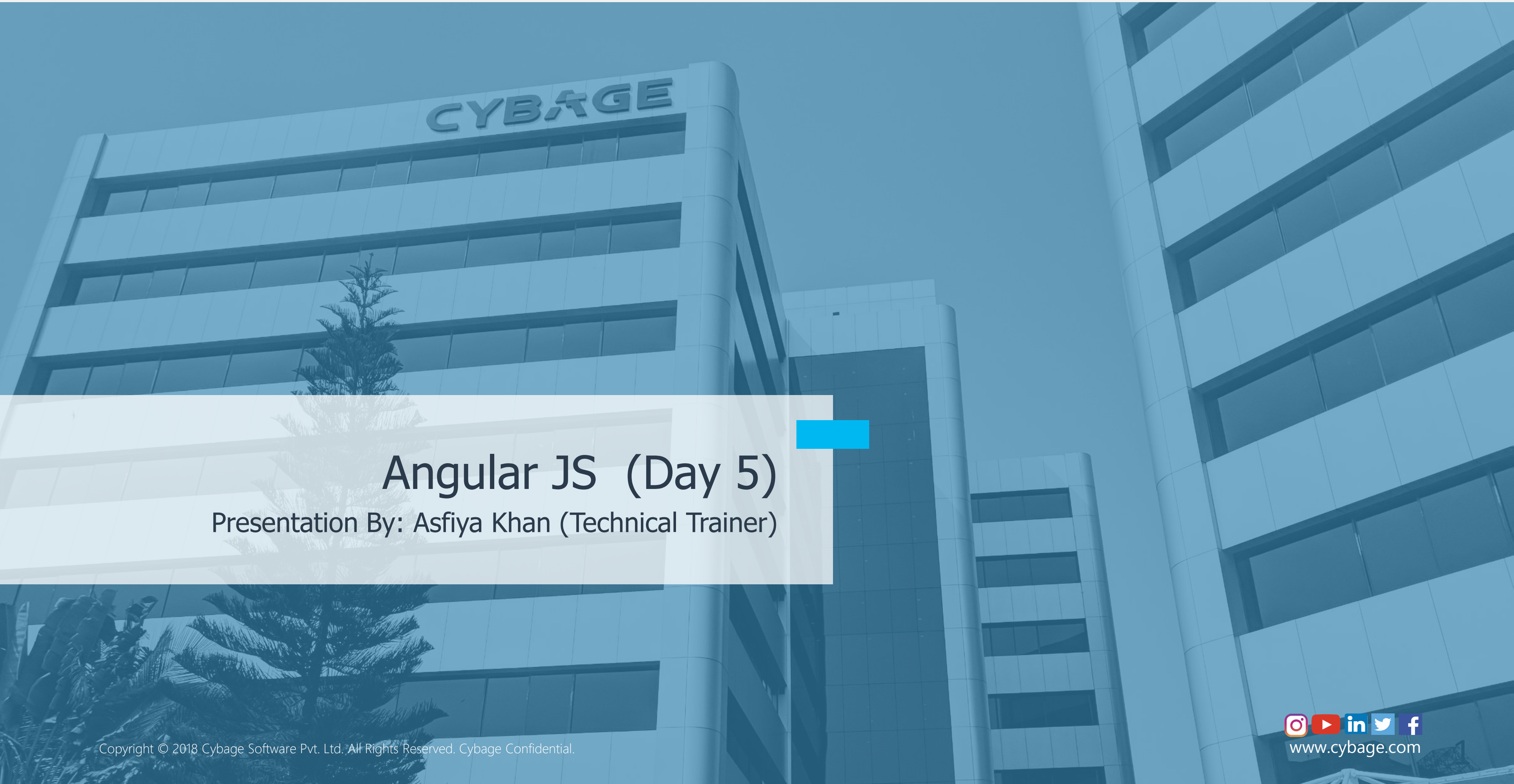


Angular JS (Day 5)

Presentation By: Asfiya Khan (Technical Trainer)

www.cybage.com

Document History

Version No.	Authored/ Modified by	Remarks/ Change History	Date <dd- mon-yy >
1.0	Asfiya Khan	First version of Angular 4	13 March 2018

Course Structure

Target audience	Trainee,SE,SSE
Level	1,2,3
Pre-requisites	Javascript,TypeScript,HTML,CSS
Training methods	Presentation , Demos, Hands-on
Evaluation	Multiple Choice Question

CYBAGE

Agenda

- Architecture and Components
- Data Binding and Pipes
- Routing and Navigation
- Templates ,Interpolation and Directives
- Angular Modules
- Services and Dependency Injection
- Ng-Forms
- Retrieving data using HTTP

Forms Module(NgForm)

There are two different types of forms:

- template-driven forms
- model-driven or reactive forms

Both the technologies belong to the @angular/forms library and are based on the same form control classes.

Template Driven Forms

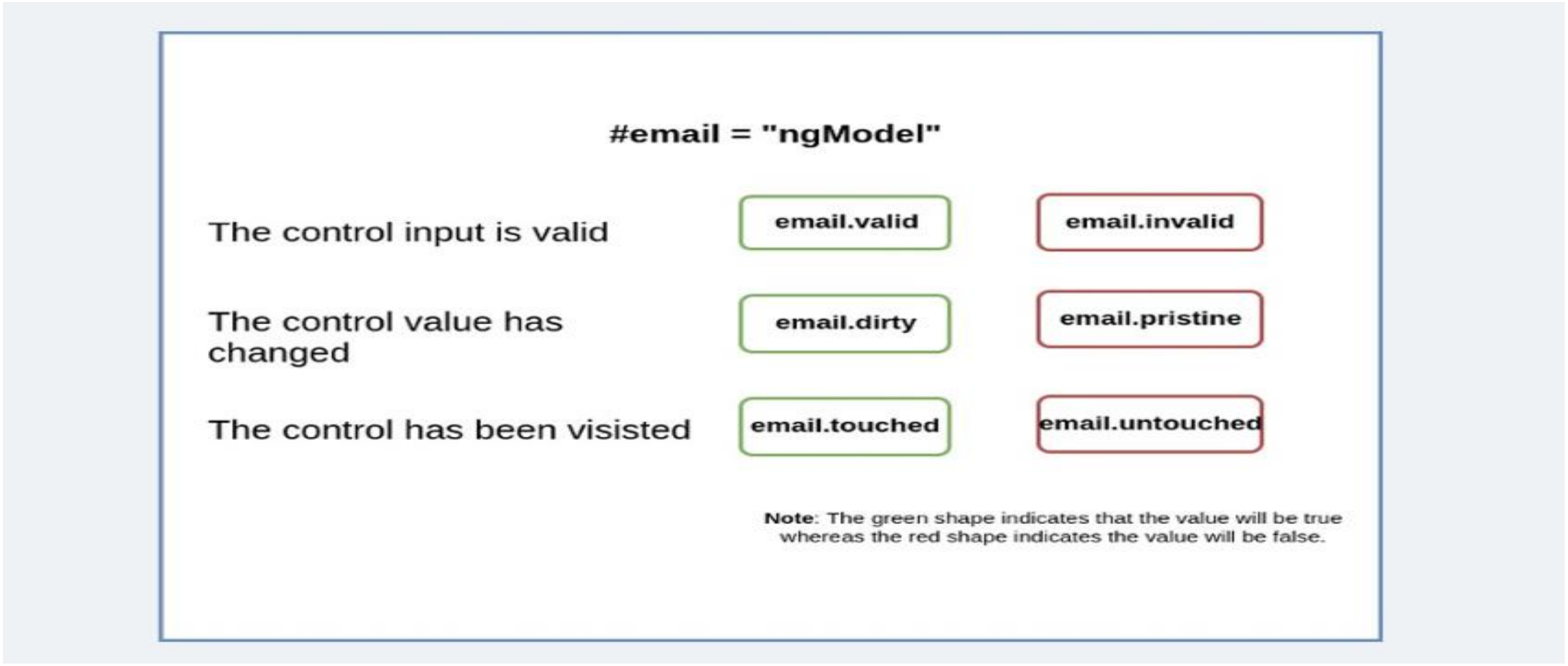
"Directives allow you to attach behavior to elements in the DOM."

- Angular provides form-specific directives that you can use to bind the form input data and the model.

- The form-specific directives add extra functionality and behavior to a plain HTML form.

- The end result is that the template takes care of binding values with the model and form validation.

Forms Module



Note: The green shape indicates that the value will be true whereas the red shape indicates the value will be false.

Reactive Forms

- In this approach we create and initialize the *form control objects* in our component class.

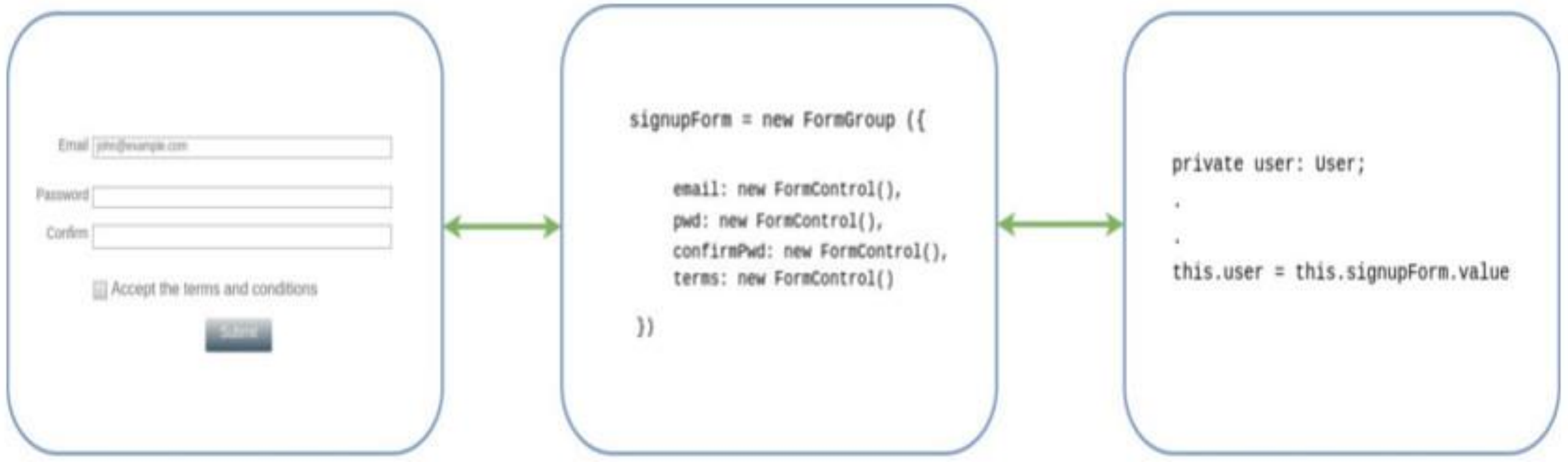
- They are intermediate objects that hold the state of the form.

- We will then bind them to the *form control elements* in the template.

Reactive Forms

- The form control object listens to any change in the input control values, and they are immediately reflected in the object's state.
- Since the component has direct access to the data model structure, all changes can be synchronized between the data model, the form control object, and the input control values.

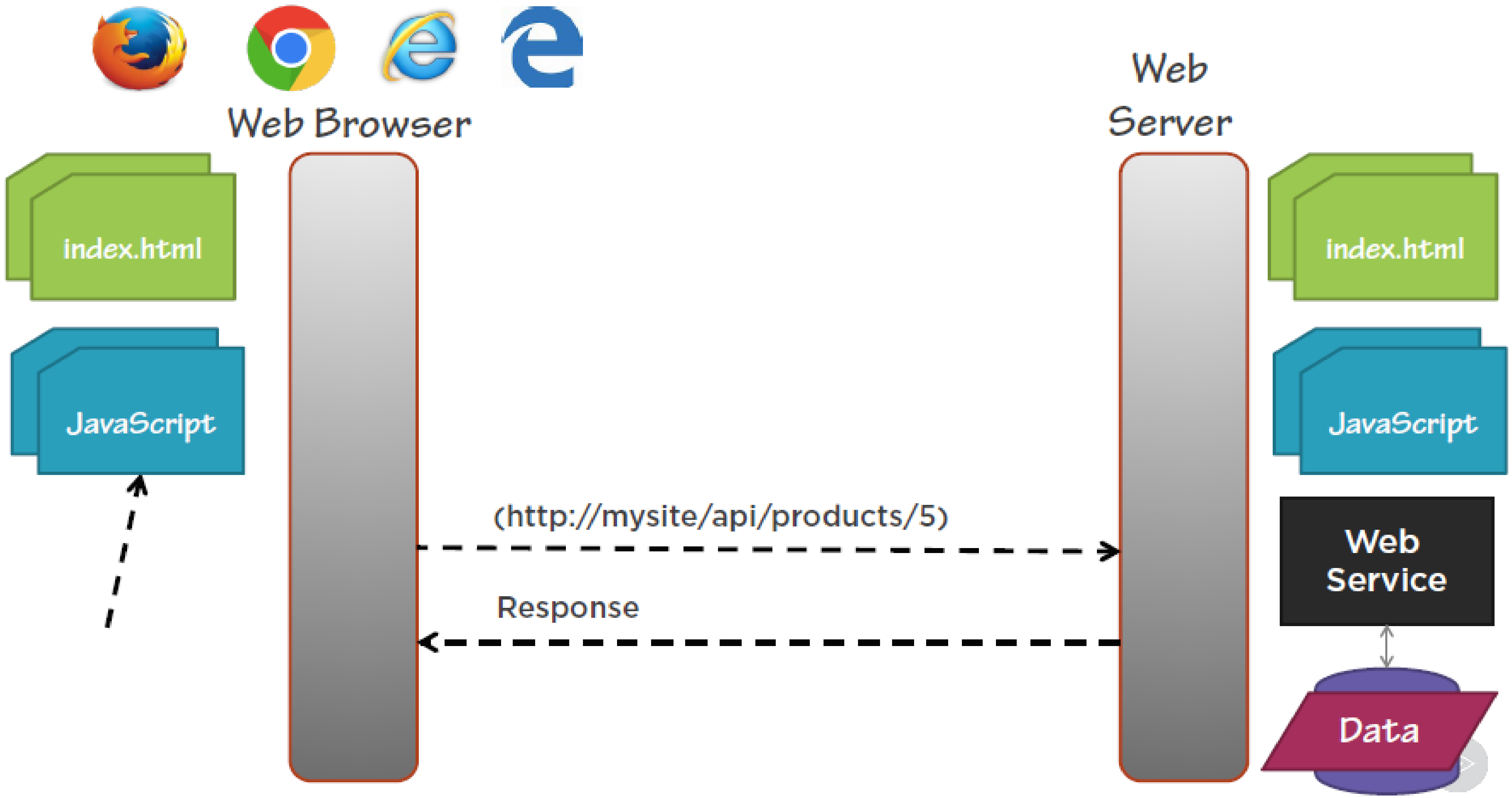
Reactive Forms



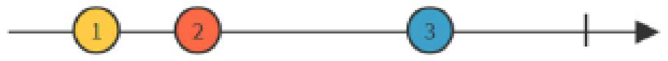
```
signupForm = new FormGroup ({

    email: new FormControl(),
    pwd: new FormControl(),
    confirmPwd: new FormControl(),
    terms: new FormControl()

})
```

```
private user: User;
.
.
this.user = this.signupForm.value
```

Form Control elements in Template **Form Model** **User Data Model**


Web Browser
Web Server
index.html
JavaScript
index.html
JavaScript
(http://mysite/api/products/5)
Response
Web Service
Data

Observables and Reactive Extensions



- Help manage asynchronous data
- Treat events as a collection
  - An array whose items arrive asynchronously over time
- Are a proposed feature for ES 2016
- Use Reactive Extensions (RxJS)
- Are used within Angular

CYBAGE

Observable Operators

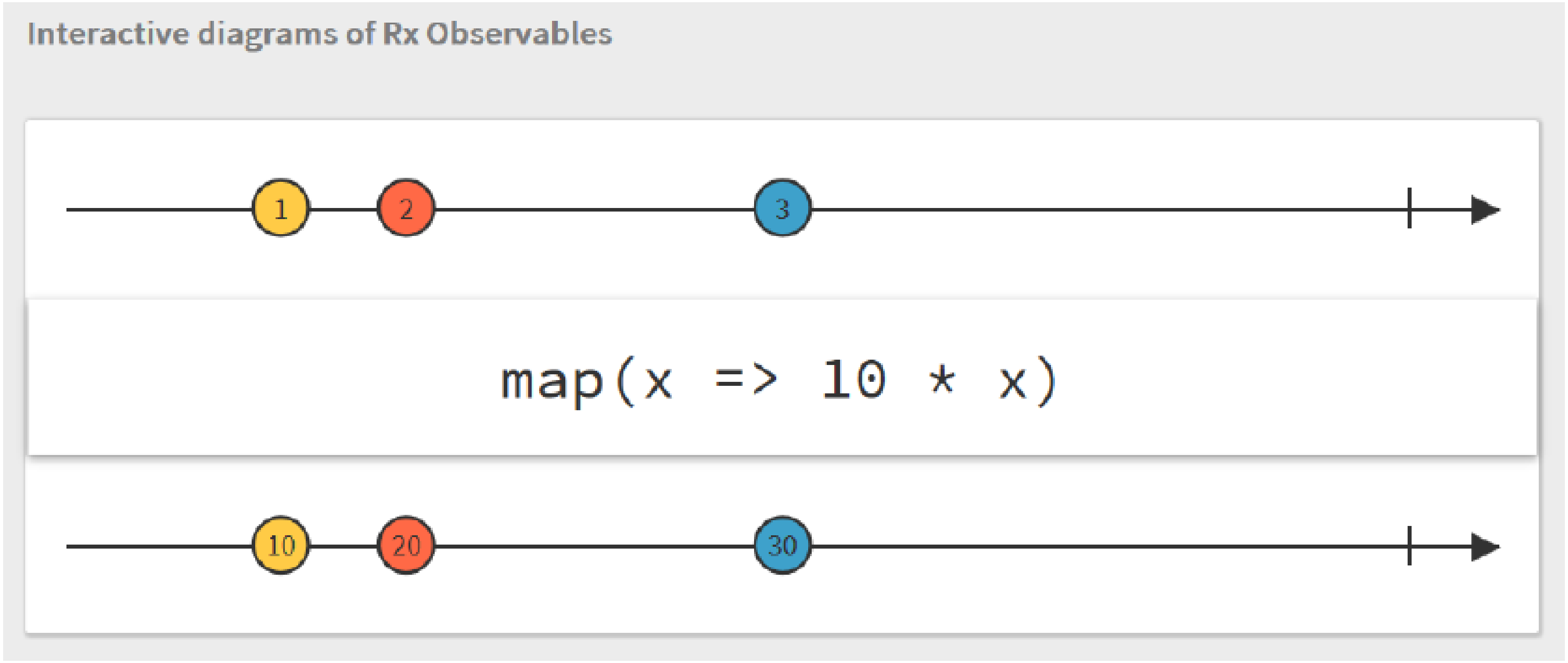
Methods on observables that compose new observables

Transform the source observable in some way

Process each value as it is emitted

Examples: map, filter, take, merge, …

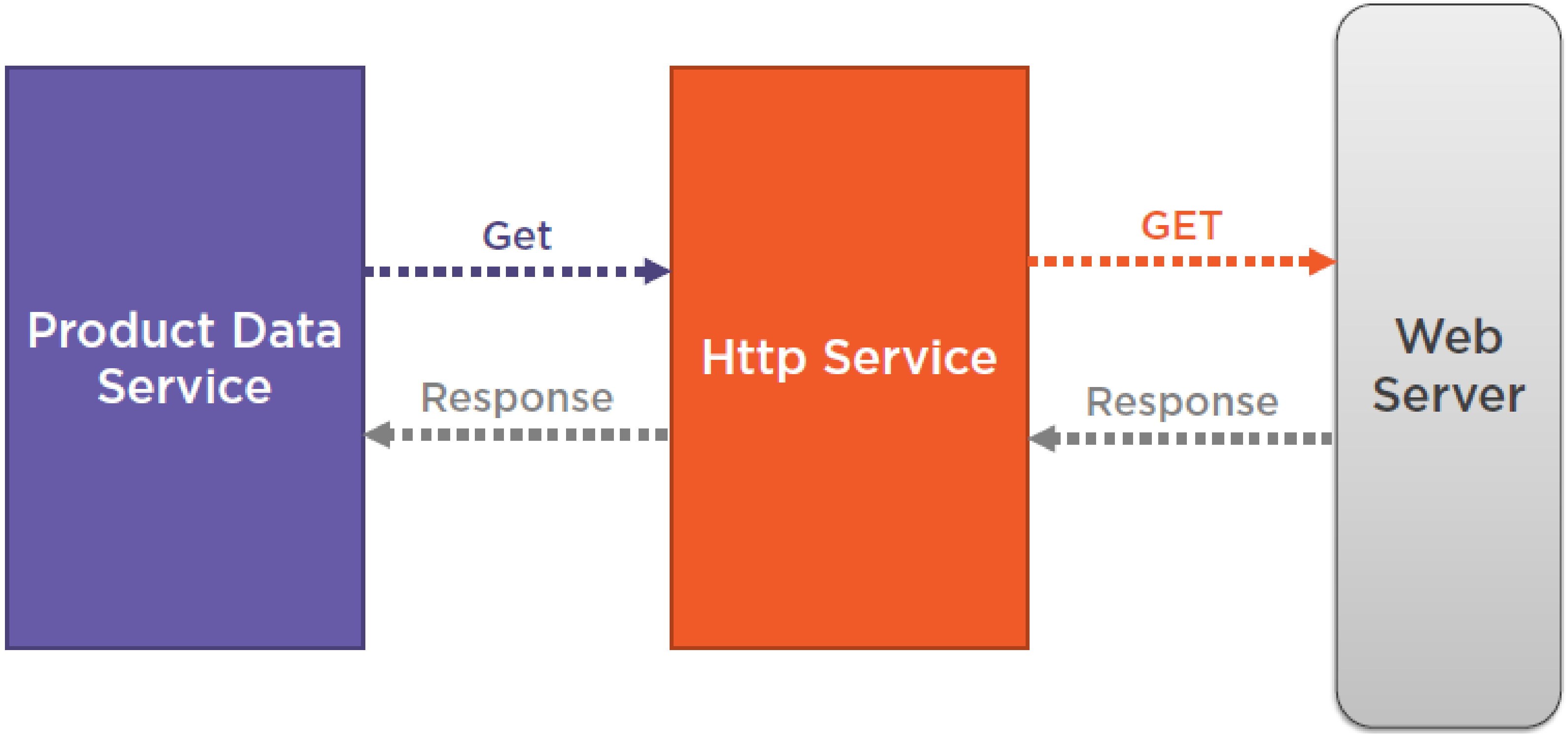
Observables



Promise vs Observable

Promise	Observable
Provides a single future value	Emits multiple values over time
Not lazy	Lazy
Not cancellable	Cancellable
	Supports map, filter, reduce and similar operators

Sending an Http Request



Sending an Http Request

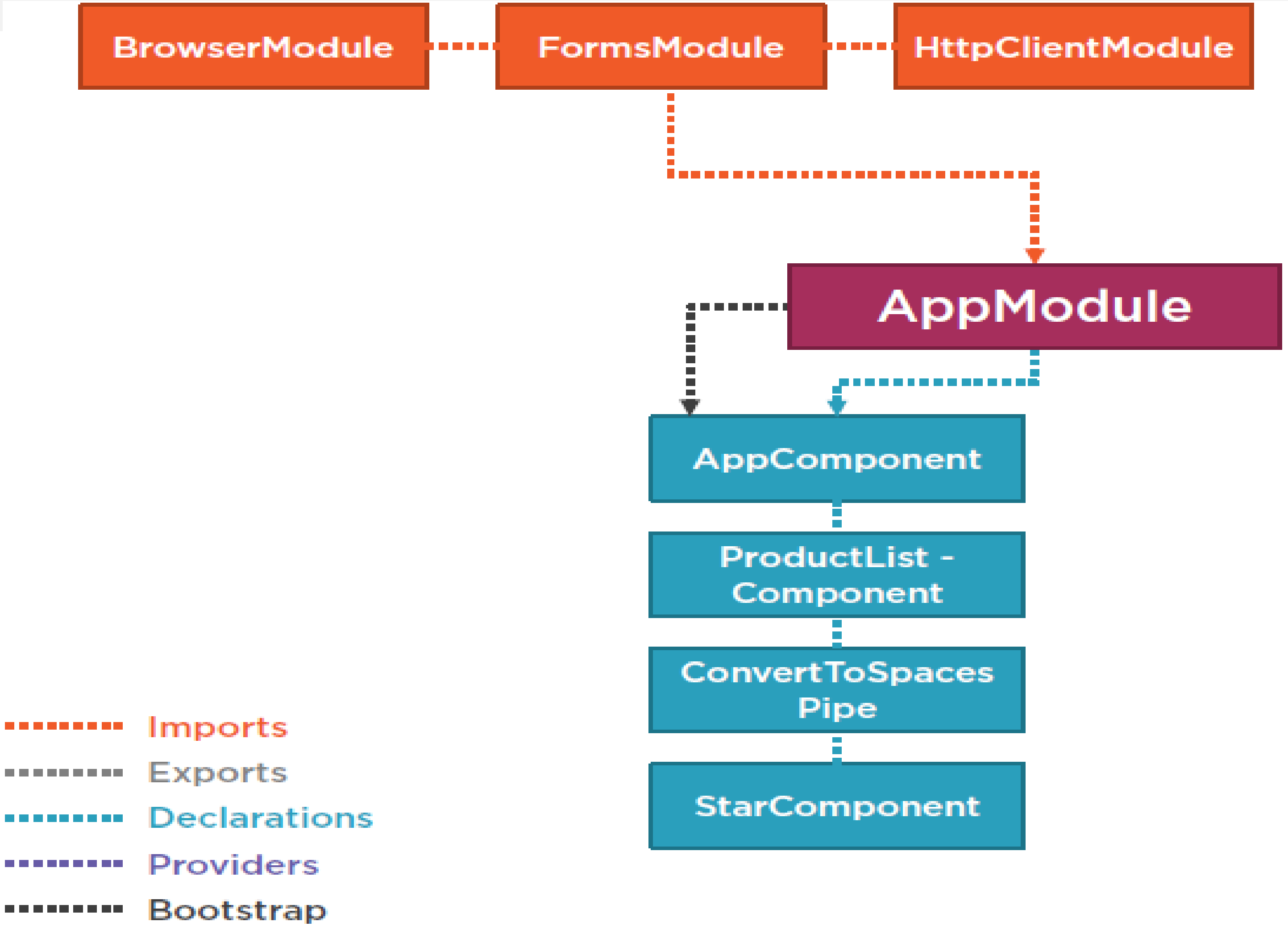
product.service.ts

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



@Injectable()
export class ProductService {
  private _productUrl = 'www.myWebService.com/api/products';

  constructor(private _http: HttpClient) { }

  getProducts() {
   return this._http.get(this._productUrl);

  }
}
```

Registering the Http Service Provider

app.module.ts

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

@NgModule({
  imports: [
      BrowserModule,
      FormsModule,
      HttpClientModule ],
  declarations: [
      AppComponent,
      ProductListComponent,
      ConvertToSpacesPipe,
      StarComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```


BrowserModule
FormsModule
HttpClientModule
AppModule
AppComponent
ProductList - Component
ConvertToSpaces Pipe
StarComponent
Imports
Exports
Declarations
Providers
Bootstrap

Sending an Http Request

product.service.ts

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



@Injectable()
export class ProductService {
  private _productUrl = 'www.myWebService.com/api/products';

  constructor(private _http: HttpClient) { }

  getProducts() {
   return this._http.get(this._productUrl);

  }
}
```

Sending an Http Request

product.service.ts

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



@Injectable()
export class ProductService {
  private _productUrl = 'www.myWebService.com/api/products';

  constructor(private _http: HttpClient) { }

  getProducts() {
   return this._http.get<IProduct[]>(this._productUrl);

  }
}
```

Sending an Http Request

product.service.ts

```
...
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';


@Injectable()
export class ProductService {
  private _productUrl = 'www.myWebService.com/api/products';

  constructor(private _http: HttpClient) { }

  getProducts(): Observable<IProduct[]> {
   return this._http.get<IProduct[]>(this._productUrl);

  }
}
```

Exception Handling

product.service.ts

```
...
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/catch';
import 'rxjs/add/operator/do';
...

  getProducts(): Observable<IProduct[]> {
   return this._http.get<IProduct[]>(this._productUrl)
              .do(data => console.log('All: ' + JSON.stringify(data)))
              .catch(this.handleError);
  }

  private handleError(err: HttpErrorResponse) {
  }
```

Subscribing to an Observable

```
x.then(valueFn, errorFn)                          //Promise
x.subscribe(valueFn, errorFn)                     //Observable
x.subscribe(valueFn, errorFn, completeFn)   //Observable
let sub = x.subscribe(valueFn, errorFn, completeFn)
```

product-list.component.ts

```
ngOnInit(): void {
    this._productService.getProducts()
            .subscribe(products => this.products = products,
                       error => this.errorMessage = <any>error);
}
```

Any questions ?

Thank You!

www.cybage.com