



<https://angular.io/>

Angular is a platform that
makes it easy to build
web client applications *in*
HTML and TypeScript.

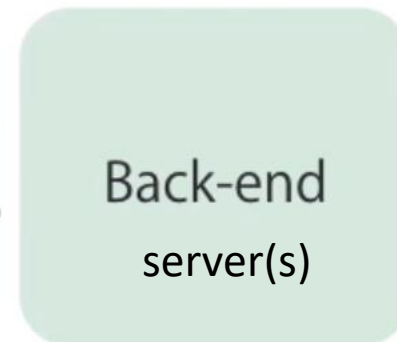
User Interface (UI)



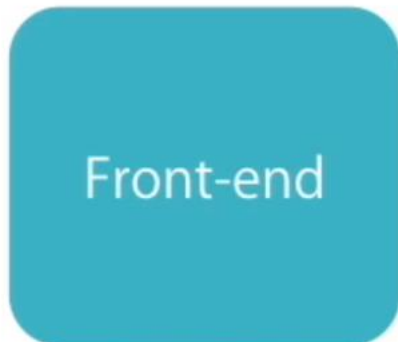
Get or save data



Data and Processing



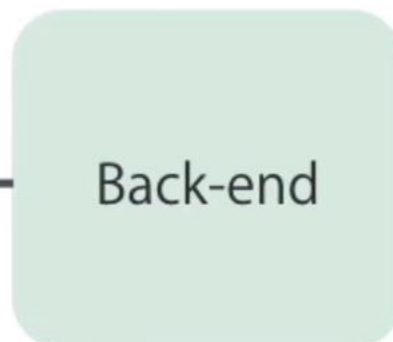
The part the user sees and interact with.
Runs in a web browser.



HTTP Request



HTTP Endpoint



HTML Templates.
Presentation logic.

- Display data
- Respond to user actions

Data + API
Business logic

1.0

AngularJS

2010

Re-written in TS

2.0

Angular 2

2016

2.0

2.1

2.2

2.3

4

5

@angular/core

2.3.0

@angular/compiler

2.3.0

@angular/http

2.3.0

@angular/router

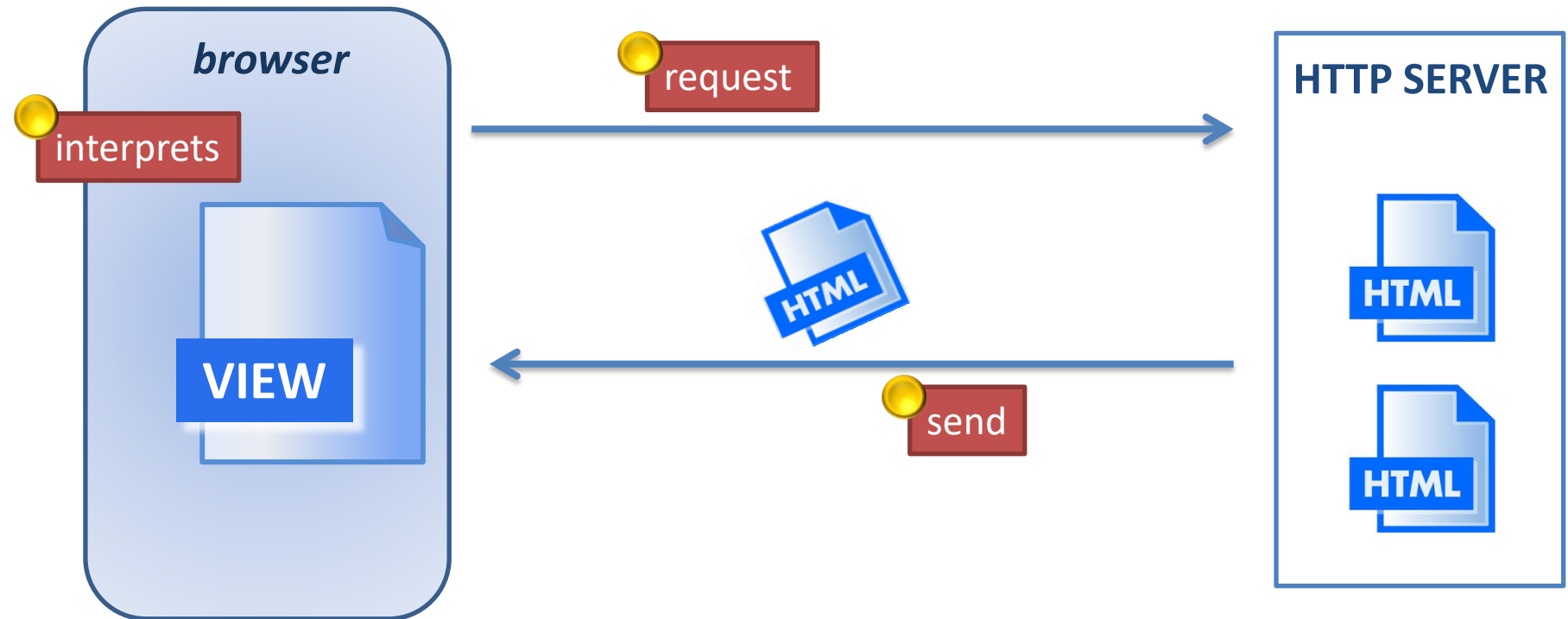
3.3.0

AngularJS (1.x)

Angular (2+)

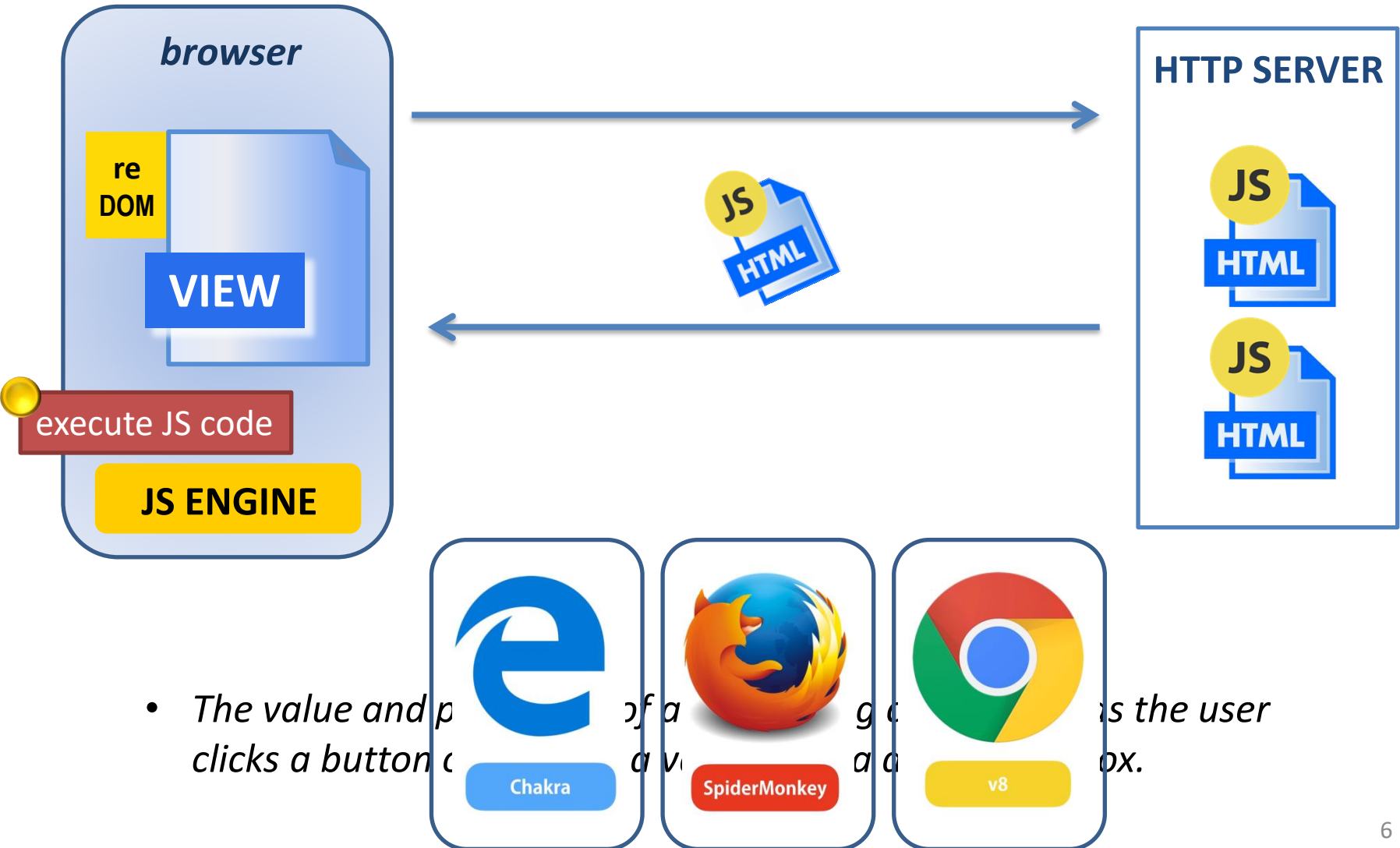
1. Introduction

Static page



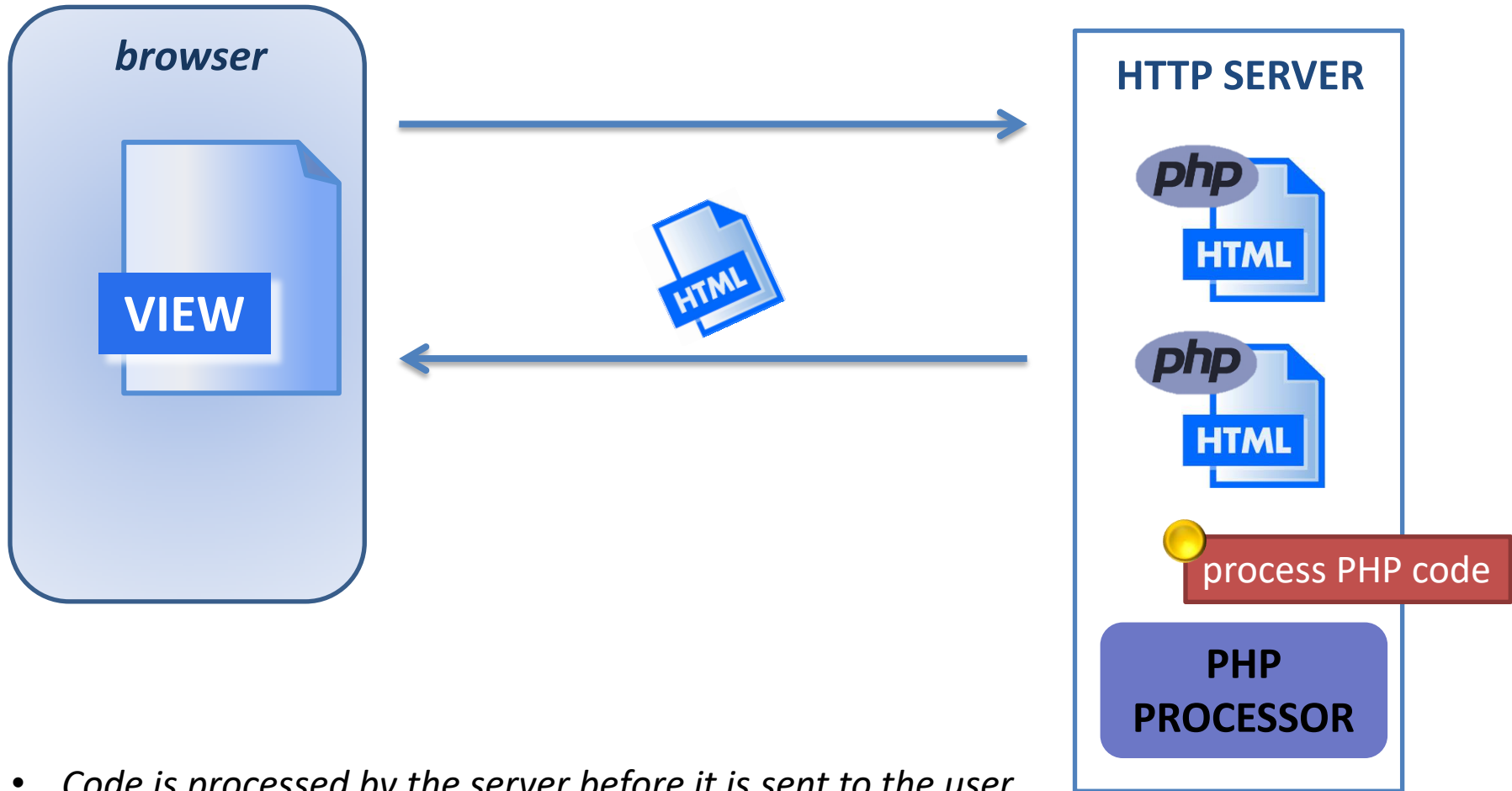
- *The view cannot change.*
- *The user cannot interact with the view.*

Dynamic page



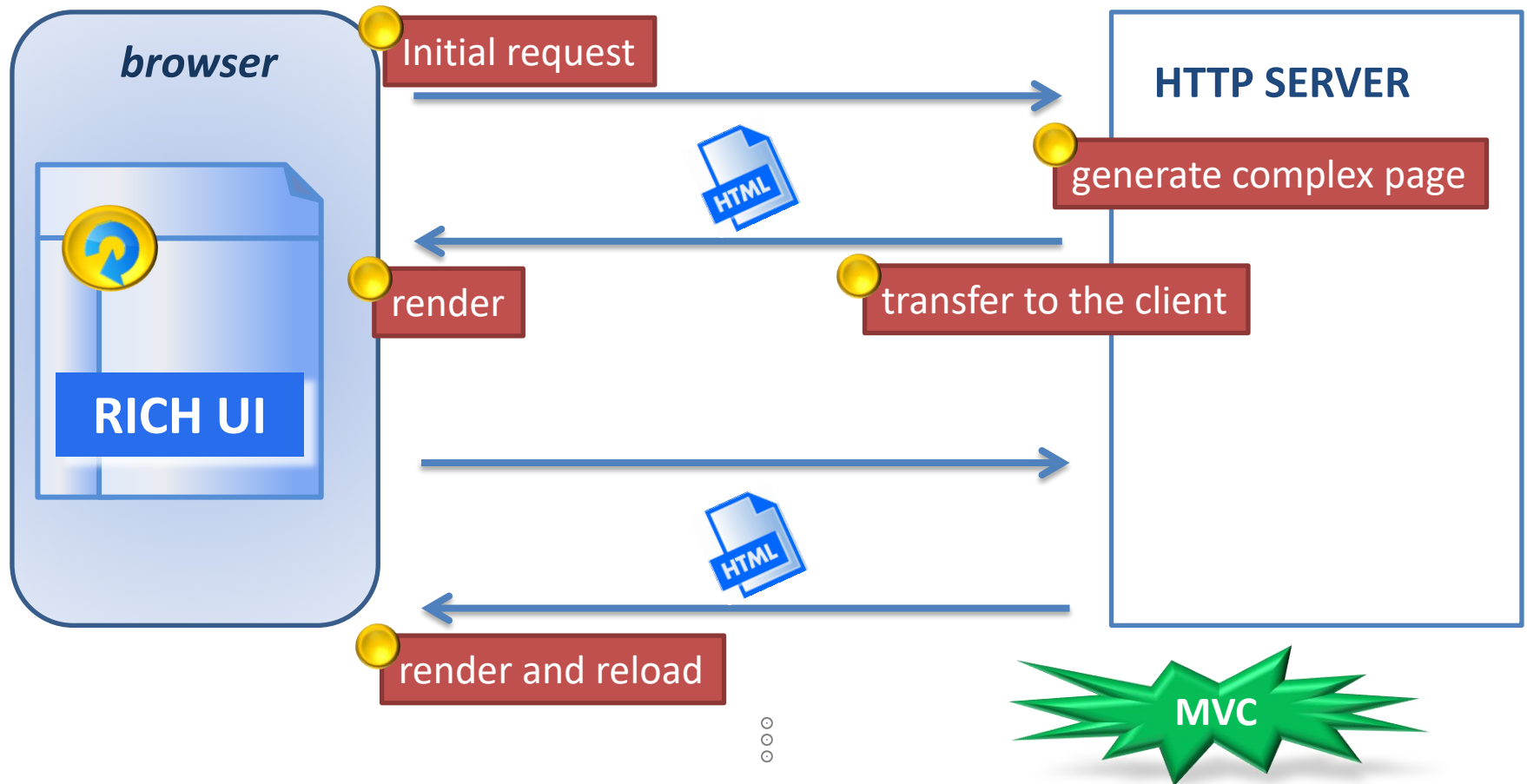
- The value and position of a button change as the user clicks a button.

Server-side render application



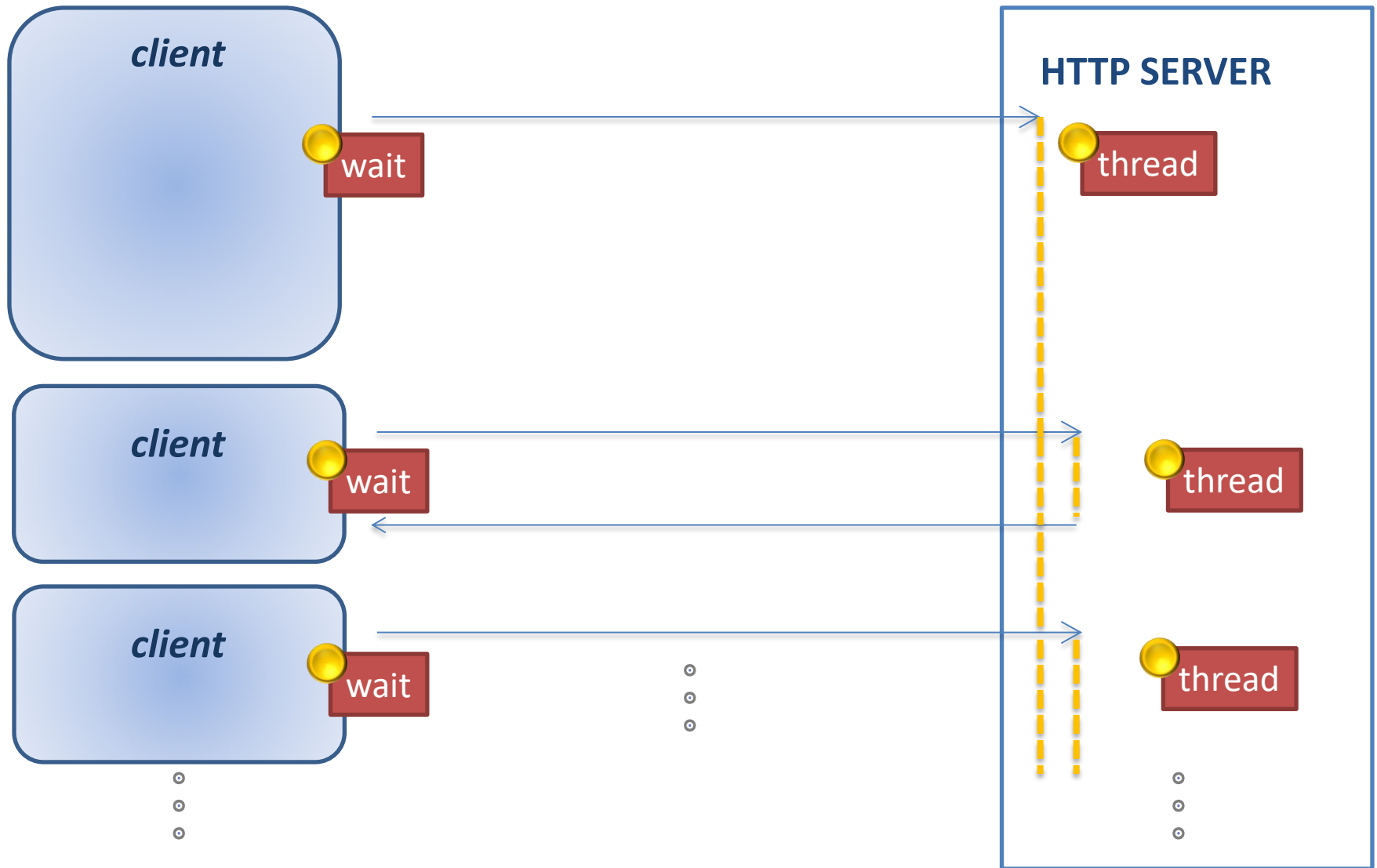
- *Code is processed by the server before it is sent to the user.*

MPA - Multi page web application

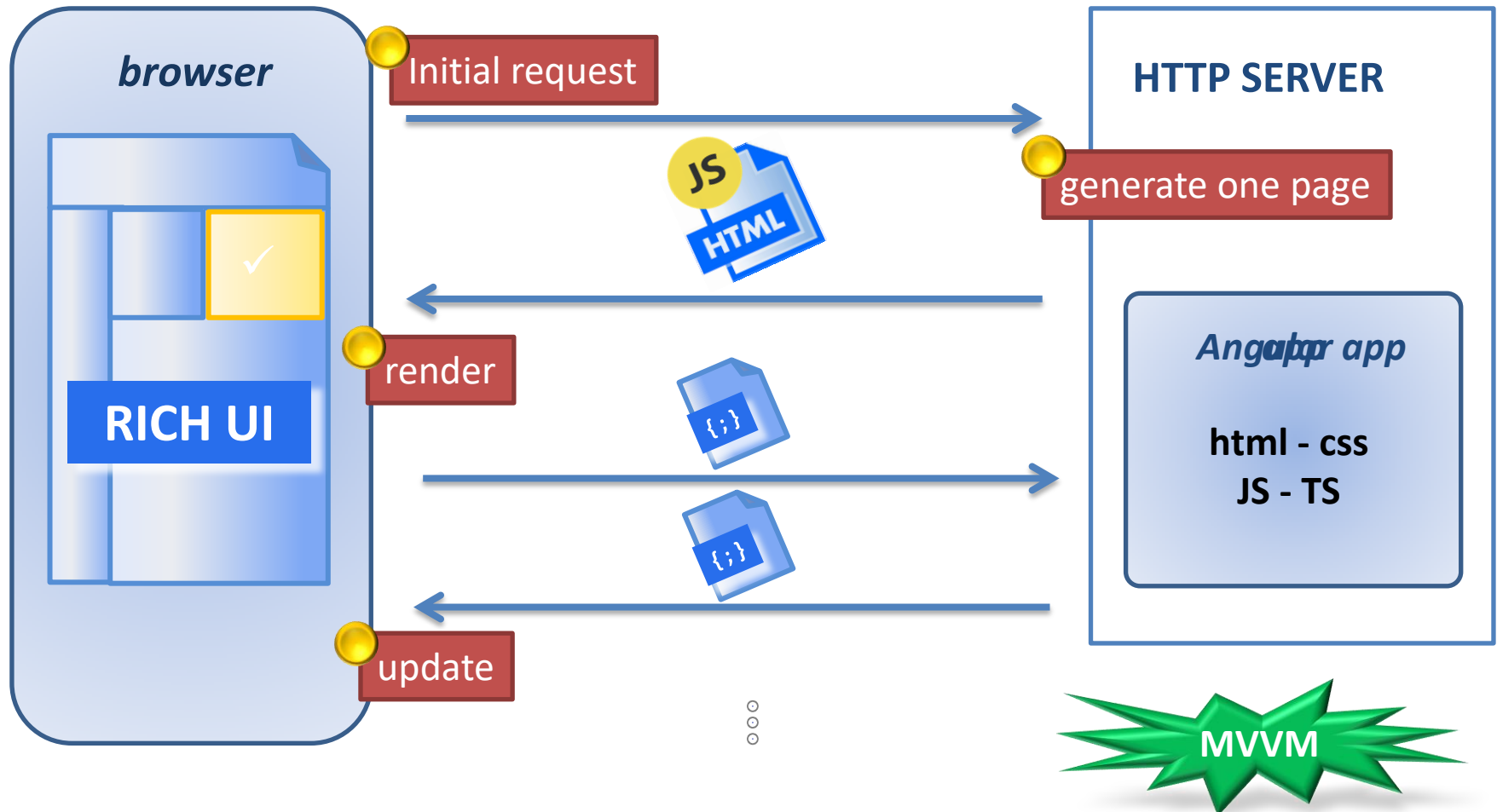


- *Every time the application needs to display data from the server it has to request a new page and then render it in the web browser.*

Blocking synchronous architecture

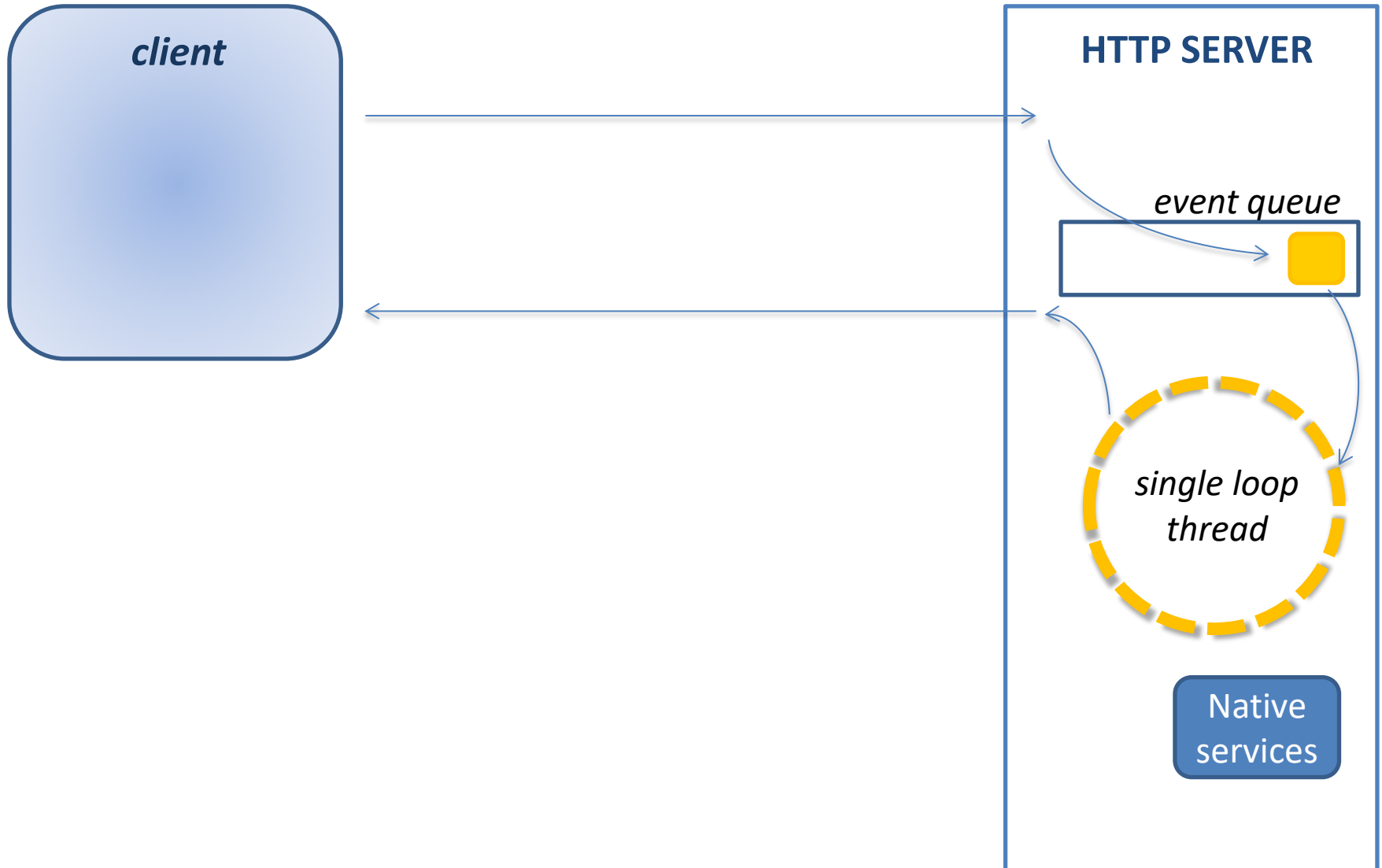


SPA - Single page web application

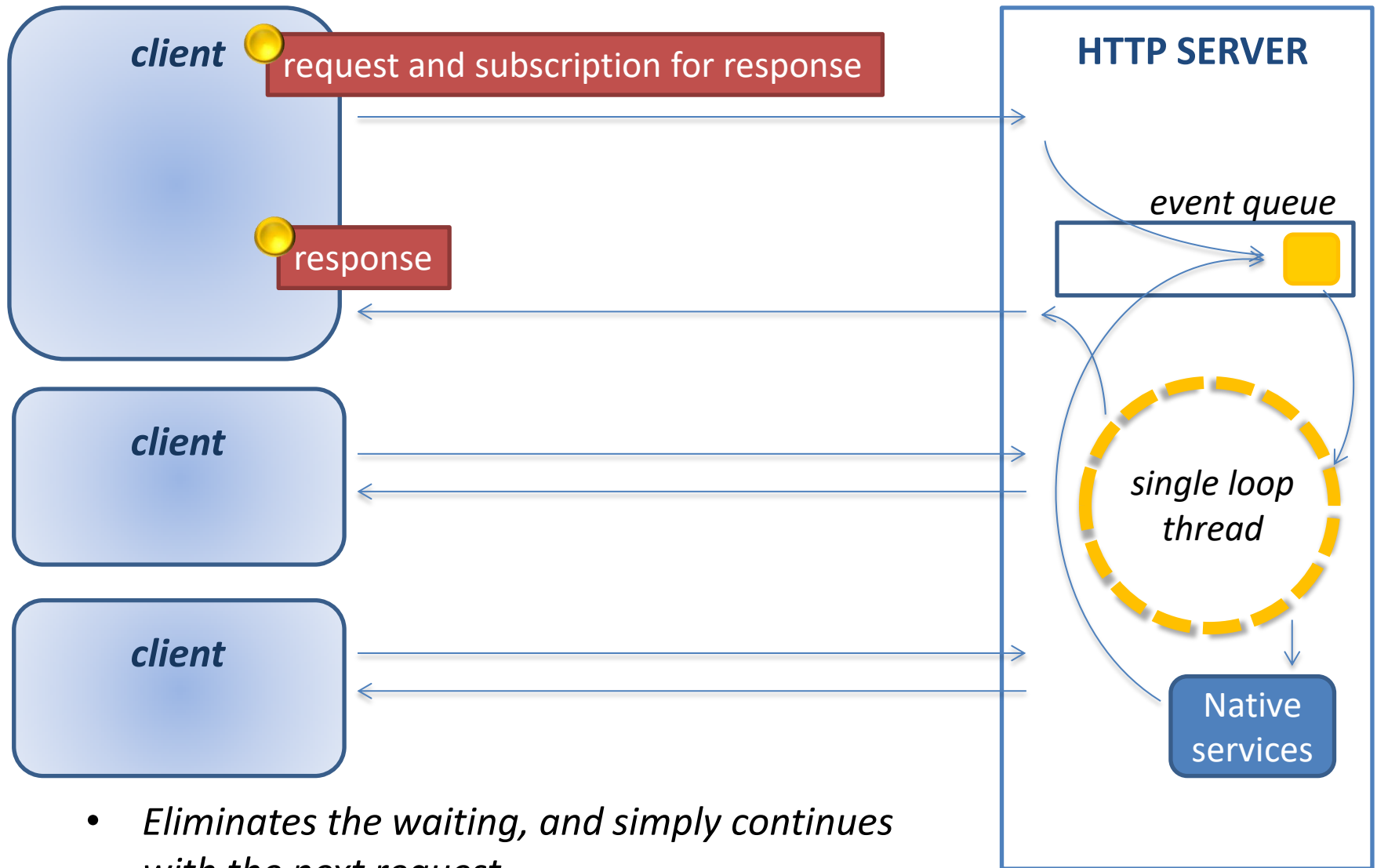


- *The first time, the only one page is generated on the server.*
- *Additional requests only send and receive update data.*

asynchronous architecture

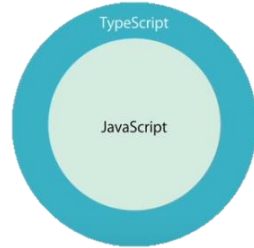


asynchronous architecture



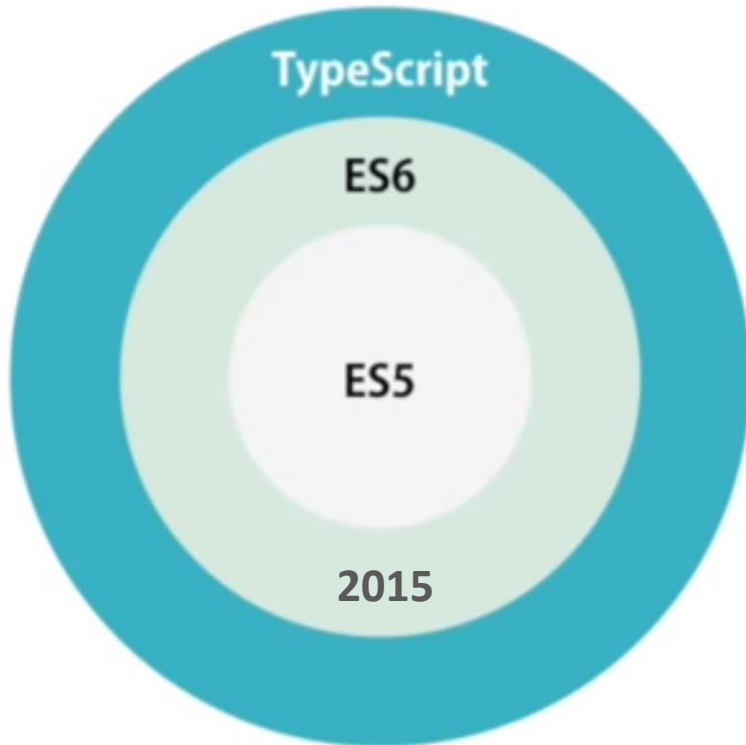
- *Eliminates the waiting, and simply continues with the next request.*

TypeScript example



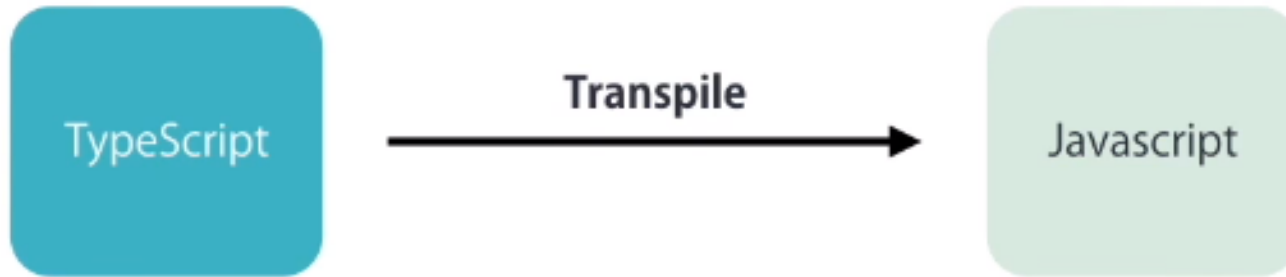
TypeScript

TypeScript is an open-source programming language developed and maintained by Microsoft.



- ❑ It is a strict syntactical superset of JavaScript, and adds optional static typing to the language (every syntax in JS is valid on TS).
- ✓ Strong typing
- ✓ Object-oriented features
- ✓ Compile-time errors
- ✓ Great tooling

tsc - TypeScript Compiler



TS hello-file.ts ✕

```
1 function consoleLog(message){  
2   |   console.log(message);  
3 }  
4 var message = 'Hola Lili';  
5 consoleLog(message);
```

JS hello-file.js ✕

```
1 function consoleLog(message) {  
2   |   console.log(message);  
3 }  
4 var message = 'Hola Lili';  
5 consoleLog(message);
```

> *tsc hello-file.ts*

2. Get Ready



- ❑ **npm** is the package manager for JavaScript, you can find libraries like bootstrap, react and **angular**.
- ❑ **npm** makes it easy for JavaScript developers to share and reuse code, and makes it easy to update the code that you're sharing.
- ❑ **npm** is distributed with **node.js** which means that when you download **node.js**, you automatically get **npm** installed on your computer.





- ❑ Before 2009, the only way to execute JS code was inside a browser.
- ❑ **node.js** was built on V8 and provides a runtime environment to execute JS code outside a browser.
- ❑ Angular does not need **node.js** directly, it is used for all the build and development tools.

e.g. there's a tool that keep watch at the files and if the file is modified then the tool automatically re run the application with new changes and re render it in the browser.




<https://nodejs.org/en/download/>


Downloads

Latest LTS Version: 8.11.1 (includes npm 5.6.0)


Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Recommended For Most Users


Windows Installer
node-v8.11.1-x64.msi


macOS Installer
node-v8.11.1.pkg

Current
Latest Features


Source Code
node-v8.11.1.tar.gz

Windows Installer (.msi)

Windows Binary (.zip)

macOS Installer (.pkg)

macOS Binary (.tar.gz)

Linux Binaries (x86/x64)

Linux Binaries (ARM)

Source Code

32-bit		64-bit	
32-bit		64-bit	
64-bit			
64-bit			
32-bit		64-bit	
ARMv6	ARMv7		ARMv8
node-v8.11.1.tar.gz			

Verifying the installation

cmd

```
> node -v
```

```
V8.11.1
```

```
> npm -v
```

```
5.6.0
```



Visual Studio Code

<https://code.visualstudio.com>

Code editing.
Redefined.

Free. Open source. Runs everywhere.

Download for Windows

Stable Build





A command line interface for Angular

cmd

```
> npm install -g @angular/cli
```

CLI

```
> ng new  
> ng generate  
> ng serve
```

- ✓ *Creation of the application.*
- ✓ *Serve the application.*
- ✓ *Open the application.*

test-app

localhost:4200

CLI

```
> ng new test-app  
> cd test-app  
> ng serve --open
```

- *If you create an angular app, and you run on your computer, then your computer now works as a server.*
- *If anyone tries to access your computer on the right port (4200), they will get to your application.*

EJERCICIO:

- Realizar la aplicación test-app.

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/test-app.pdf

Tiempo disponible: 10 mins.

Compilation

An angular application consists largely of components and their HTML templates. Before the browser can render the application, the components and templates must be converted to executable JavaScript by an *Angular compiler*.



```
PS C:\angularProjects\hello-app> ng serve -o
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200
12% building modules 23/24 modules 1 active ...node_modules\style-loader\lib\urls.jswebpack: wait
2018-05-01T17:03:08.058Z
Hash: e8f186c030cfb13d4fdd
Time: 8513ms
chunk {inline} inline.bundle.js (inline) 3.85 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 18 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 554 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 41.5 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 7.43 MB [initial] [rendered]


webpack: Compiled successfully.
```

ng serve ---> angular CLI calls tsc under the hood

<> index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>HelloApp</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```



```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>HelloApp</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
  <script type="text/javascript" src="inline.bundle.js"></script><script
type="text/javascript" src="polyfills.bundle.js"></script><script
type="text/javascript" src="styles.bundle.js"></script><script
type="text/javascript" src="vendor.bundle.js"></script><script
type="text/javascript" src="main.bundle.js"></script></body>
</html>
```

Injected at runtime

3. Architecture



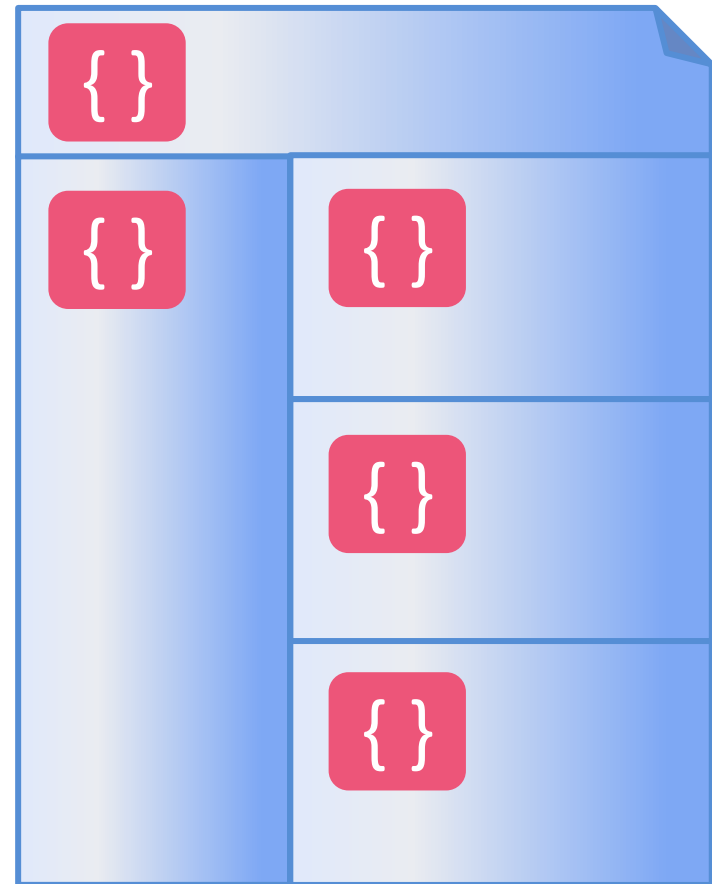
Component

Encapsulates the template, data and the behavior of a view.

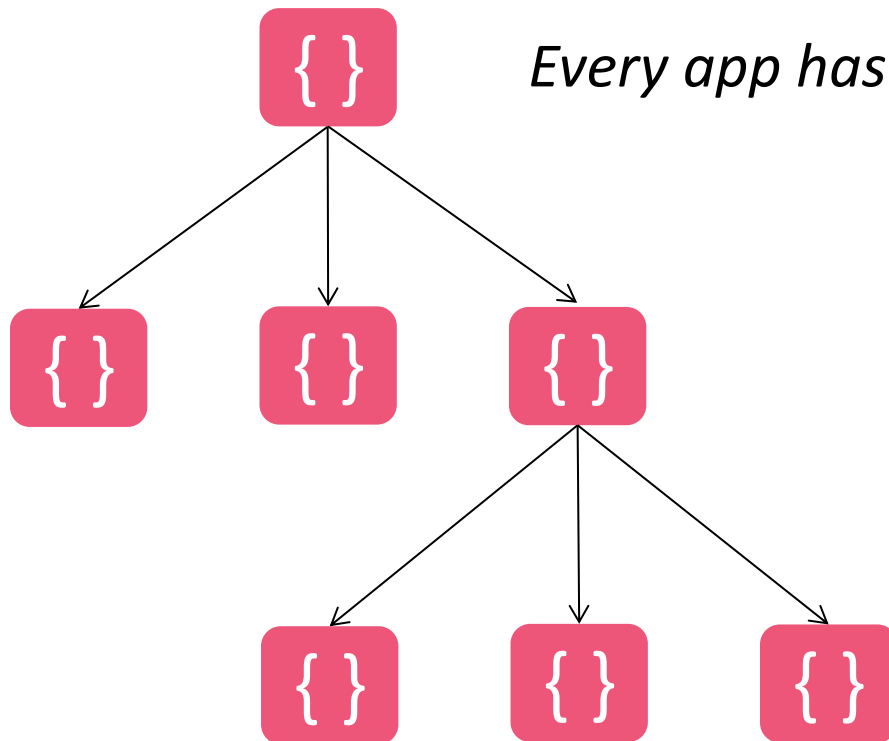
VIEW

Screen element that Angular can modify according to the program logic and data.

UI



Component



Every app has at least a root component.



Component

- ✓ Components are simply classes with decorators.
- ✓ The **@Component** decorator identifies a class as a component.
- ✓ A **decorator** provide related component-specific **metadata** that tells Angular how to use them.
- ✓ The component contains application **data and logic**.

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent {  
  emptytitle = 'Towa';  
  apptitle: string = 'data';  
}
```

metadata

data and logic

.ts-file



Component

*Each component is associated with an HTML that **defines a view** (patch of screen) to be displayed in a target environment.*

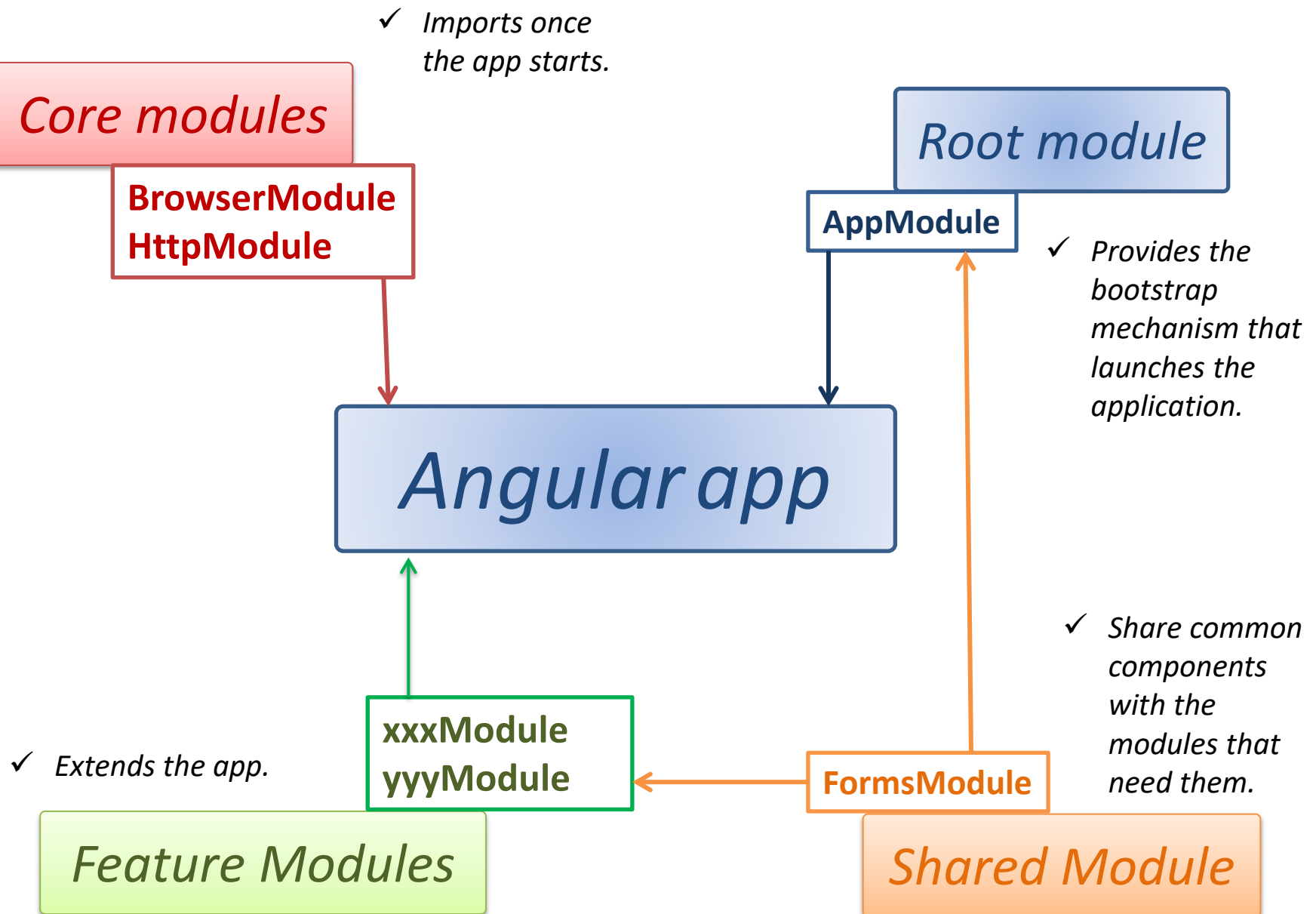
```
{ # xxx.component.css  
  <> xxx.component.html  
  TS xxx.component.ts  
}
```

NgModules

- ✓ **The code of an Angular application is organized in Ngmodules**, that helps in managing development of complex applications, and in designing for reusability.
- ✓ **This technique lets you take advantage of lazy-loading**—that is, *loading modules on demand*—in order to minimize the amount of code that needs to be loaded at startup.
- ✓ **Every Angular app has a root module**, conventionally named **AppModule**, which provides the bootstrap mechanism that launches the application.

NgModules

- ✓ **A Node module is a reusable block of code** whose existence does not accidentally impact other code.
- ✓ You can write your own modules and use it in various application.
- ✓ Node.js has a set of built-in modules which you can use without any further installation.



```
@NgModule({
  declarations: [
    AppComponent,
    TowadataComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  providers: [
    MovieService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Root module

AppModule

...more architectural elements will be presented later.

- ✓ *String interpolation.*
- ✓ *Hot module replacement.*
- ✓ *Creation of components.*
- ✓ *Structure of an angular project*

hello-app

Interpolation

To communicate properties
from the **component** class to the **template**.



(variables, objects, arrays, etc..)

The format for defining interpolation in a template is:

{{ propertyName }}

String interpolation

TS app.component.ts ●

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'app';
10 }
```

String interpolation

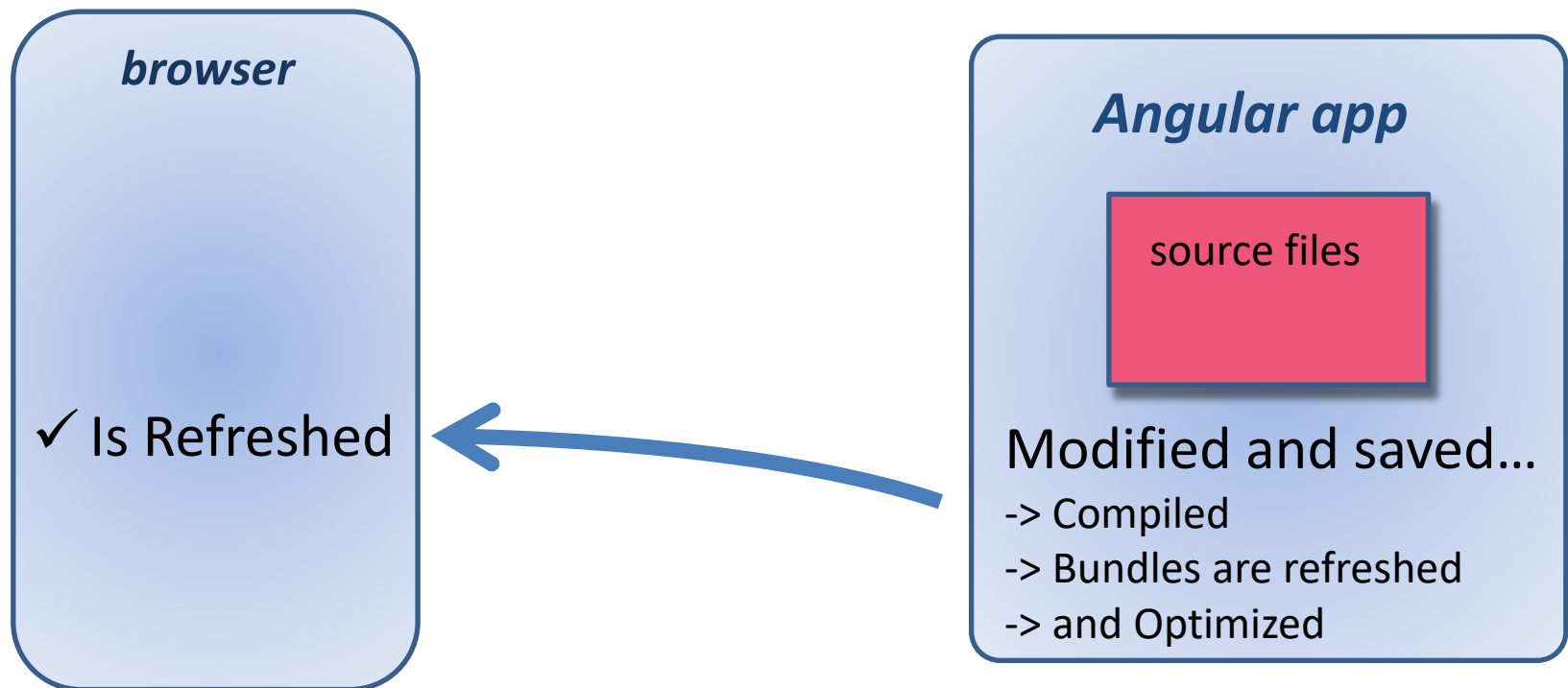
<> app.component.html ●

```
1  <div style="text-align:center">
2    <h1>
3      Welcome to {{ title }}!
4    </h1>
5    <img width="300" alt="Angular Logo" ...
6  </div>
7  <p>
8    Welcome text message.
9  </p>
```

Hot module replacement/reloading

HMR

Whenever one of the source file is modified and saved, the app is automatically compiled, bundles are refreshed and optimized and browser is refreshed.



- **Create** a component.
- **Register** it in a module.
- **Import** it in a component (ts).
- **Use** it in that component (html).

- **Delete** the component directory.
- **Remove** it from the module in which it is declared.

EJERCICIO:

- Realizar la aplicación hello-app.

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/hello-app.pdf

Tiempo disponible: 10 mins.

Structure of an angular app

e2e – end to end test, simulating a real user

node_modules – all the third party libraries that the app may depend upon
part of that party libraries are put in a bundle and deploy the application

src – source code

app – modules (at least one) and components (at least one)

assets – images, text, ...

environments – configuration for different environments (development or production)

favicon.ico – favorite icon display in the browser

index.html – index to the html

main.ts – starting point of a program, bootstrap the main module of our application

polyfills.ts –for features of JS that are not available in the current version of JS supported by most browsers, fills the gap

styles.css – global styles for the application

participants-app

participants-app

1. Requirements.

- a. A list with the name of the course participants.
- b. The detail of one participant.
- c. Add a participant.

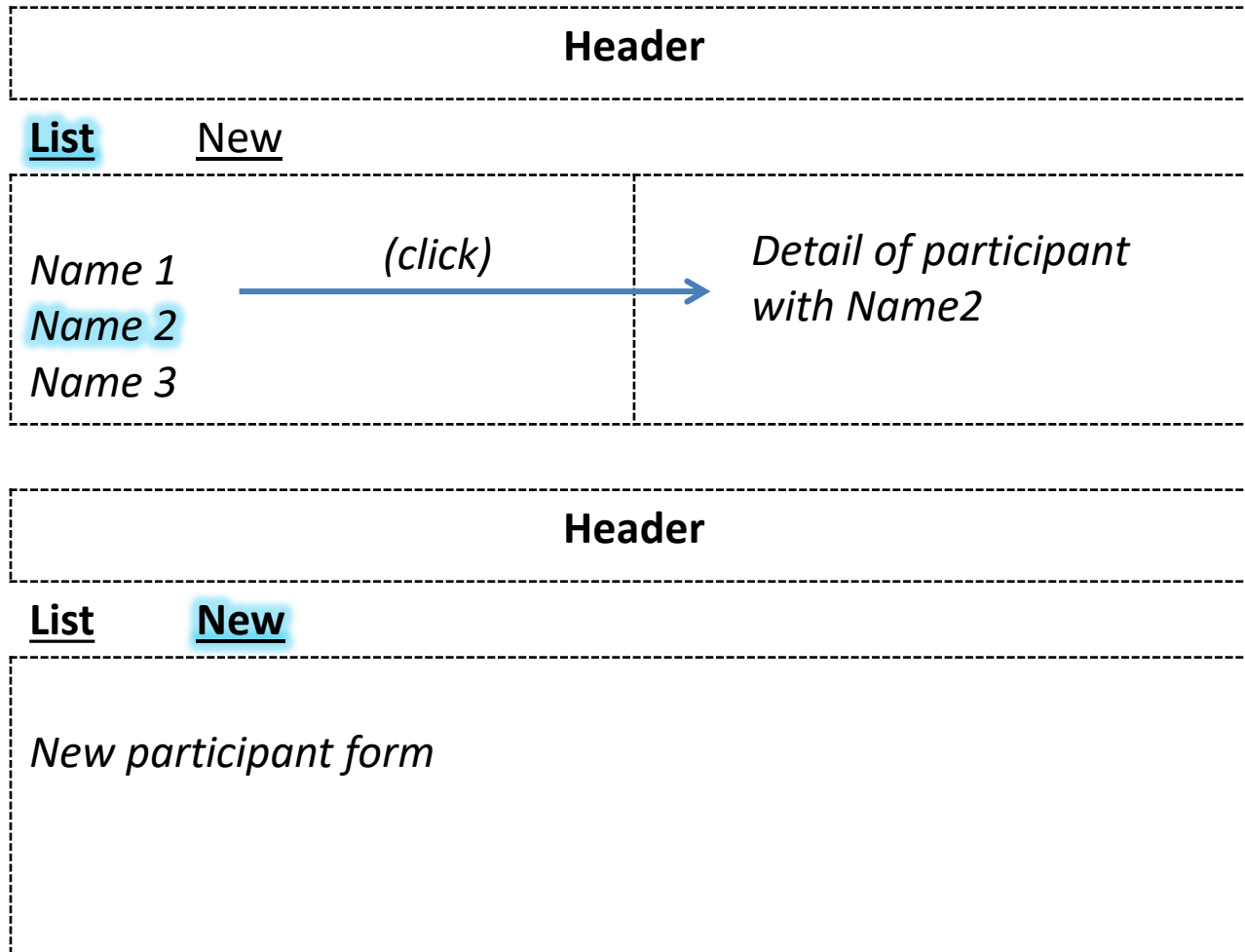
participants-app

2. Model.

Participant
initials: String
name: String
address: String
preferredLanguage: String

3. Views.

participants-app



4. Components.

`{ } app.component`

{ Header
Navigation bar

```
<nav>  
  <a routerLink = "/participants">Participants</a>  
  <a routerLink = "/new">New</a>  
</nav>
```

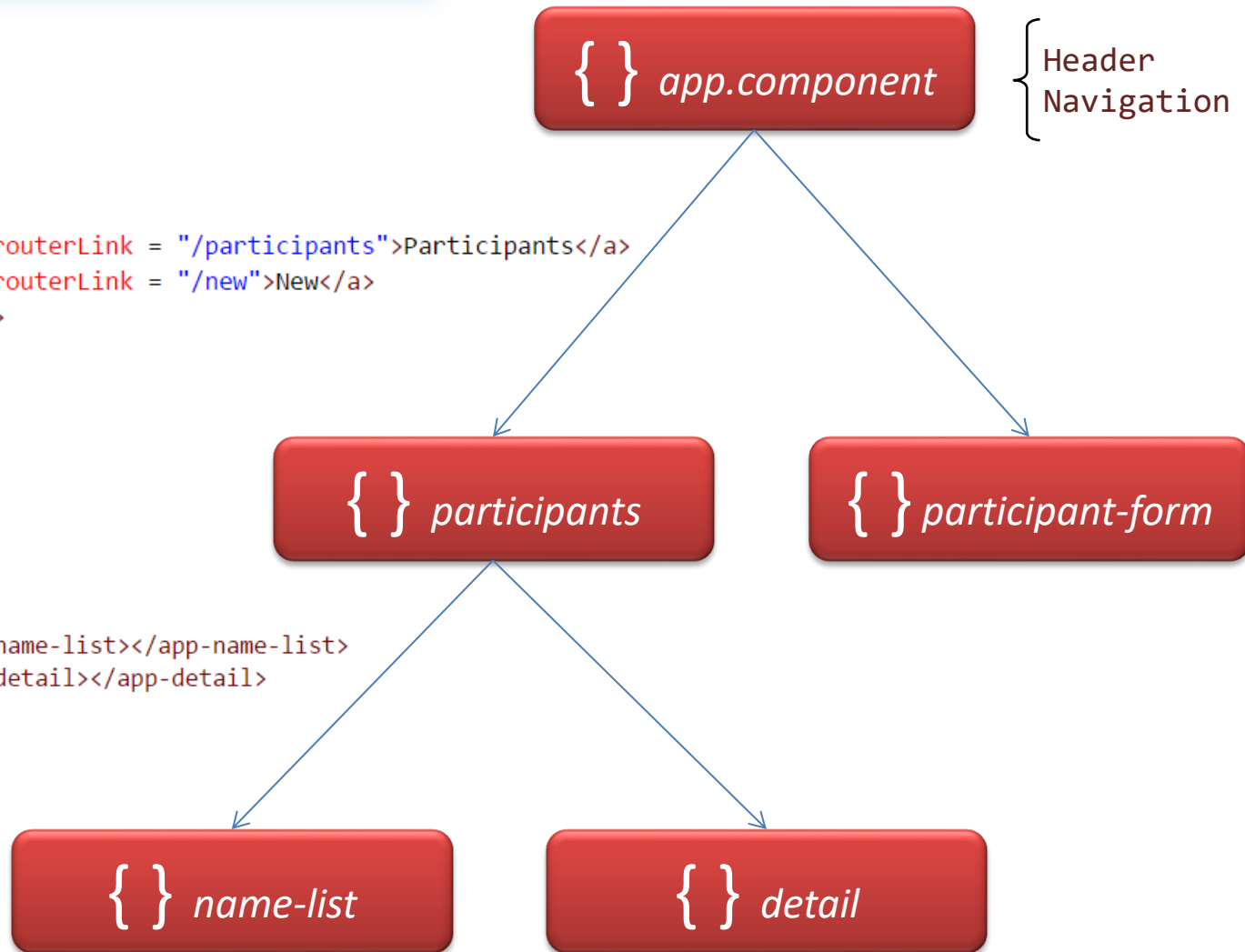
`{ } participants`

`{ } participant-form`

```
<div>  
  <app-name-list></app-name-list>  
  <app-detail></app-detail>  
</div>
```

`{ } name-list`

`{ } detail`



**Ver la aplicación funcionando.*

EJERCICIO:

Iniciar la aplicación participants-app:

- Realizar puntos 1-3.
- Realizar en grupo puntos 4-6
(routing and bootstrap).

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/participant-app.pdf

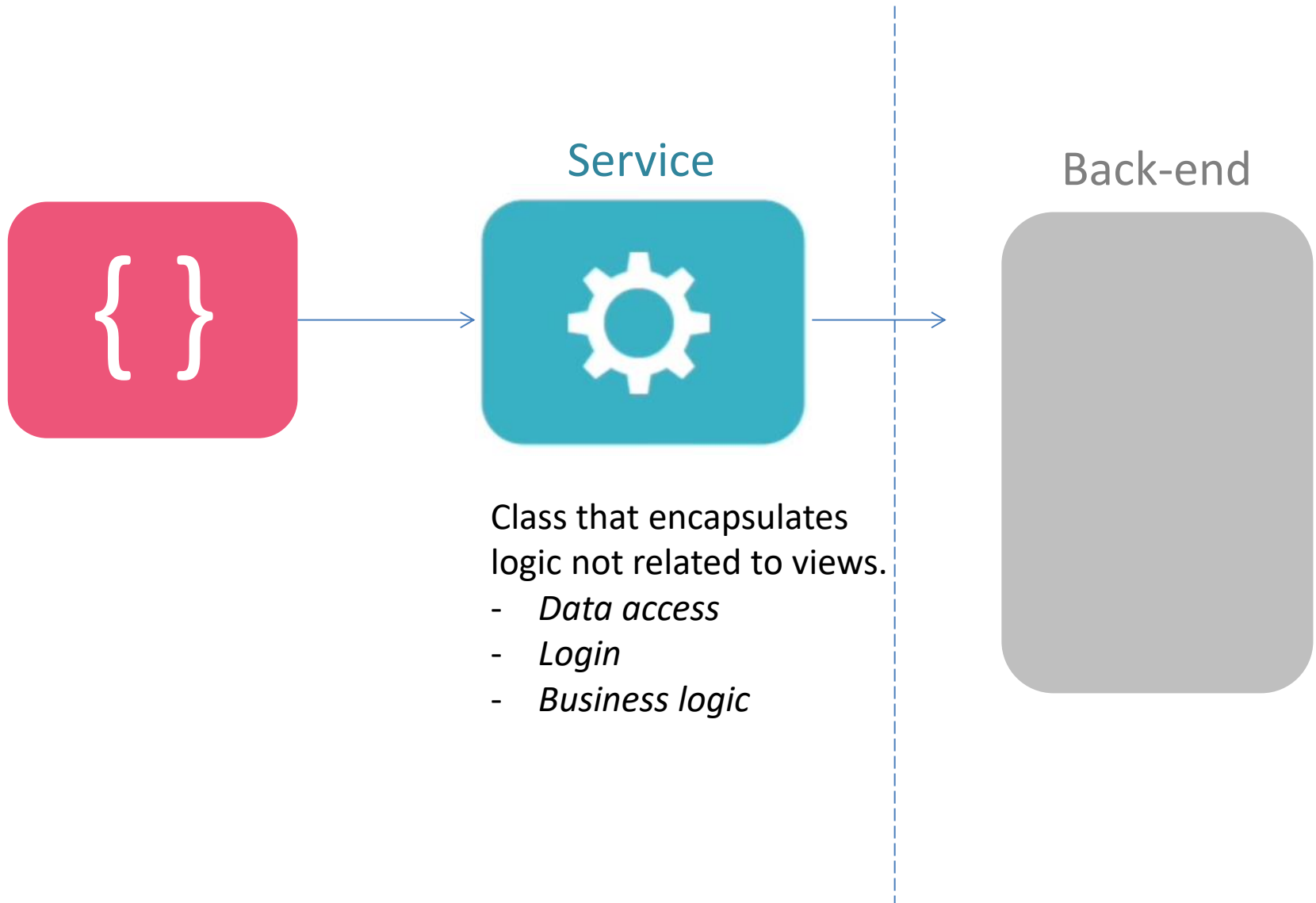
Tiempo disponible: 10+20 mins.

4. Services



Service

- ✓ Components use *services*, which provide specific functionality not directly related to views.
- ✓ Service providers can be *injected* into components as *dependencies*, making your code modular, reusable, and efficient.



1. Create the service.

ng generate service participant-data

participant-data.service.ts

```
import { Injectable } from '@angular/core';
```

```
@Injectable()
```

```
export class ParticipantDataService {
```

```
    constructor( )
```

```
}
```

2. Modify the service.

participant-data.service.ts

```
import { Injectable } from '@angular/core';
import { Participant } from '../model/participant';

@Injectable()
export class ParticipantDataService {

    participants: Participant[];
    constructor( ) {...}

    getParticipants(): Participant[] {...}
    putParticipant(participant1: Participant) {...}
}
```

3. Add the service as a provider in the module.

```
@ngModule({  
  
...  
providers: [  
    ParticipantDataService  
],  
...  
})
```

4. Import the service.

In the component where we are going to use the service

name-list.component.ts

```
import { ParticipantDataService } from '../participant-data.service'
```

```
participants: Participant[];
```

```
constructor() {
```

```
}
```

5. Use the service

In the component where we are going to use the service

name-list.component.ts

```
import { ParticipantDataService } from '../participant-data.service'

participants: Participant[];
private participantDataService: ParticipantDataService;

constructor(participantDataService: ParticipantDataService ) {
    this.participantDataService = participantDataService;
    this.participants = participantDataService.getParticipants();
}
```

In the component where we are going to use the service

name-list.component.ts

```
import { ParticipantDataService } from '../participant-data.service'
```

```
participants: Participant[];
```

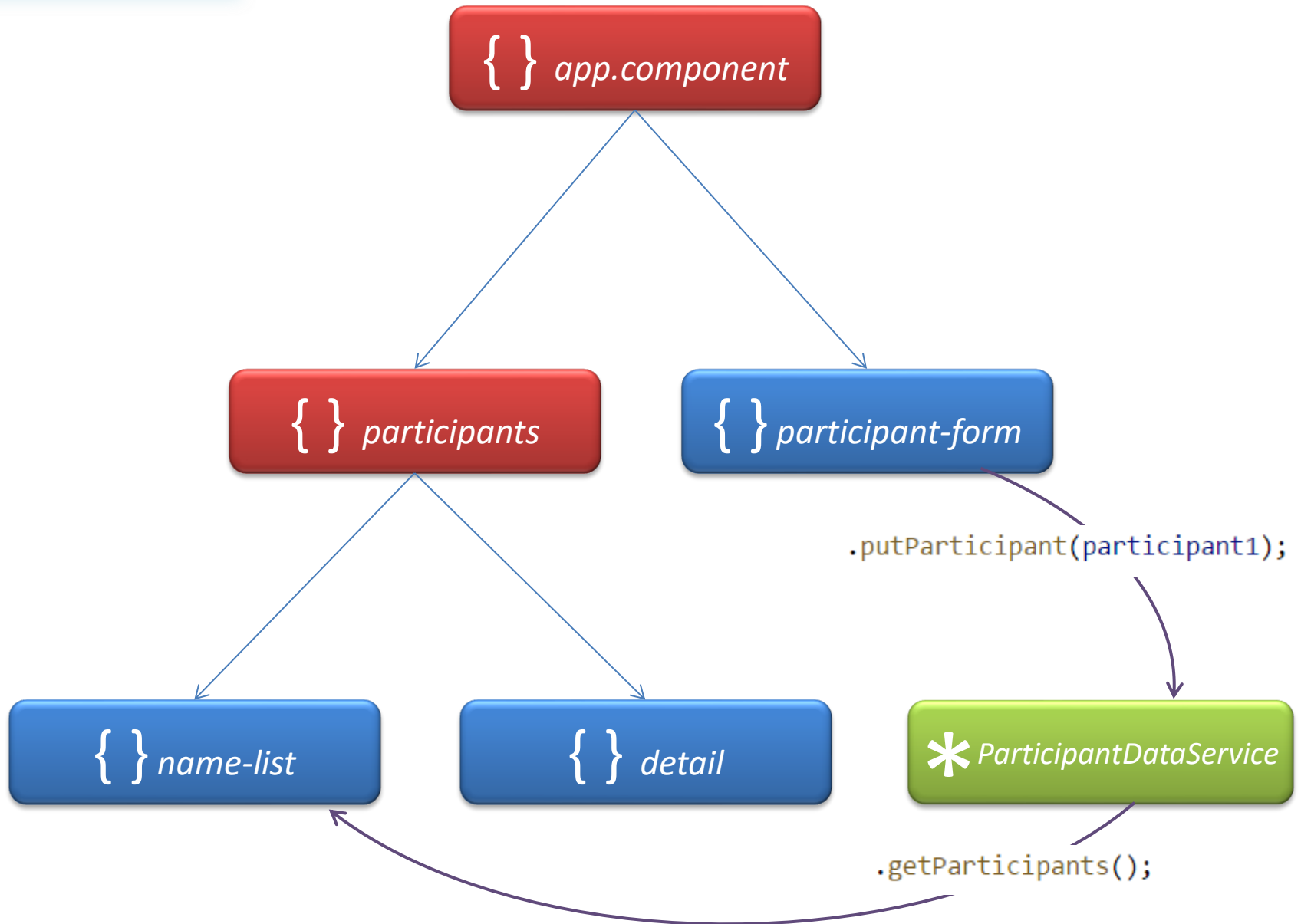
```
constructor(  
  private participantDataService: ParticipantDataService  
) { }
```

- ✓ Create a private data
- ✓ Create a constructor parameter
- ✓ Initialize the data with the parameter

```
ngOnInit() {  
  this.participants = this.participantDataService.getParticipants();  
}
```


participants-app

5. Services.



EJERCICIO:

Continuar la aplicación participants-app:

- Realizar el ejercicio 7
(*services*).

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/participant-app.pdf

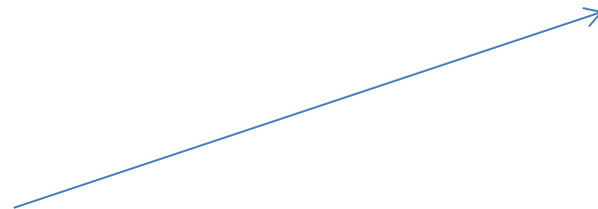
Tiempo disponible: 20 mins.

❑ *ngIf

❑ *ngFor

Property Binding

```
<p>  
  <input type="text" placeholder = 'Name:' />  
  <input type="text" placeholder = "{{ placeholderName }}" />  
  <input type="text" [placeholder] = "placeholderName" />  
  <input type="text" bind-placeholder = "placeholderName" />  
</p>
```



VIEW

```
export class ParticipantFormComponent implements OnInit {  
  placeholderName: String = "Name:" //Just for property binding
```

Two-Way Data Binding

[(ngModel)]

TS app.module.ts x

```
20 imports: [  
21   BrowserModule,  
22   AppRoutingModule,  
23   FormsModule  
24 ],  
25 providers: [  
26   ParticipantDataService  
27 ],  
28 bootstrap: [AppComponent]  
29 })  
30 export class AppModule { }
```



```
<p>  
  <input type="text" placeholder = 'Name:' [(ngModel)]="participant.name" />  
</p>  
<p>  
  {{ participant.name }}  
</p>
```

Event Binding

(ngModel)

```
<p>  
  <button (click)="newHandler(participant)"> New </button>  
</p>
```

VIEW



```
newHandler(participant1: Participant){  
  //...  
}
```

EJERCICIO:

Continuar la aplicación participants-app:

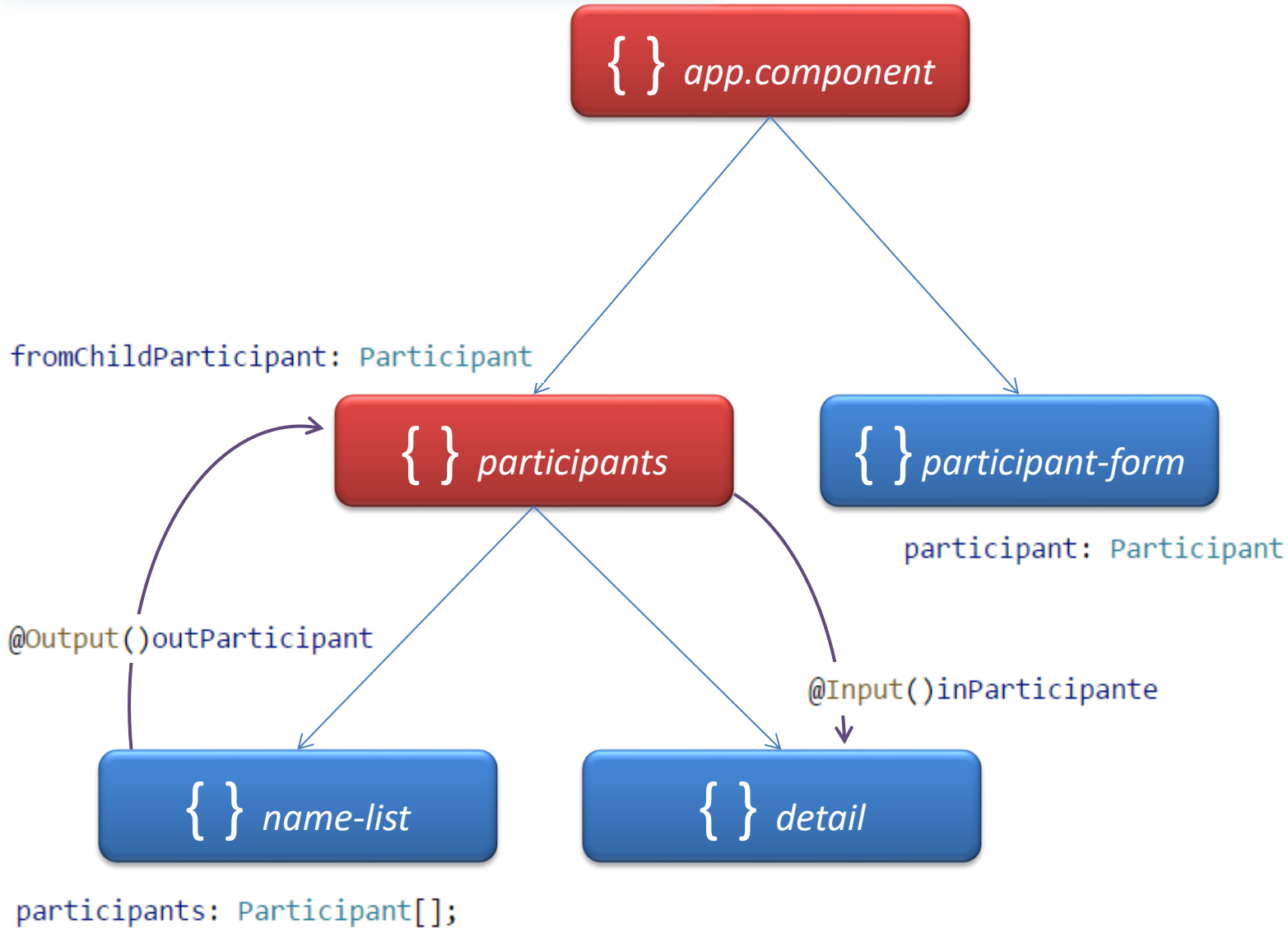
- Realizar el ejercicios 8 y 9
(directivas y binding).

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/participant-app.pdf

Tiempo disponible: 20 mins.

6. Data communication.



EJERCICIO:

Continuar la aplicación participants-app:

- Realizar el ejercicio 10
(*@output, @input*).

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/participant-app.pdf

Tiempo disponible: 20 mins.

Web API

- ✓ Use **SSMS – MSSQL Server Management Studio** to create a DB and a table.
- ✓ Use **MSVS – MS Visual Studio** to create a new C# - ASP.NET Web Application.
- ✓ Use ADO.NET Entity Data Model to retrieve data from the database.
- ✓ Add the ApiController.

More Get Ready



Developer

SQL Server 2017 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.

Download now ↓



Microsoft **SQL Server Management Studio 17**

Desktop app

SSMS is free!

SSMS 17.x is the latest generation of *SQL Server Management Studio* and provides support for SQL Server 2017.



[Download SQL Server Management Studio 17.7](#)

Visual Studio Installer

Productos

Instalados



Visual Studio Community 2017

15.7.1

IDE gratuito y rico en contenido para estudiantes y desarrolladores individuales y de código abierto

[Notas de la versión](#)

Modificar

Iniciar

Más ▼

Web y nube (7)



Desarrollo de ASP.NET y web

Compila aplicaciones web con ASP.NET, ASP.NET Core, HTML/JavaScript y Containers, además de ofrecer...



Desarrollo de Python

Edición, depuración, desarrollo interactivo y control de código fuente de Python.



Almacenamiento y procesamiento de datos

Conecte, desarrolle y pruebe soluciones de datos con SQL Server, Azure Data Lake o Hadoop.



HTTP Methods

- GET: Retrieve data from a specified resource
- POST: Submit data to be processed to a specified resource
- PUT: Update a specified resource
- DELETE: Delete a specified resource

Endpoint

The URL where api/service can be accessed by a client application

GET <http://localhost:56618/api/participants>

GET <http://localhost:56618/api/participants/LGF>

PUT <http://localhost:56618/api/participants/LGF>

EJERCICIO:

Continuar la aplicación participants-app:

- Realizar los ejercicios 11 y 12
(DB and Web API to get data).

Seguir las indicaciones de:

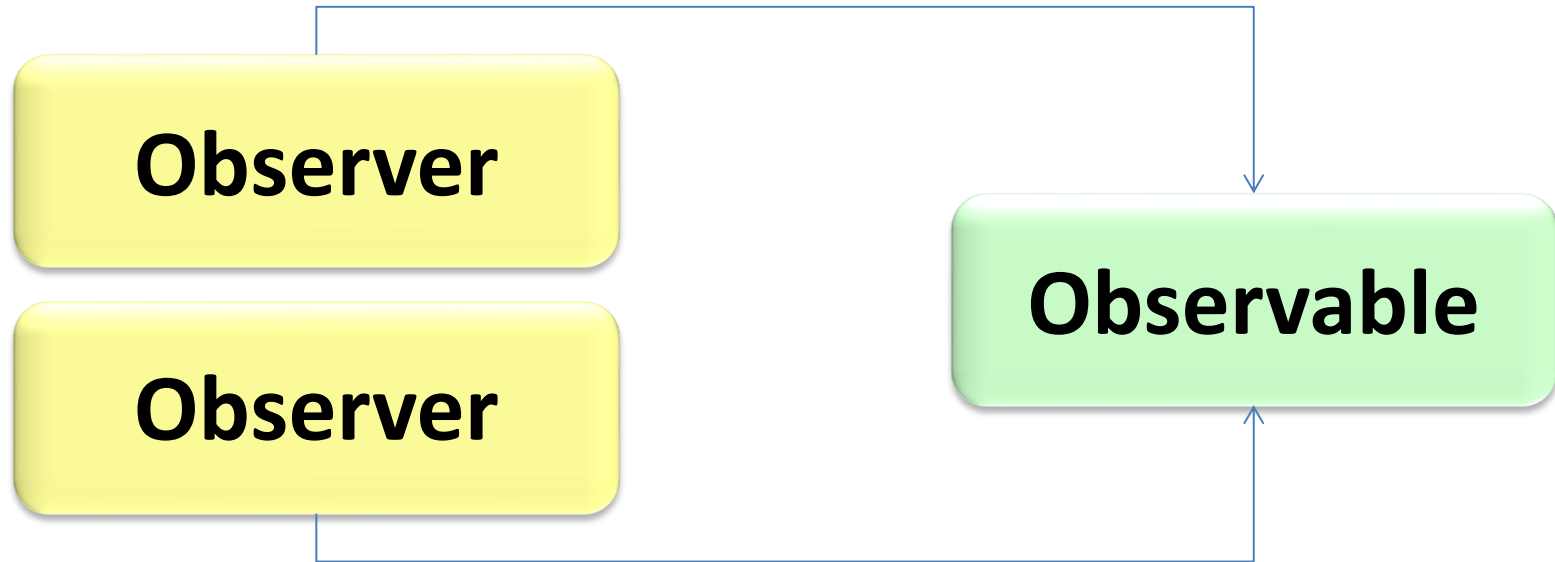
https://github.com/LilianaGF/angular_course/blob/master/participant-app.pdf

Tiempo disponible: 1 hora.

Observable

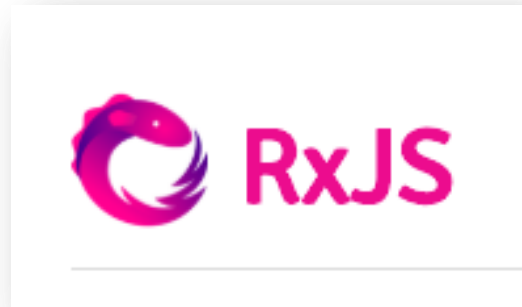
When version 2 of Angular came out, it introduced us to observables. The Observable isn't an Angular specific feature, but a new standard for managing async data that will be included in the ES7 release. Angular uses observables extensively in the event system and the HTTP service.

Asynchronous pattern

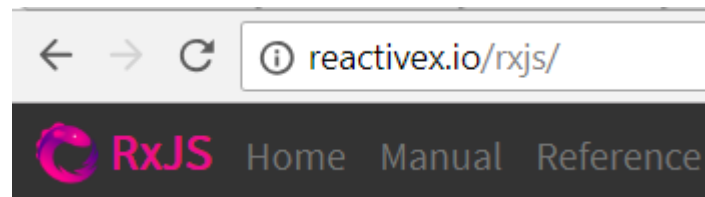


- *observers subscribe to an observable.*
- *observable emits an item or a sequence of items.*
- *observers react to whatever the observable emits.*

When the observer subscribes to the observable, it specifies the callback function. The callback function is notified whenever the observable emits data items. The callback function has code to handle the data.



The ReactiveX programming library for JavaScript



The ReactiveX library for JavaScript.

RxJS is a library for reactive programming using Observables, to make it easier to compose asynchronous or callback-based code. This project is a rewrite of [Reactive-Extensions/RxJS](#) with better performance, better modularity, better debuggable call stacks, while staying mostly backwards compatible, with some breaking changes that reduce the API surface.

```
import { Observable } from 'rxjs/Observable';  
import 'rxjs/add/operator/map'
```

_http:Http

get(url): Observable<Response>

: Observable<Response>

map()

observableParticipants:
Observable<Participant[]>

```
@Injectable()
export class ParticipantDataService {

  observableParticipants: Observable<Participant[]>;

  constructor(
    | private _http: Http
  ) {
    | this.observableParticipants =
    | | this._http.get("http://localhost:56618/api/participants/")
    | | .map((response: Response) => <Participant[]>response.json());
  }

  getParticipants(): Observable<Participant[]> {
    | return this.observableParticipants;
  }
}
```

observableParticipants:
Observable<Participant[]>

TS name-list.component.ts ✕

```
ngOnInit() {  
  this.participantDataService.getParticipants()  
    .subscribe((participantsData)=>this.participants = participantsData);  
}
```

EJERCICIO:

Continuar la aplicación participants-app:

- Realizar el ejercicio 13
(*http get*).

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/participant-app.pdf

Tiempo disponible: 1 hora.

EJERCICIO:

Continuar la aplicación participants-app:

- Realizar los ejercicios 14 y 15
(Web API to post data ,http post).

Seguir las indicaciones de:

https://github.com/LilianaGF/angular_course/blob/master/participant-app.pdf

Tiempo disponible: 1 hora.

Interesting links:

Angular 4 tutorial video:

<https://www.youtube.com/watch?v=k5E2AVpwsko>

Angular 2 tutorial video:

https://www.youtube.com/watch?v=_-CD_5YhJTA