParentComponent {
  public onChange(ev: boolean) {
    console.log('Month navigation');
    console.log('moving ' + ev + 'month');
  }
}
```

```
<child (change)="onChange($event)"</child>
```

```
export class ChildComponent {
  @Output()
  change = new EventEmitter<number>();
  public add(value) {
    this.change.emit(value);
  }
}
```

```
<button (click)="add(1)">Next</child>
<button (click)="add(-1)">Prev</child>
```

Imports

```
import { Input } from '@angular/core';
import { Output, EventEmitter } from '@angular/core';
```

PIPES

Transforming data on the fly

for displaying (in template)

via {{ }}

(interpolation)

Uppercase category name

```
<!-- category-selector.component.html -->
<p *ngFor="let c of categories">
  <button (click)="onCategoryClicked(category)">
    {{ c.name | uppercase }} ({{ c.products.length }})
  </button>
</p>
```

Format money

```
<!-- products-list.component.html -->
<div *ngFor="let product of products">
  <p>{{ product.name }}</p>
  <img [src]="product.imgUrl"/>
  <p>{{ product.price | currency }}</p>
  <button (click)="addToCart(product)">
    Add to cart
  </button>
</div>
```

Parametrized pipes

(configurable)

```
{{ product.price | currency }}
```

```
{{ product.price | currency:'USD' }}
<!-- $9.99 -->
```

```
{{ product.price | currency:'EUR' }}
<!-- €9.99 -->
{{ product.price | currency:'EUR':'eur' }}
<!-- eur9.99 -->
```

Chaining pipes

```
{{ product.name }}
<!-- bread -->
```

```
{{ product.name | uppercase }}
<!-- b -->
```

```
{{ product.name | uppercase | slice:0:1 }}
<!-- B -->
```

PIPES

1. Another template feature (via interpolation)
2. Used for data transformation
3. Wrapper for function
4. Transform input value

```
{{ pipeInput | somePipe }}
```

5. Can take parameters (separated by ':')

```
{{ pipeInput | somePipe:someParam }}
```

6. Chainable

```
{{ pipeInput | somePipe | anotherPipe }}
```

Pipes are 'change detection safe'

- Pure functions
- Engage on
  - input change (left side)
  - param change (right side)

BUILT IN PIPES

- currency
- uppercase
- date
- percent
- json
- **async**

Custom pipes

sortuj po: domyślnie pok 30

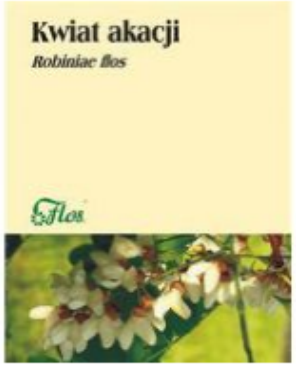
Adaptomix 50g Natura Wita

15.90 zł

do koszyka

porównaj

Akacja kwiat 50g Flos



5.65 zł

do koszyka

porównaj

Lets make some pipes

```
$ ng generate pipe sort
```

```
1 CREATE src/app/sort.pipe.spec.ts (179 bytes)
2 CREATE src/app/sort.pipe.ts (213 bytes)
3 UPDATE src/app/app.module.ts (431 bytes)
```

```ts
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'sort'
})
export class SortPipe implements PipeTransform {
  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

```
1  import { Pipe, PipeTransform } from '@angular/core';
2
3  @Pipe({
4    name: 'sort'
5  })
6  export class SortPipe implements PipeTransform {
7    transform(items: Product[], field, dir = "asc"): Product[] {
8      return null;
9    }
10 }
```

```
1  import { Pipe, PipeTransform } from "@angular/core";
2
3  @Pipe({
4    name: 'sort'
5  })
6  export class SortPipe implements PipeTransform {
7    transform(items: Product[], field, dir = "asc"): Product[] {
8      if (!items) {
9        return null;
10     }
11
12     return items.sort(...);
13   }
14 }
```

How to use our custom pipe?

```
// app.module.ts
@NgModule({
  declarations: [
    ...
    SortPipe
  ],
})
```

```
<!-- products-list.component.html -->
<div *ngFor="let product of products | sort:'price':'desc'">
  <p>{{ product.name }}</p>
  <img [src]="product.imgUrl"/>
  <p>{{ product.price | currency }}</p>
  <button (click)="addToCart(product)">
    Add to cart
  </button>
</div>
```

Custom pipe summary

- It is basically a function
  (wrapped in a special class)
- Decorated with @Pipe() decorator
- Listed in module declarations next to components