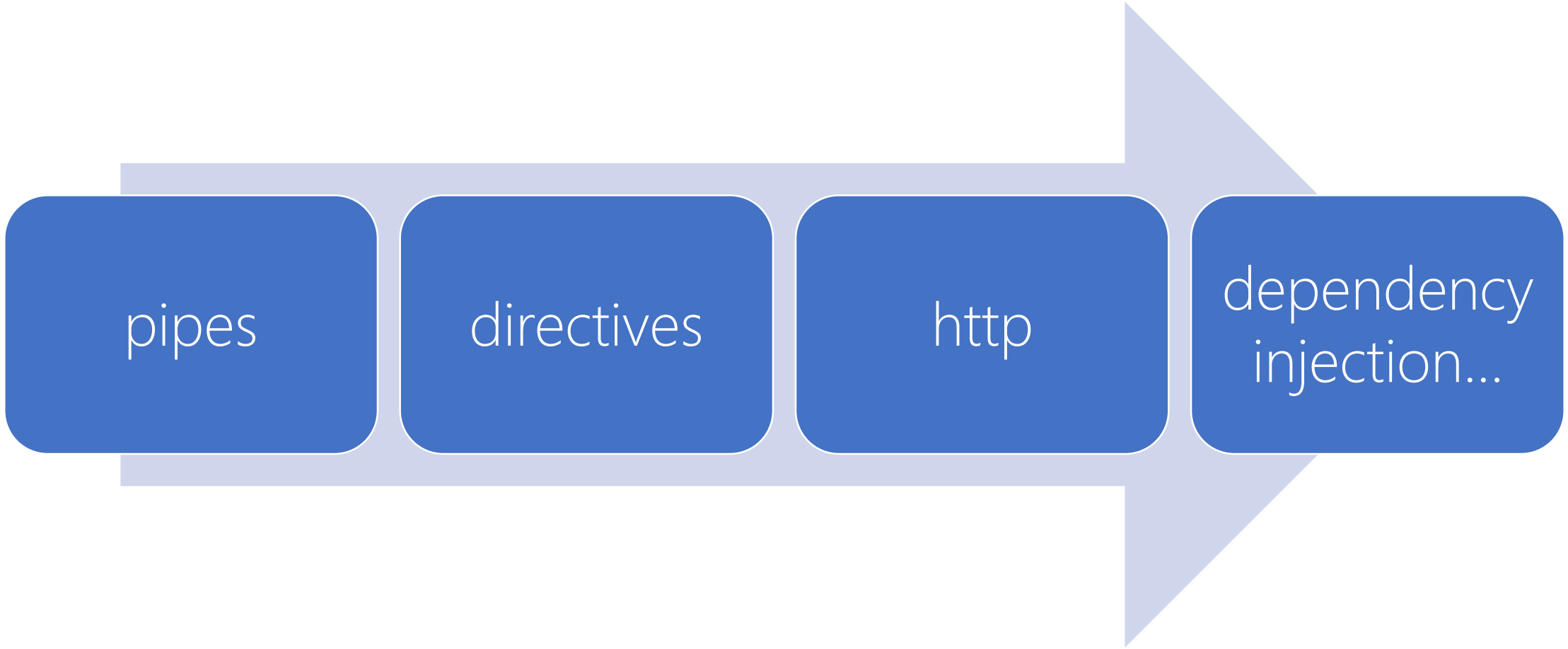
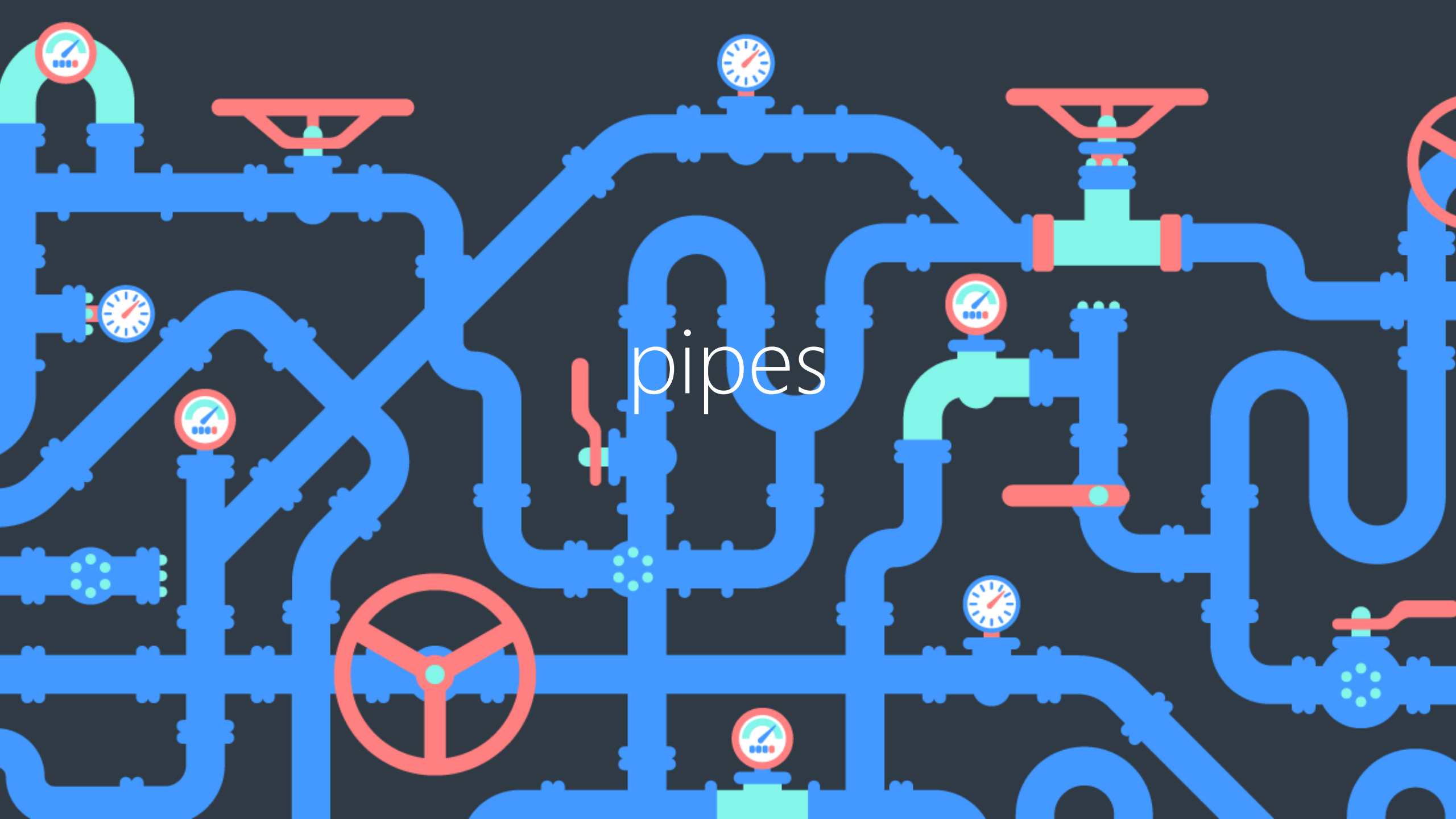
part 7

kamil mijacz

plan

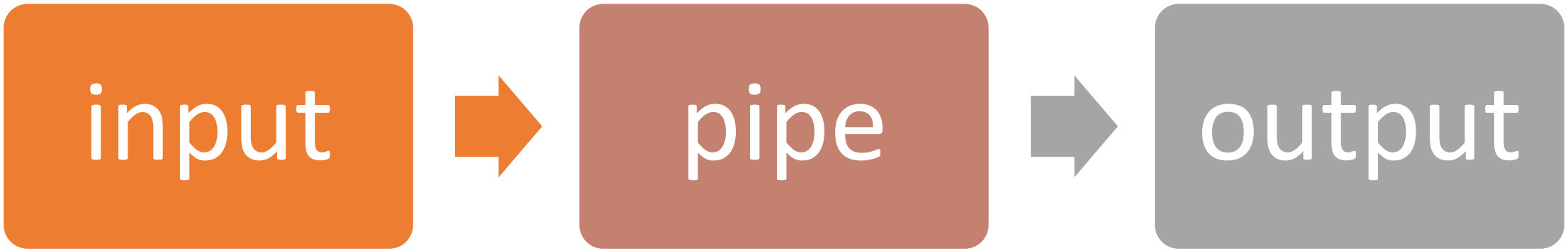



pipes

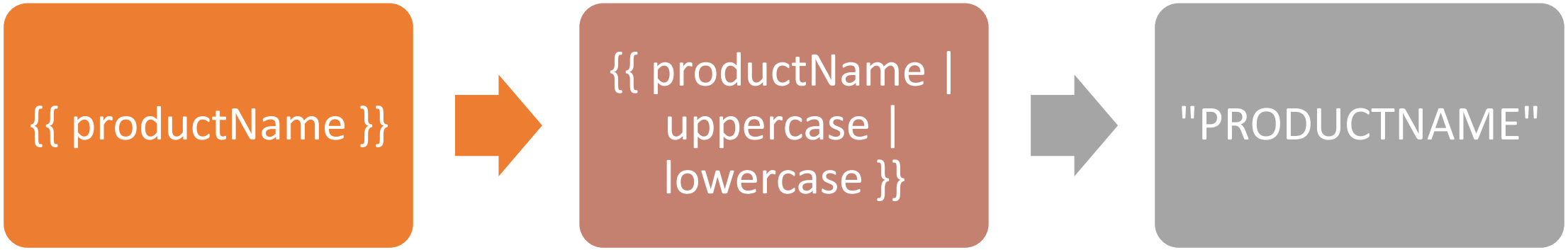
what is pipe?

- classes
- prepare data for display
- transforming text into another
- @Pipe decorator
- for code reuse

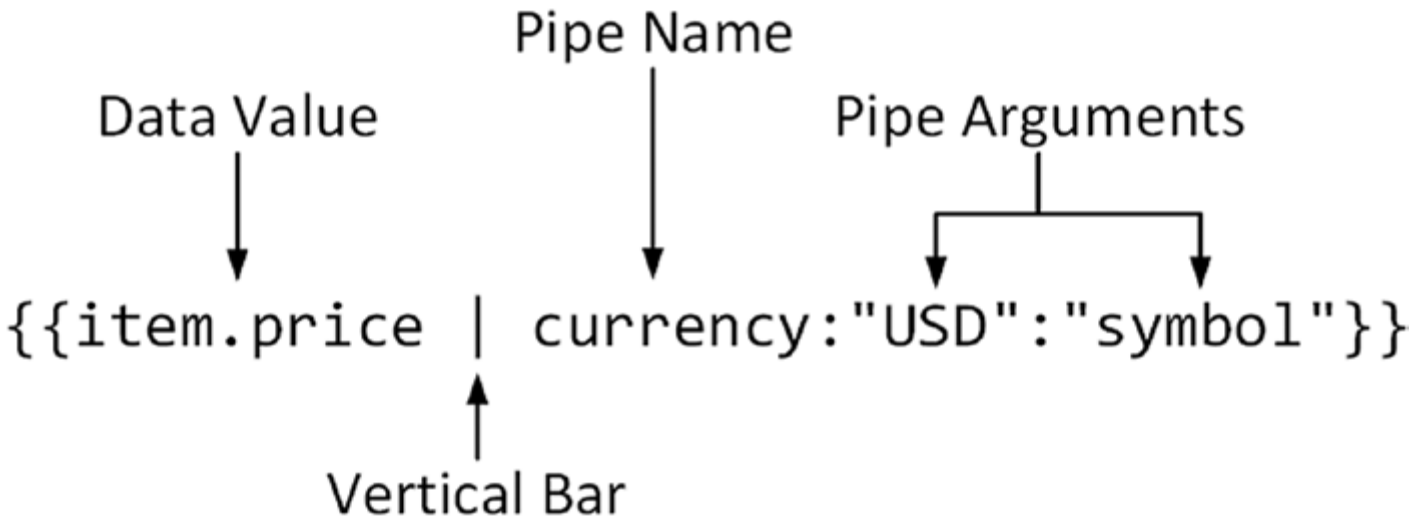
understanding pipe



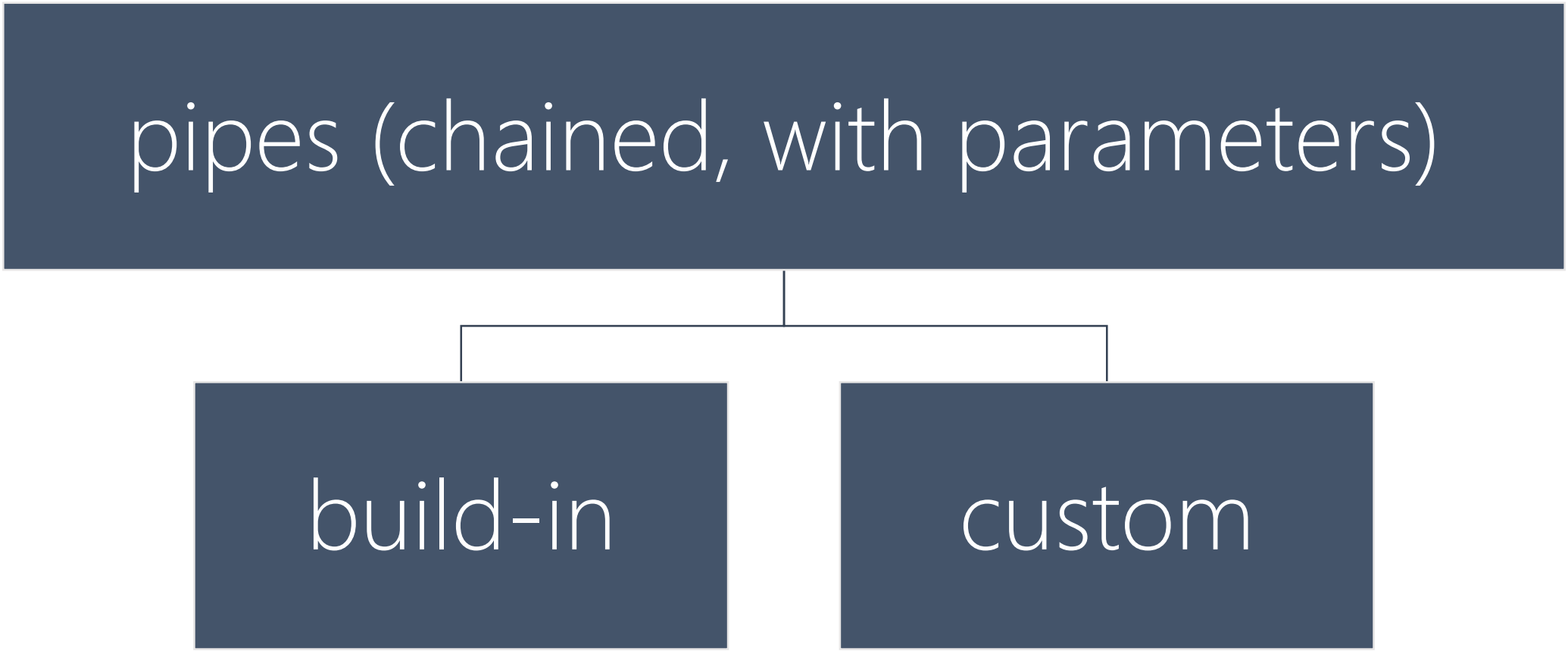
understanding pipe



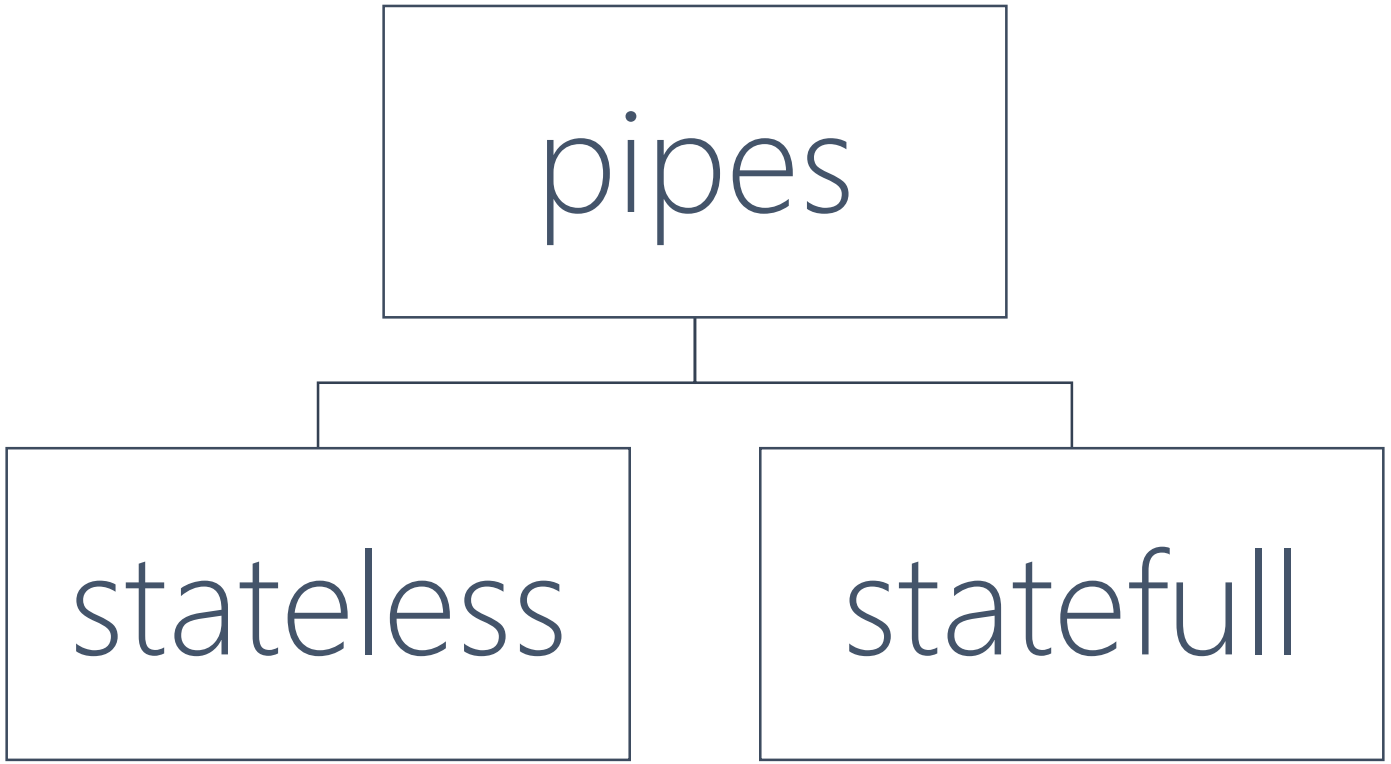
understanding pipe



types of pipes



types of pipes



examples

```
{{ 'Hello World' | uppercase }}
{{ 'hELLO wORLD' | lowercase }}
{{ 'jOHN' | titlecase }}
{{ currentDate | date }}
{{ currentDate | date:'shortDate' }}
{{ '12.123456' | number }}
{{ 150 | currency:'GBP' }}
{{ complexObject | json }}
```

…

custom pipe

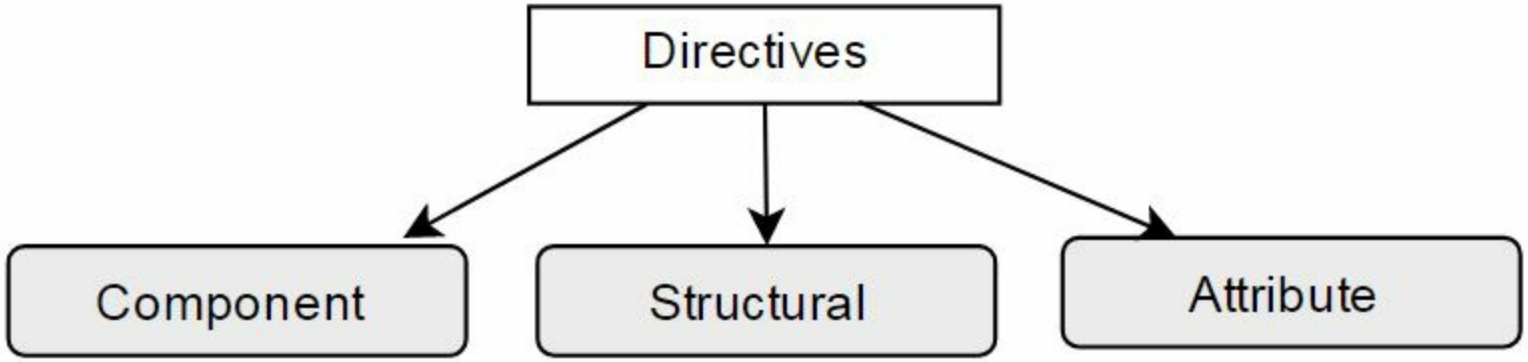
```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'fileSize’ })
export class FileSizePipe implements PipeTransform {

  transform(bytes: number = 0, decimals: number = 2): string {
    if (bytes === 0) return '0 Bytes';
    const k = 1024, dm = decimals || 2,
    sizes = ['Bytes', 'KB', 'MB', 'GB’, …];
    i = Math.floor(Math.log(bytes) / Math.log(k));
    return parseFloat((bytes / Math.pow(k, i)).toFixed(dm)) + ' ' + sizes[i];
  }
}
```

DIRECTIVE

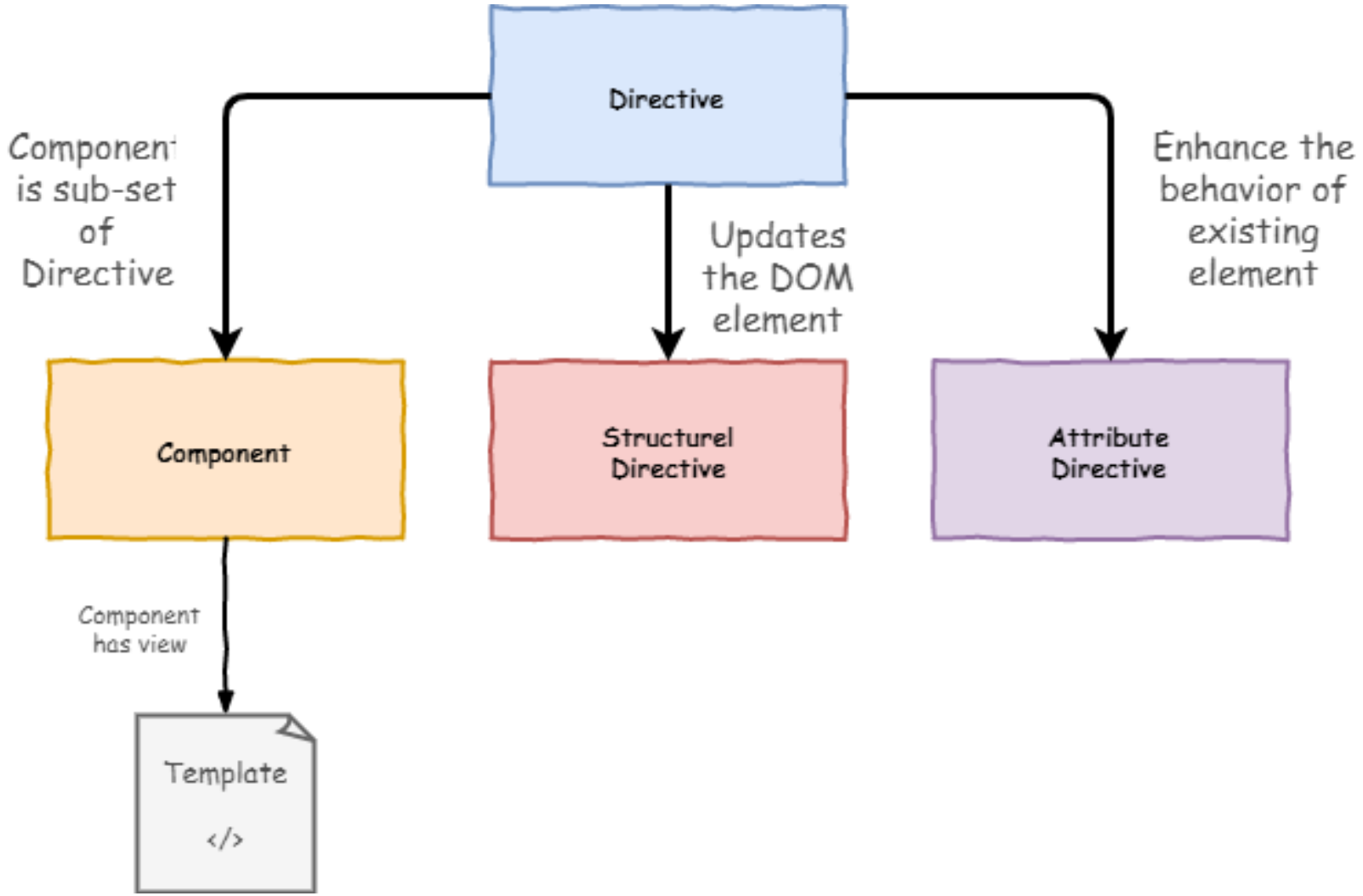
directives

- extend the behawior of the elements
- can manipulate DOM
- @Directive
- 3 types

directive types



directive types



directive types

components

- directives with a template

structural directives

- change the DOM layout by adding and removing DOM elements

attribute directives

- change the appearance or behavior of an element/component/directive

https://angular.io/guide/cheatsheet

Built-in directives

```
<section *ngIf="showSection">
```

```
<li *ngFor="let item of list">
```

```
<div [ngSwitch]="conditionExpression">
<ng-template [ngSwitchCase]="case1Exp">...</ng-template>
<ng-template ngSwitchCase="case2LiteralString">...</ng-template>
<ng-template ngSwitchDefault>...</ng-template>
</div>
```

```
<div [ngClass]="{'active': isActive, 'disabled': isDisabled}">
```

```
<div [ngStyle]="{'property': 'value'}">
<div [ngStyle]="dynamicStyles()">
```

custom directive

ng generate directive custom-directive

custom directive

```
<div sizer="72px">
   {{title}}
</div>
```

custom directive

```
import {
      Directive, ElementRef, Renderer
} from '@angular/core';

@Directive({ selector: '[sizer]' })
export class SizerDirective {
…
  constructor(
    private element: ElementRef,
    private renderer: Renderer
  ) { }
}
```

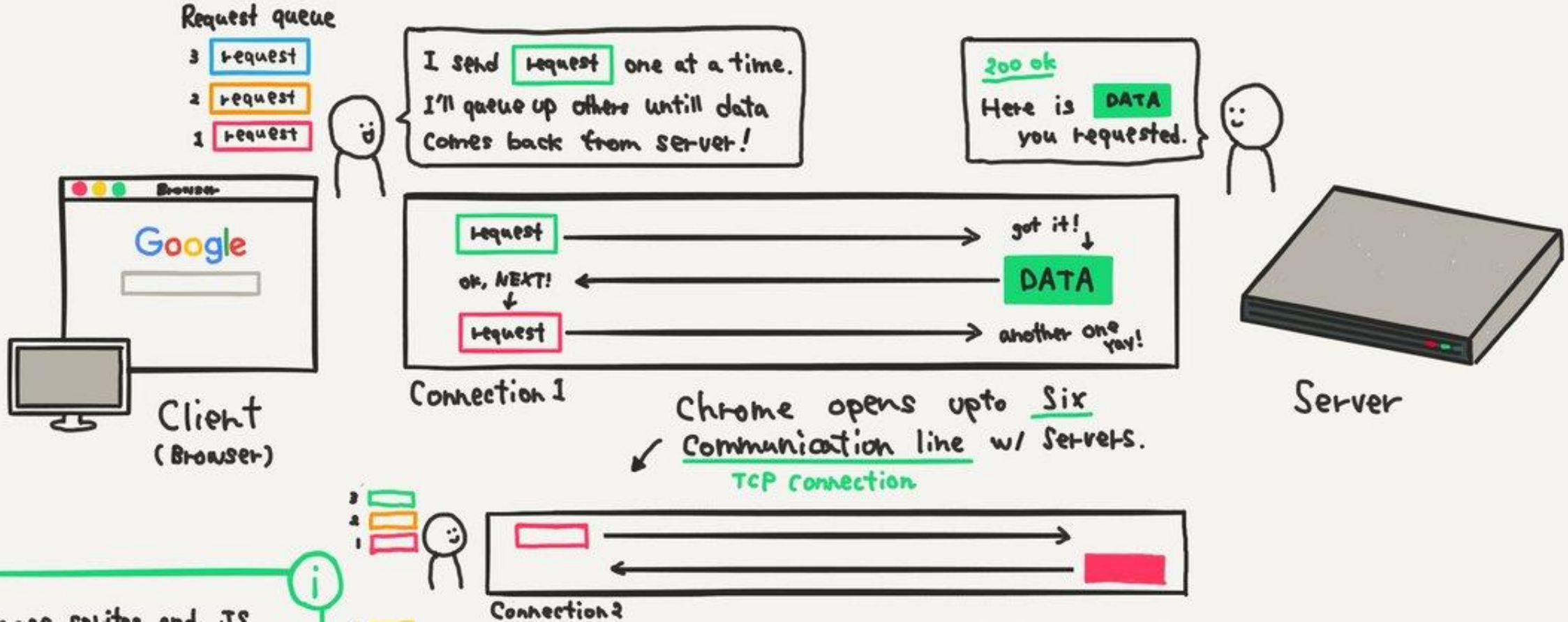
custom directive

```
@Directive({ selector: '[sizer]' })
export class SizerDirective implements OnInit {
  @Input() sizer: string;

  constructor(private element: ElementRef,private renderer: Renderer)
  { }

  ngOnInit() {
    this.renderer.setElementStyle(
     this.element.nativeElement, 'font-size', this.sizer);
  }
}
```

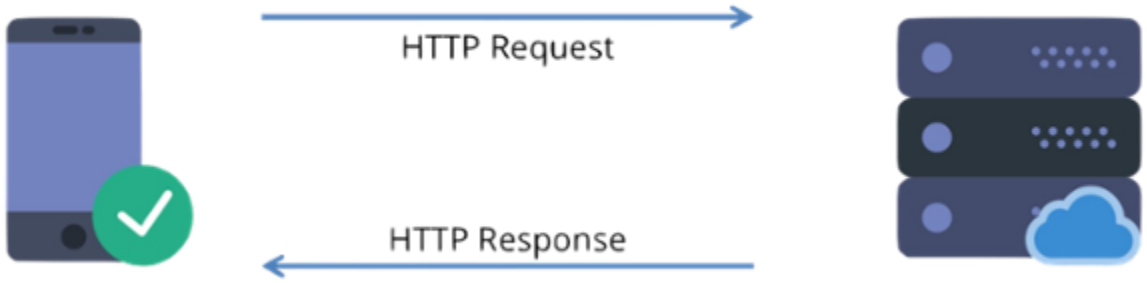
HTTP What? - by @kosamari May 3, 2017

Protocol (P in HTTP) is a communication rule between Client (browser) & Server.

HTTP 1.x



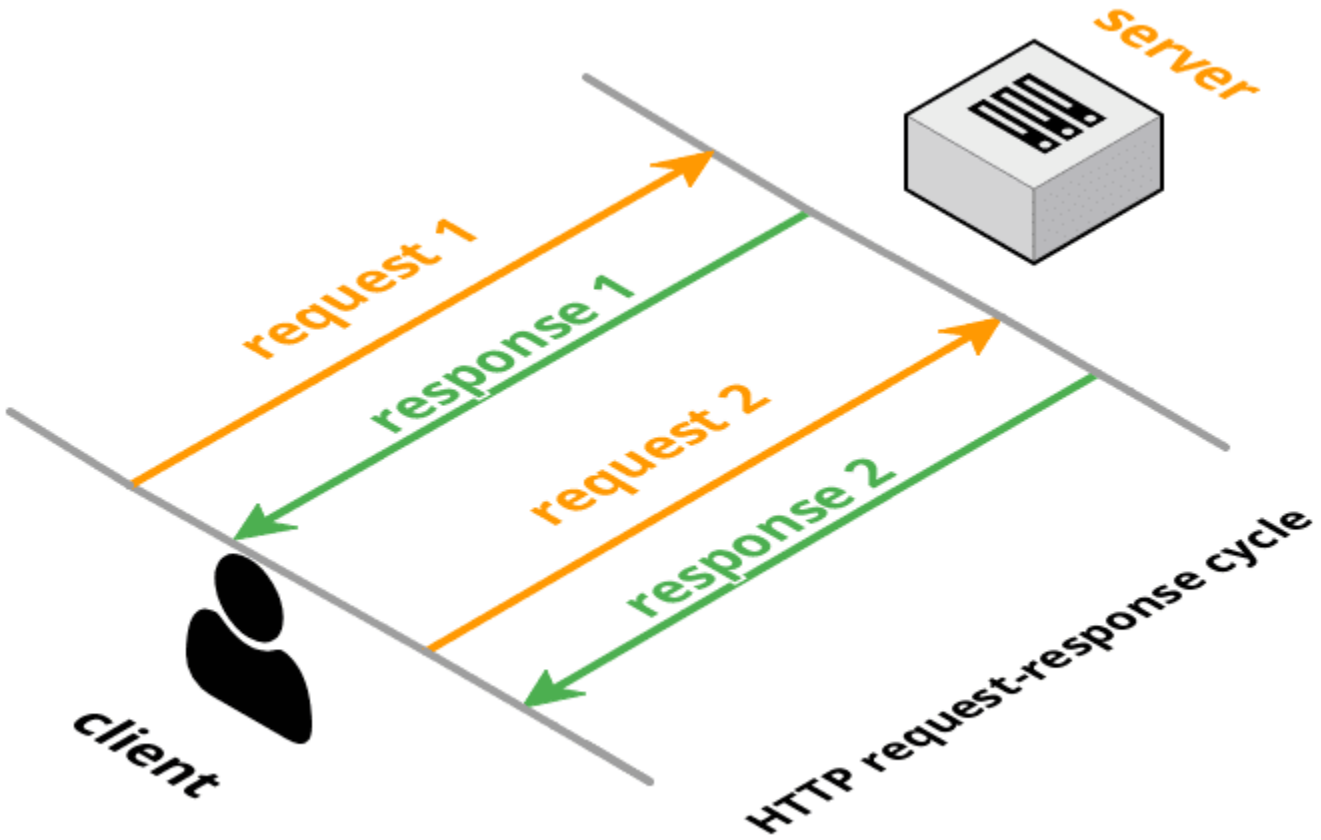
http(s) request/response



client application

API server

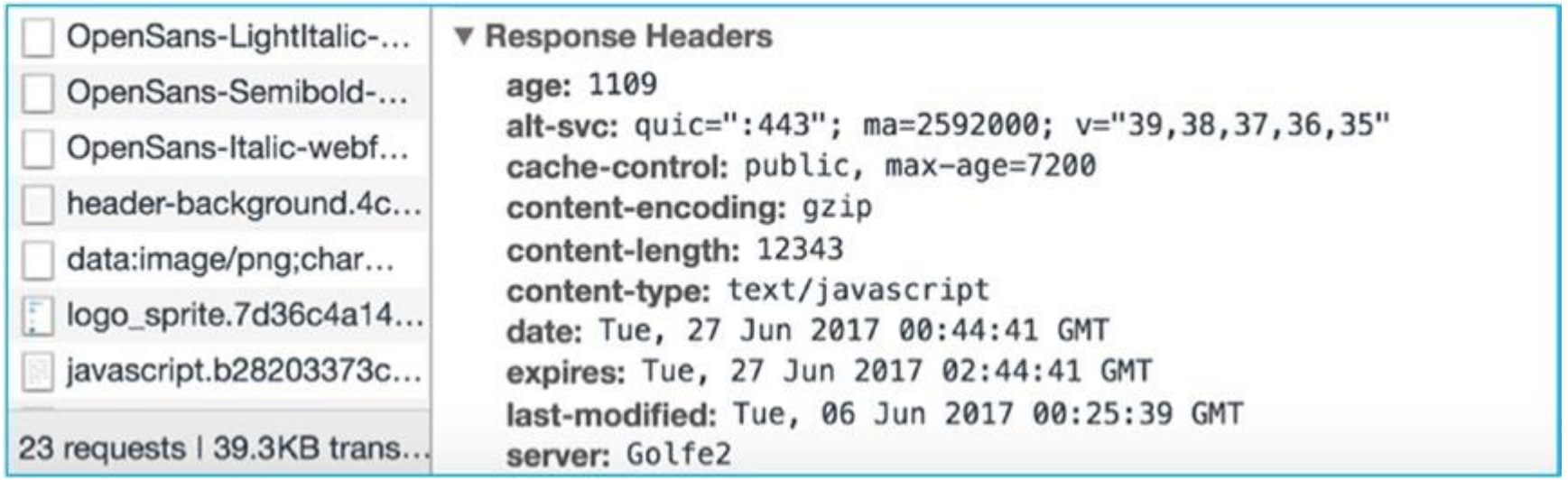
http(s) request-response cycle



HTTP methods

POST	GET	PUT	DELETE
• modify	• read	• modify	• delete

HTTP request and response headers



HTTP body

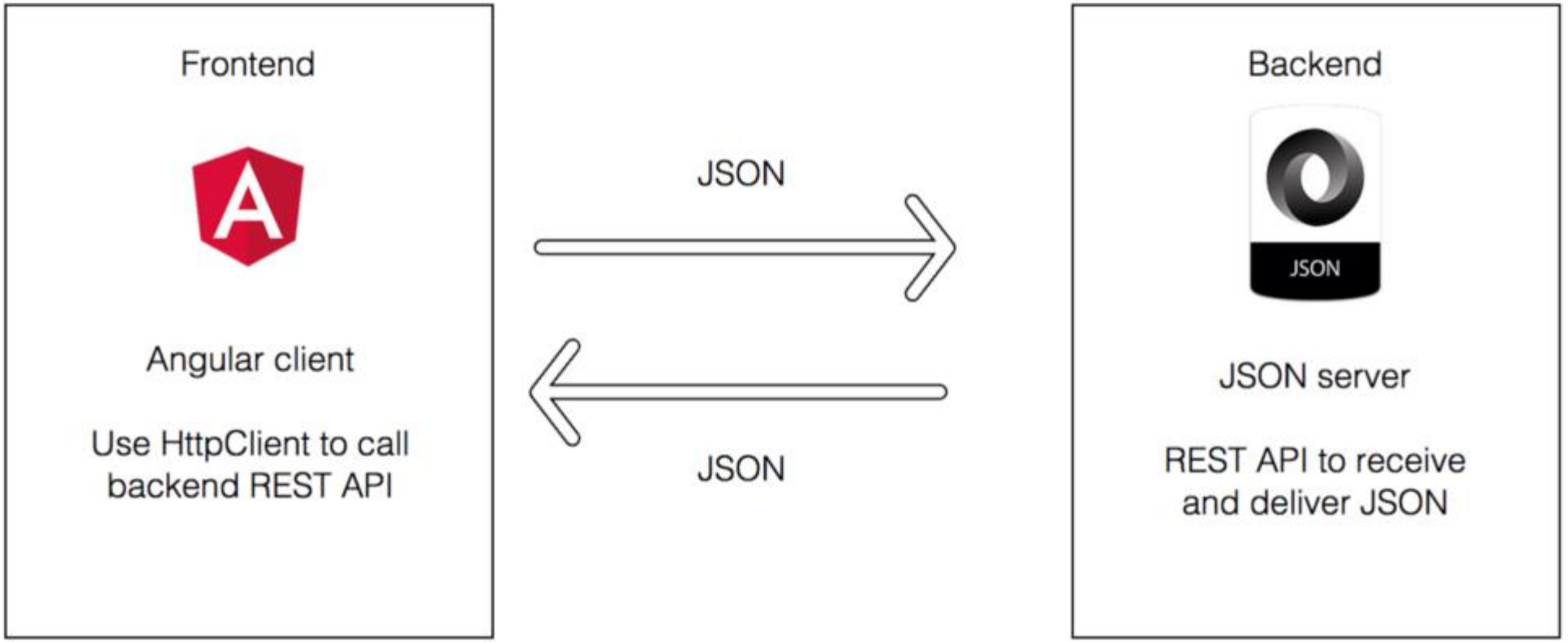
HTML

JSON

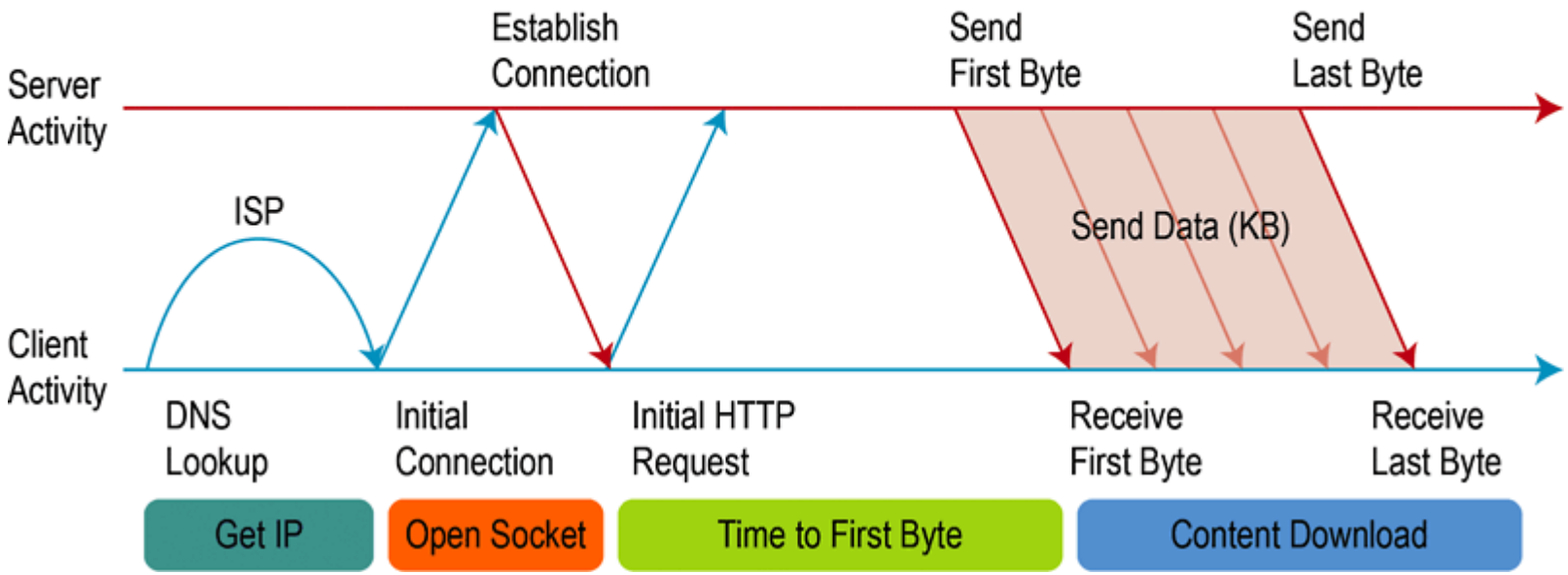
XML

Binary

HTTP communication



The HTTP Request



REST Resource Addressability

`<api_url>/`**`<resource-collection>`**`/[<id>]`

The server where the API is hosted	The resource collection to be accessed	Optional parameter to identify the resource

`http://pizzaService/`**`pizzas`**`/1`

GET	/pizzas	List all pizzas
POST	/pizzas	Create a pizza
GET	/pizzas/1	Get pizza 1
PUT	/pizzas/1	Update pizza 1
DELETE	/pizzas/1	Delete pizza 1

angular http communication

```
import {
        HttpClientModule
} from '@angular/common/http';
```

angular http communication

```
export class PizzaService {
    …
}

export class Pizza {
    …
}
```

angular http communication

```
private url = 'http://pizzaService';
private endpoint = 'pizzas';

constructor(
  protected httpClient: HttpClient
) { }
```

angular http communication

```
list(): Observable<Pizza[]> {
  return
    this.httpClient.get<Pizza[]>(
      `${this.url}/${this.endpoint}`
    );
}
```

angular http communication

```
read(id:number): Observable<Pizza> {
 return
    this.httpClient.get<Pizza>(
    `${this.url}/${this.endpoint}/${id}`
    );
}
```

angular http communication

```
public create(pizza:Pizza): Observable<Pizza>{
   return
      this.httpClient.post<Pizza>(
         `${this.url}/${this.endpoint}`, pizza
      );
}
```

https://github.com/Banndzior

#slack

kamil.mijacz@gmail.com

kamil.mijacz@softwarehut.com