ANGULAR INTRO

LUXOFT

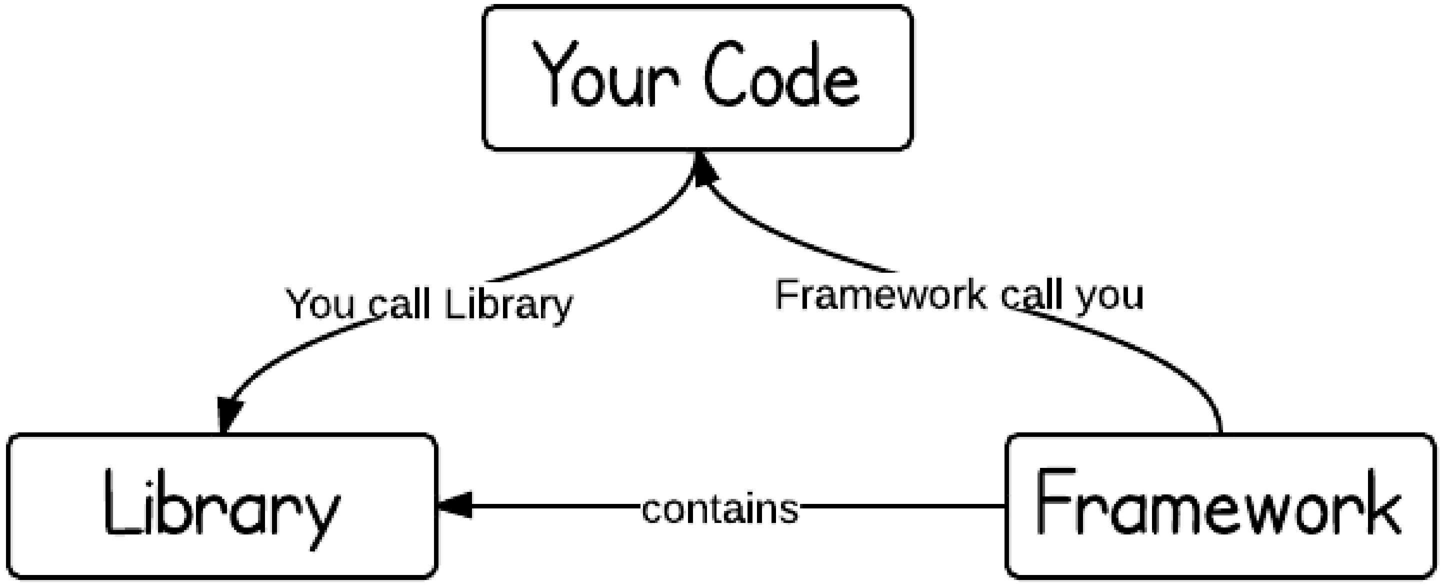
LXFT LISTED NYSE

WHY TO USE HTML5 FRAMEWORKS

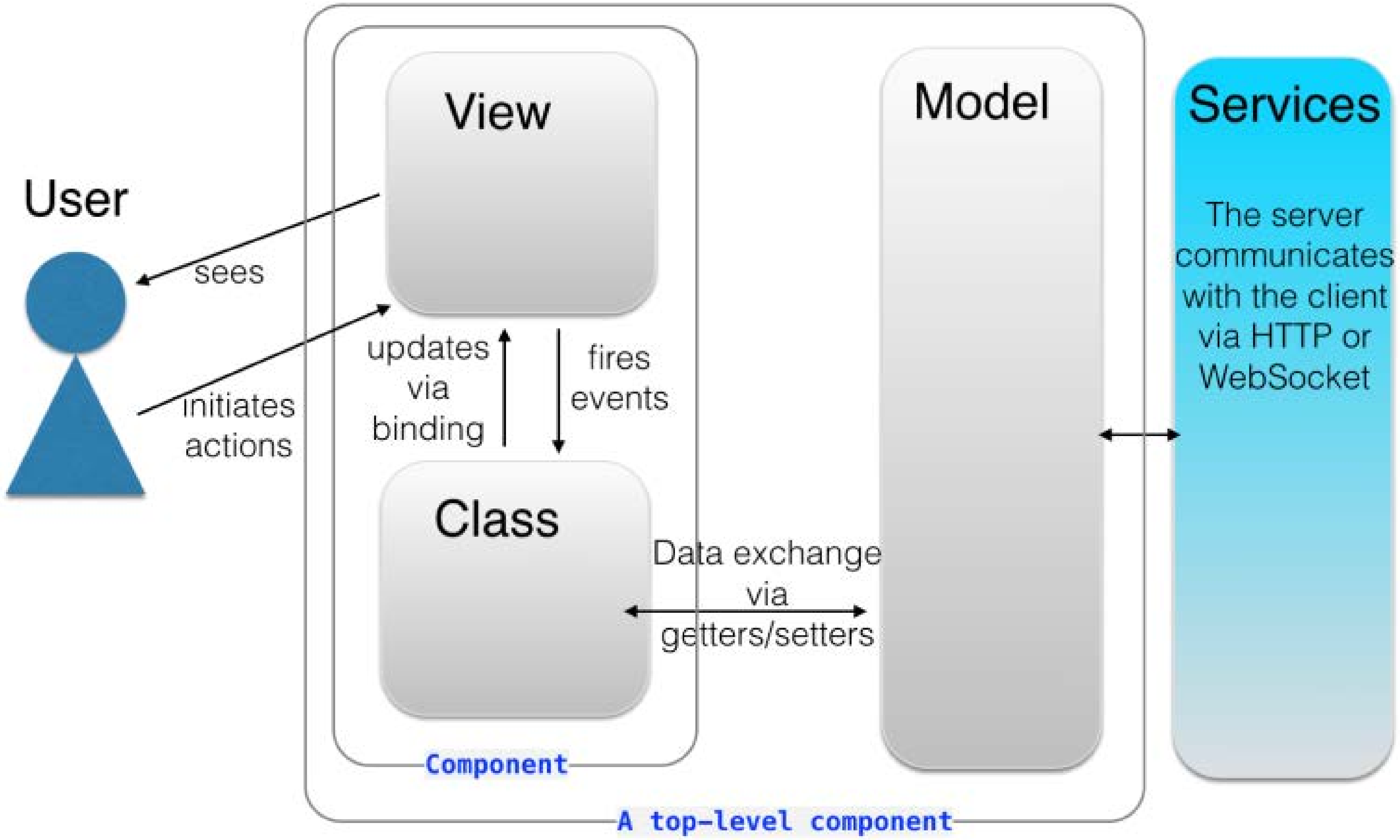
- Deal with cross-browser compatibility
- Make your application more structured
- May include reusable components
- Make programmers more productive
- Lower the amount of manually written code

FRAMEWORKS VS. LIBRARIES

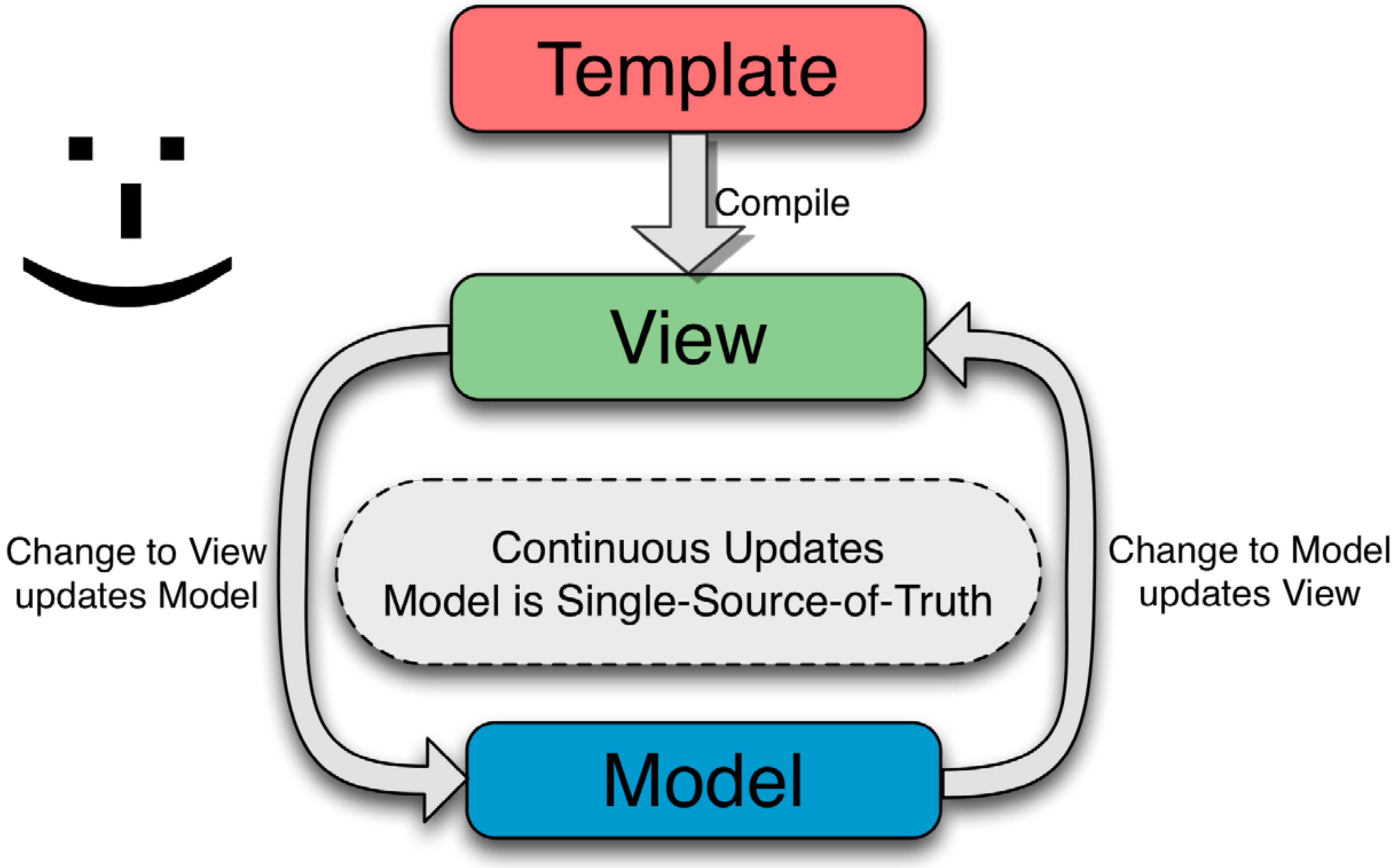
- Frameworks expect you to develop using well defined rules.
- Libraries just offer reusable components



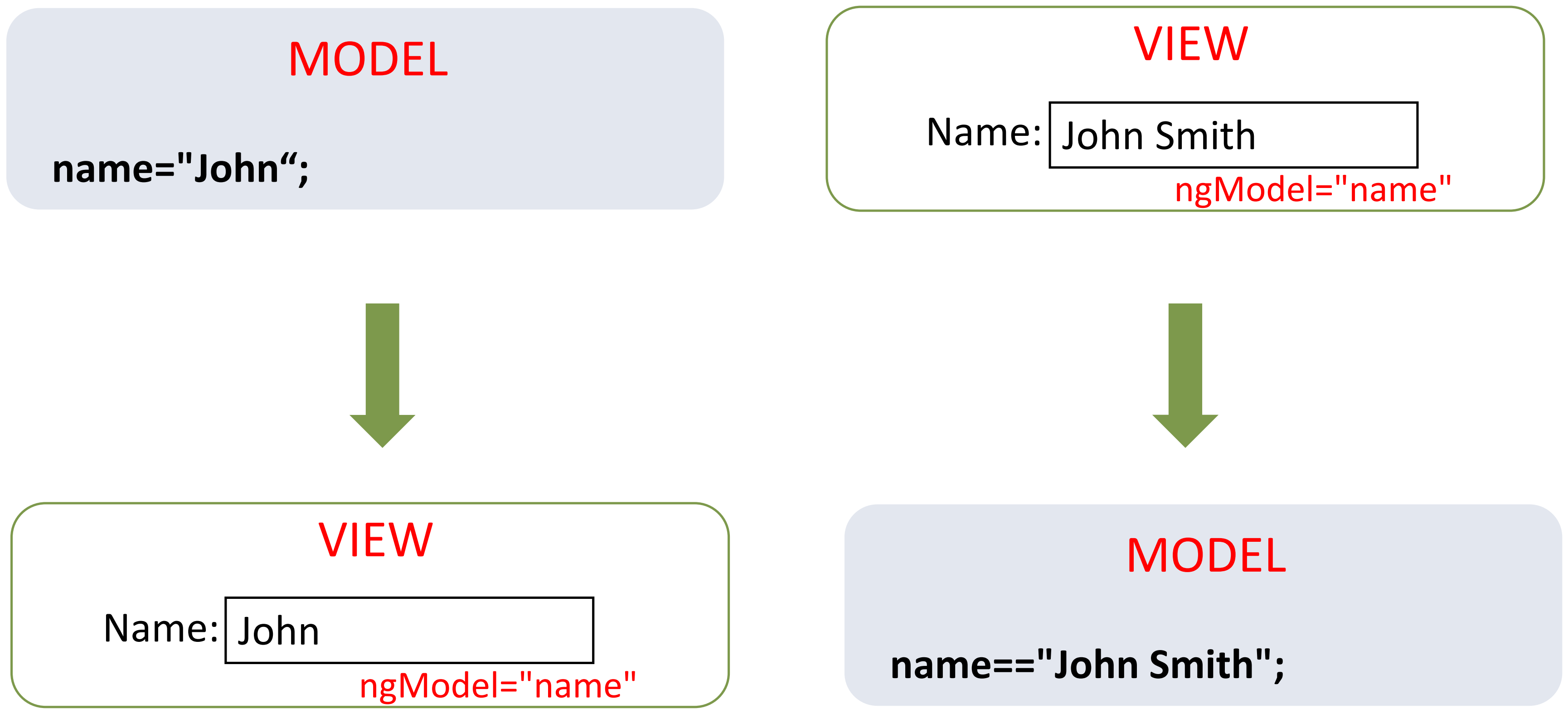
MVC model


User
sees
initiates actions
View
updates via binding
fires events
Class
Component
Data exchange via getters/setters
Model
Services
The server communicates with the client via HTTP or WebSocket
A top-level component

Two-way data binding

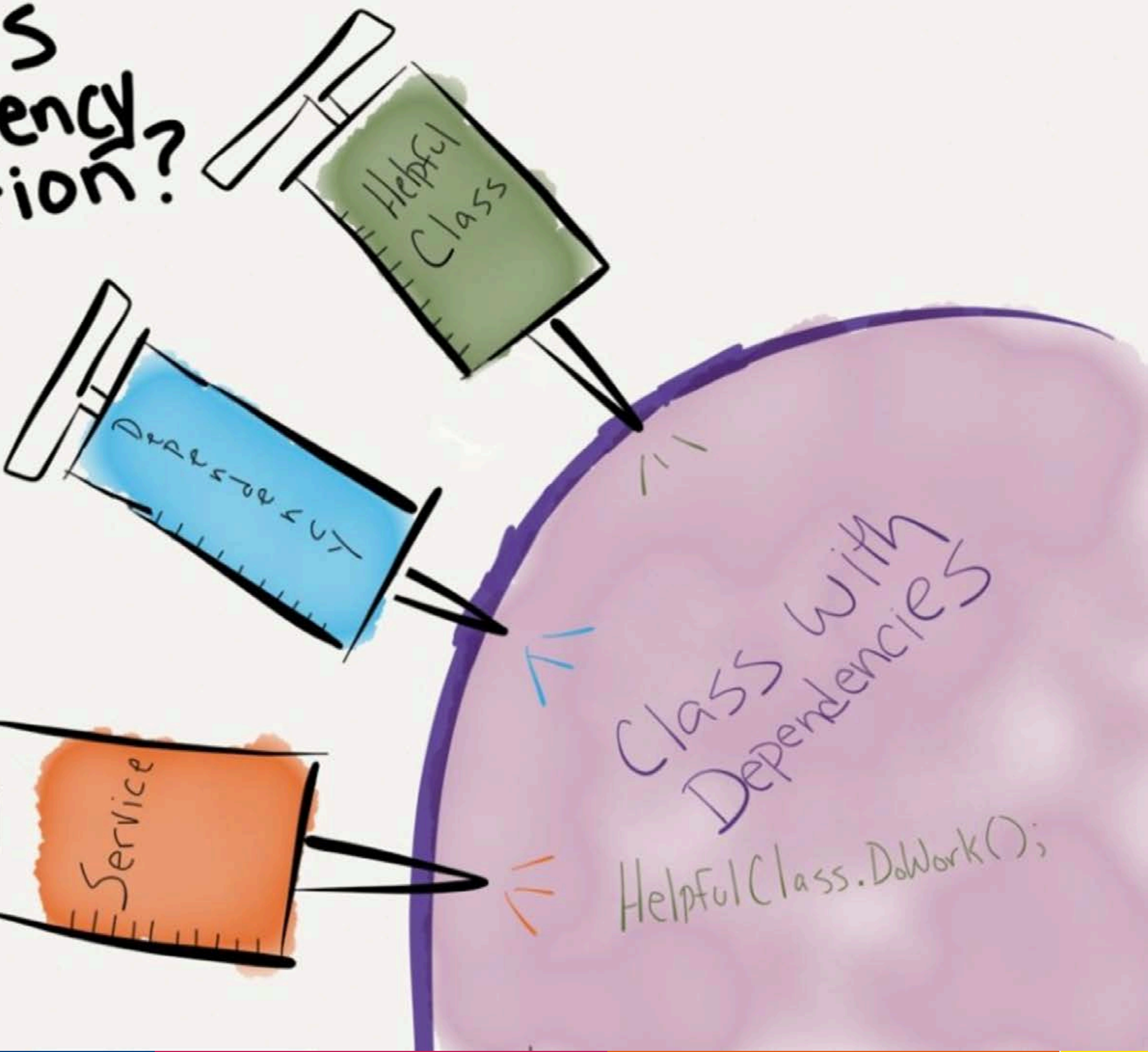

Template
Compile
View
Change to View updates Model
Continuous Updates
Model is Single-Source-of-Truth
Change to Model updates View
Model

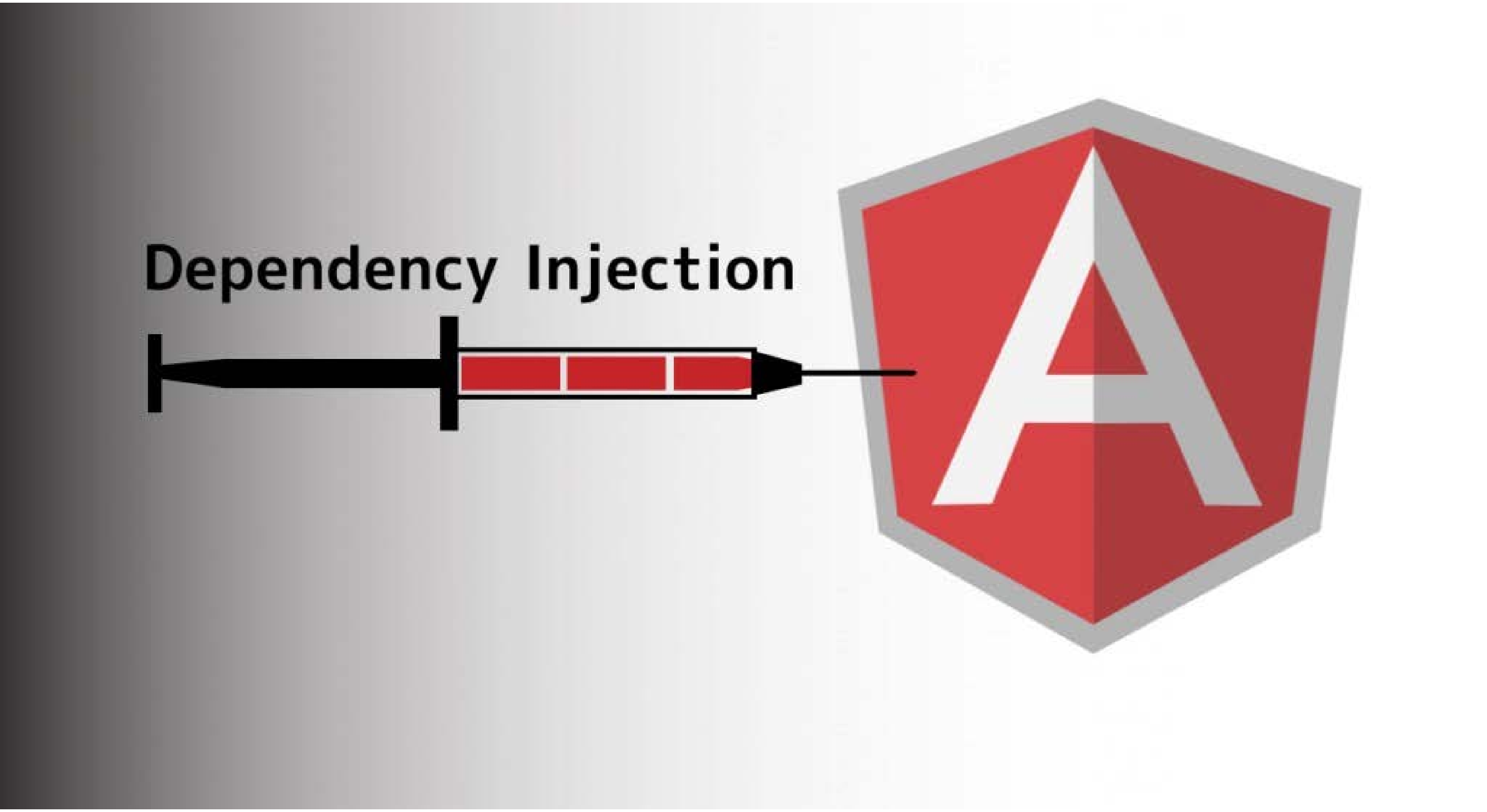
TWO WAY DATA BINDING EXAMPLE



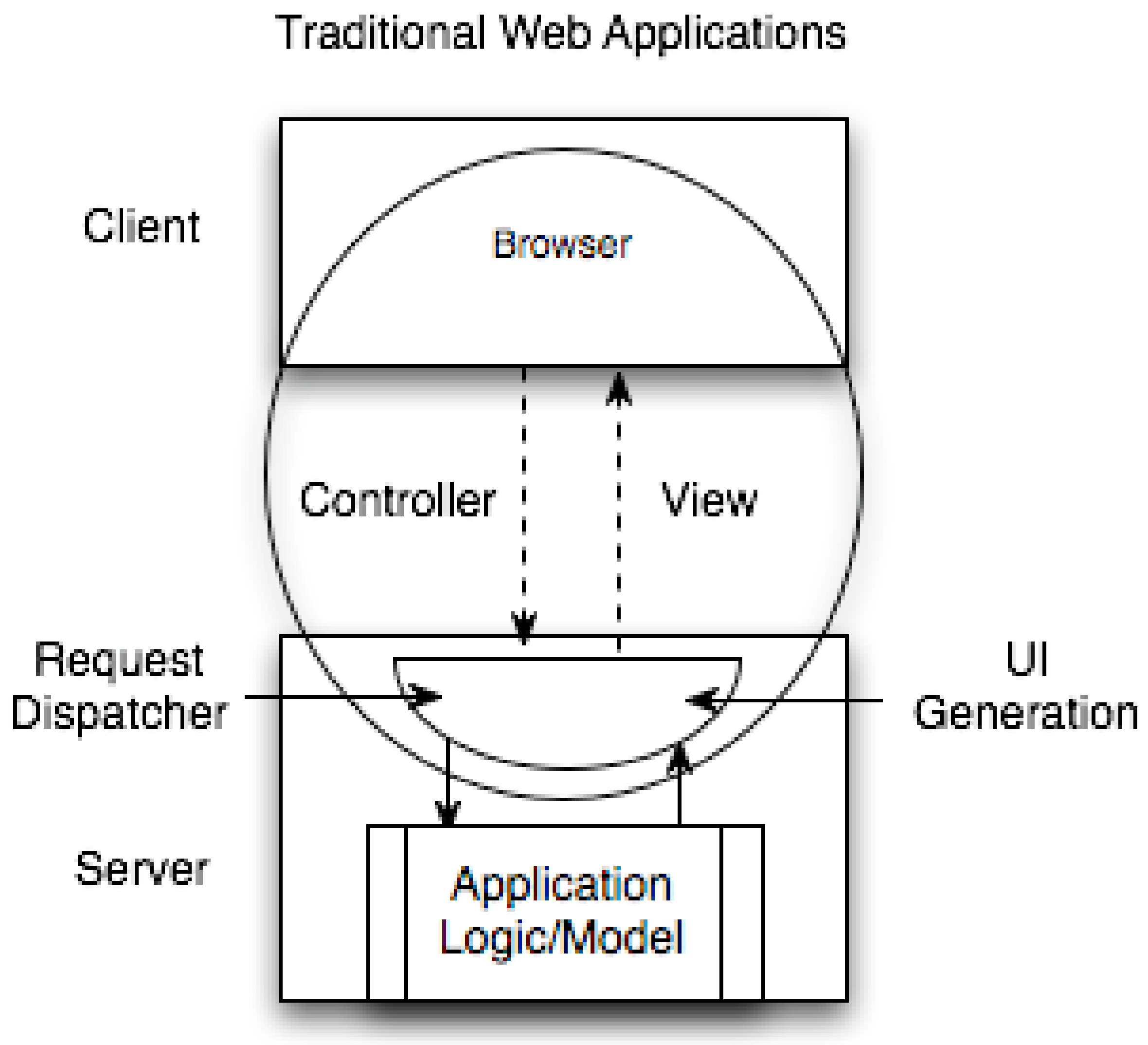
What is Dependency Injection?

& Why is it useful?




Dependency Injection

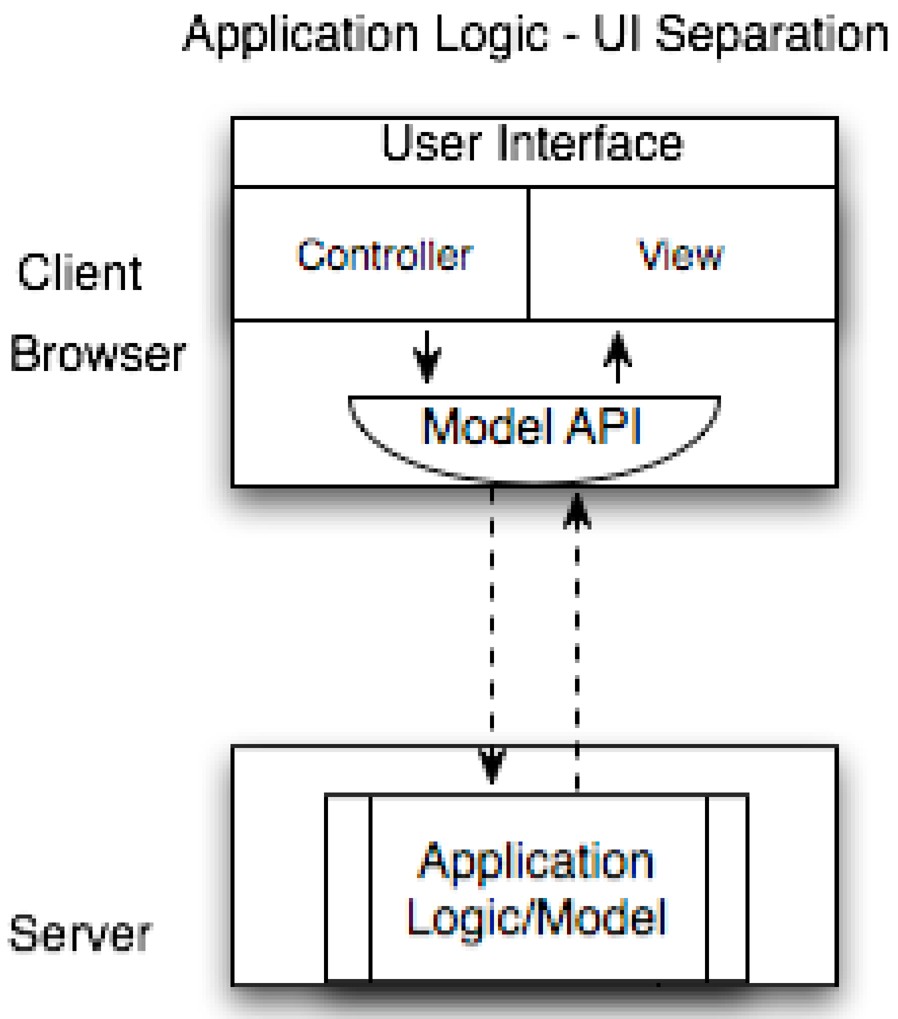
Why use a JS MVC framework



- **Poor distribution of processing** – With a large number of clients, doing all the processing on the server is inefficient.
- High user response latency
- Difficult programming model
- Increased vector of attack
- Heavy state management on the servers
- Offline Difficulties
- Reduced opportunity for interoperability

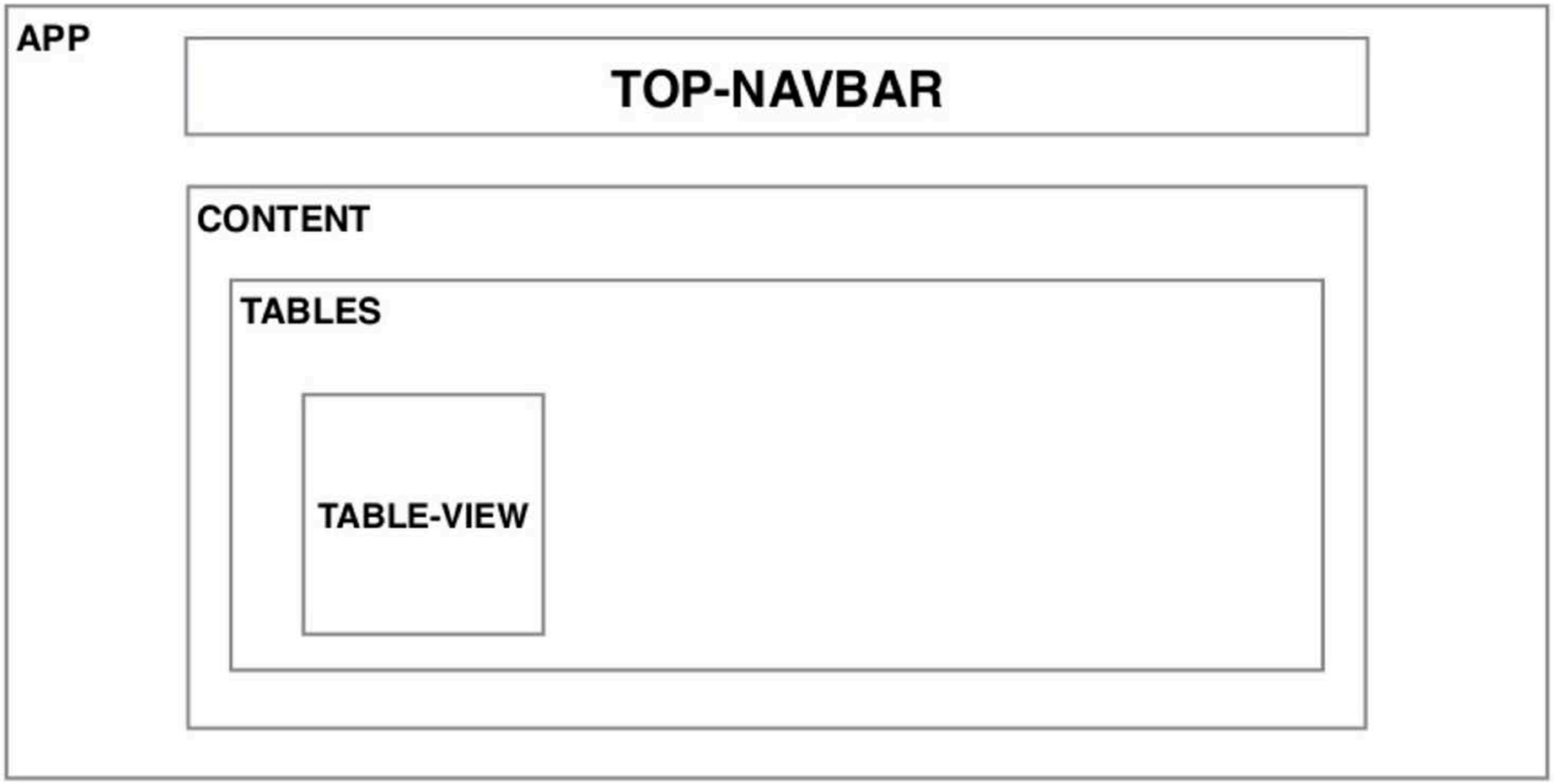
Why use a JS MVC framework

JSMVC Web Applications

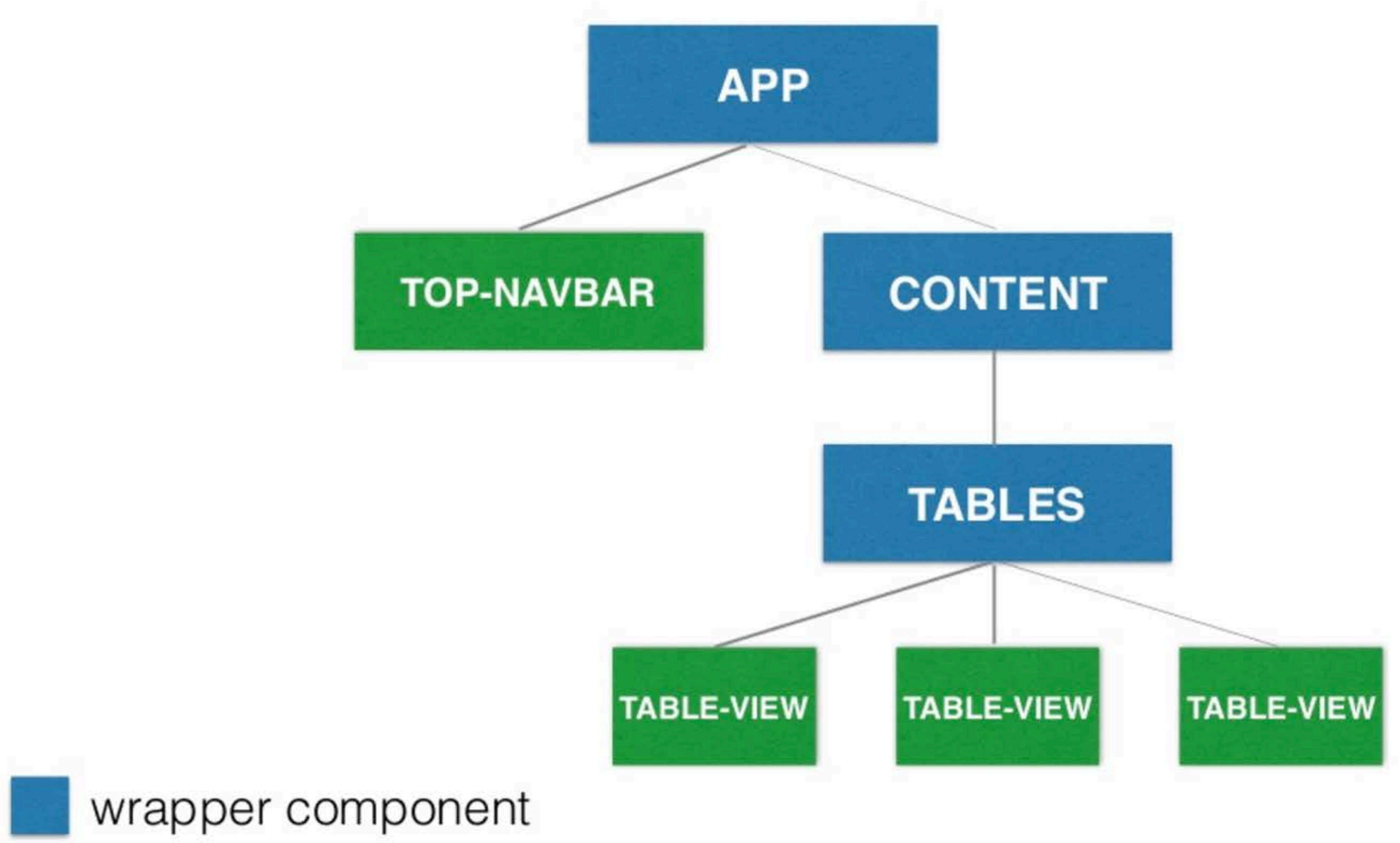


- **Scalability** - The more clients that use an application, the more client machines that are available, whereas the server processing capabilities remain constant
- Immediate **user response**
- Organized **programming model**
- Client side **state management**
- **Offline** applications
- Interoperability

Thinking in components



Thinking in components



MOST BASIC COMPONENT

In JavaScript

```
import { Component } from 'angular2/core';

@Component({
  selector: 'App',
  template: ‘<h1>Hello Component!</h1>’
})

class App {}
```

Use in HTML

```
<body>

  <App></App>

</body>
```

COMPONENT COMPOSITION

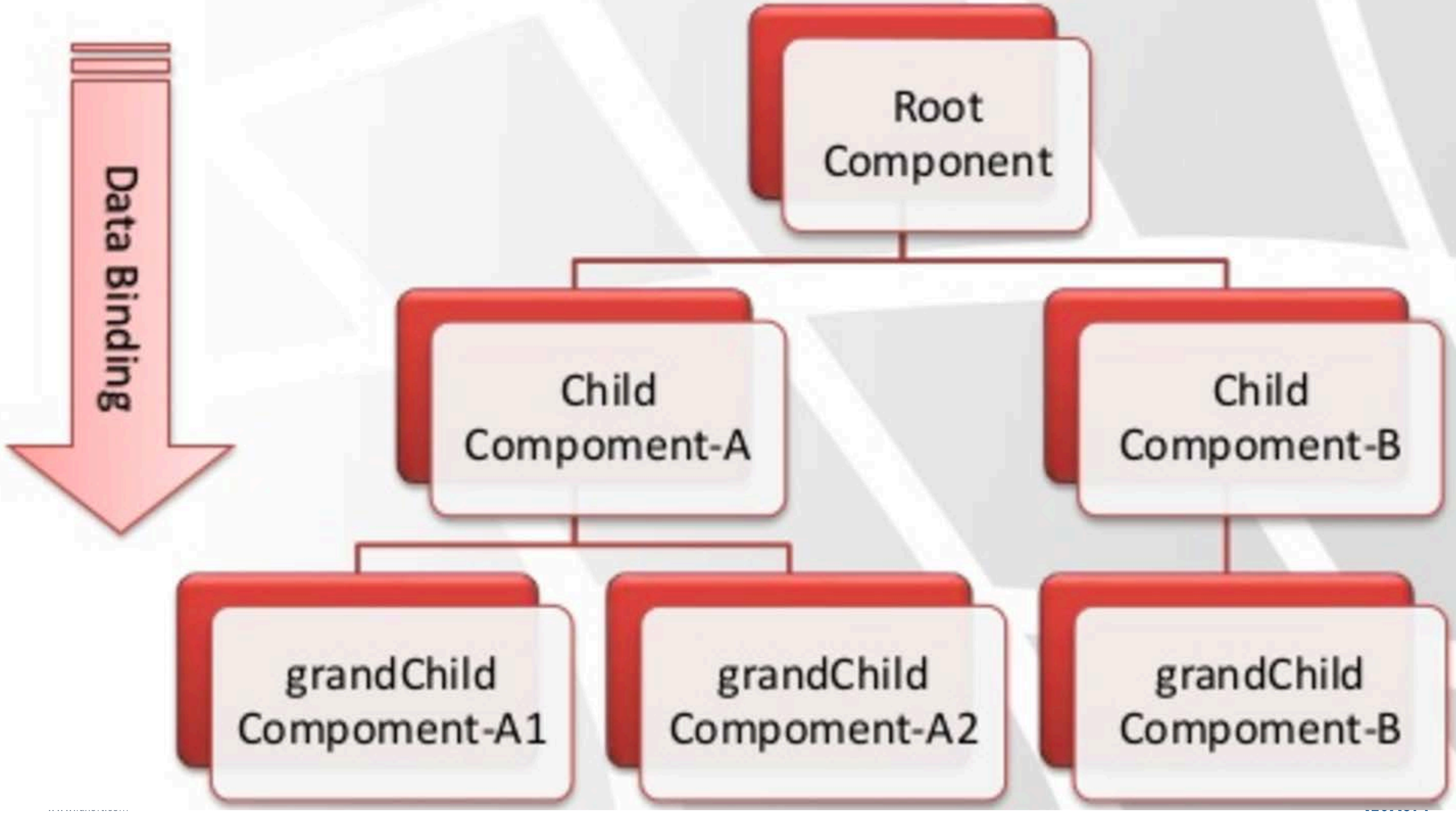
content.ts

```
import {Component} from "angular2/core";

@Component({
  selector:'content',
  template:`<div class="container"></div>`
})

export class Content {}
```

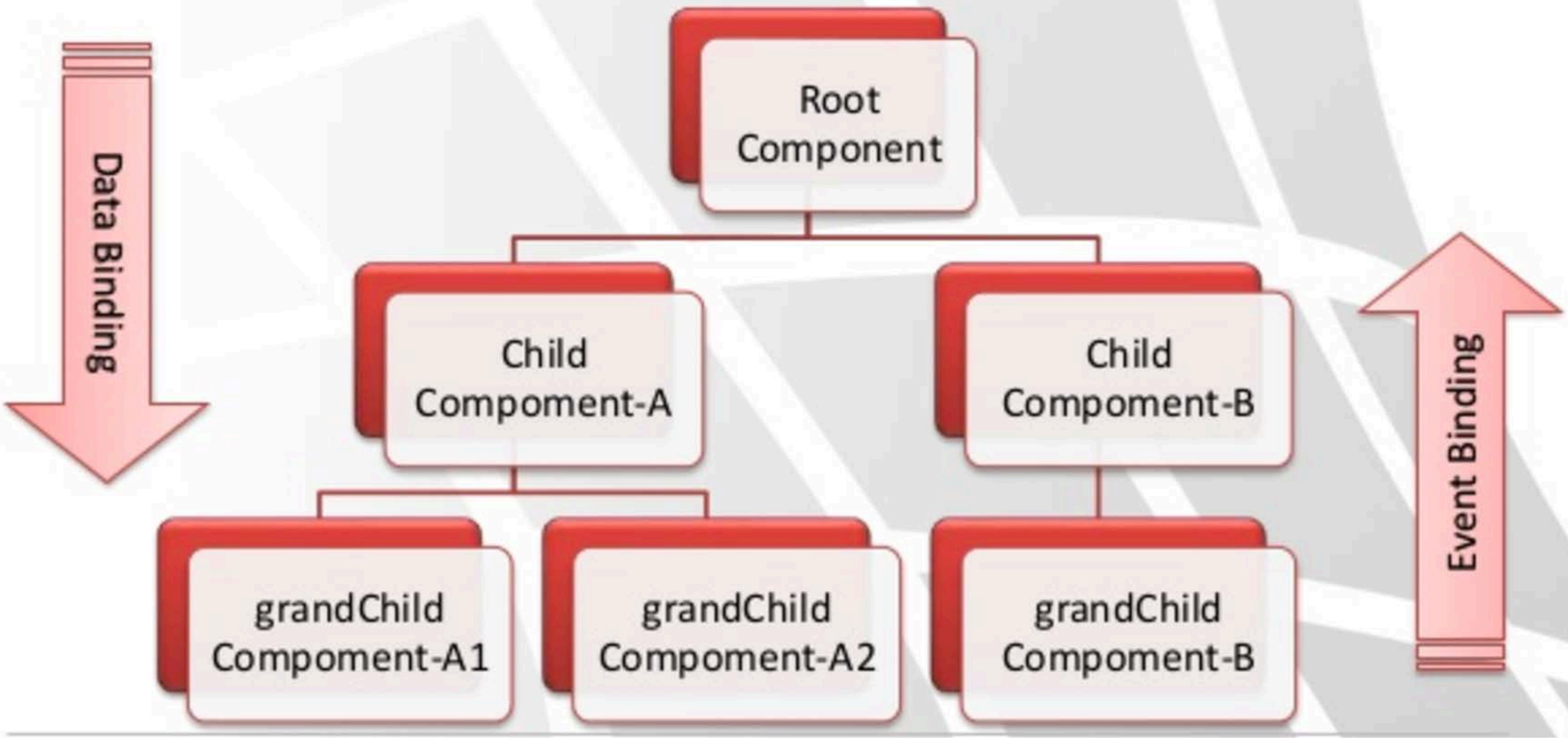
app.ts

```
import {Component} from 'angular2/core';
import {TopNavBar} from './top-navbar';
import {Content}   from './content';

@Component({
  selector: 'app',
  directives: [Content, TopNavBar],
  template: `
    <top-navbar></top-navbar>
    <content></content>
    `
})

export class App {}
```

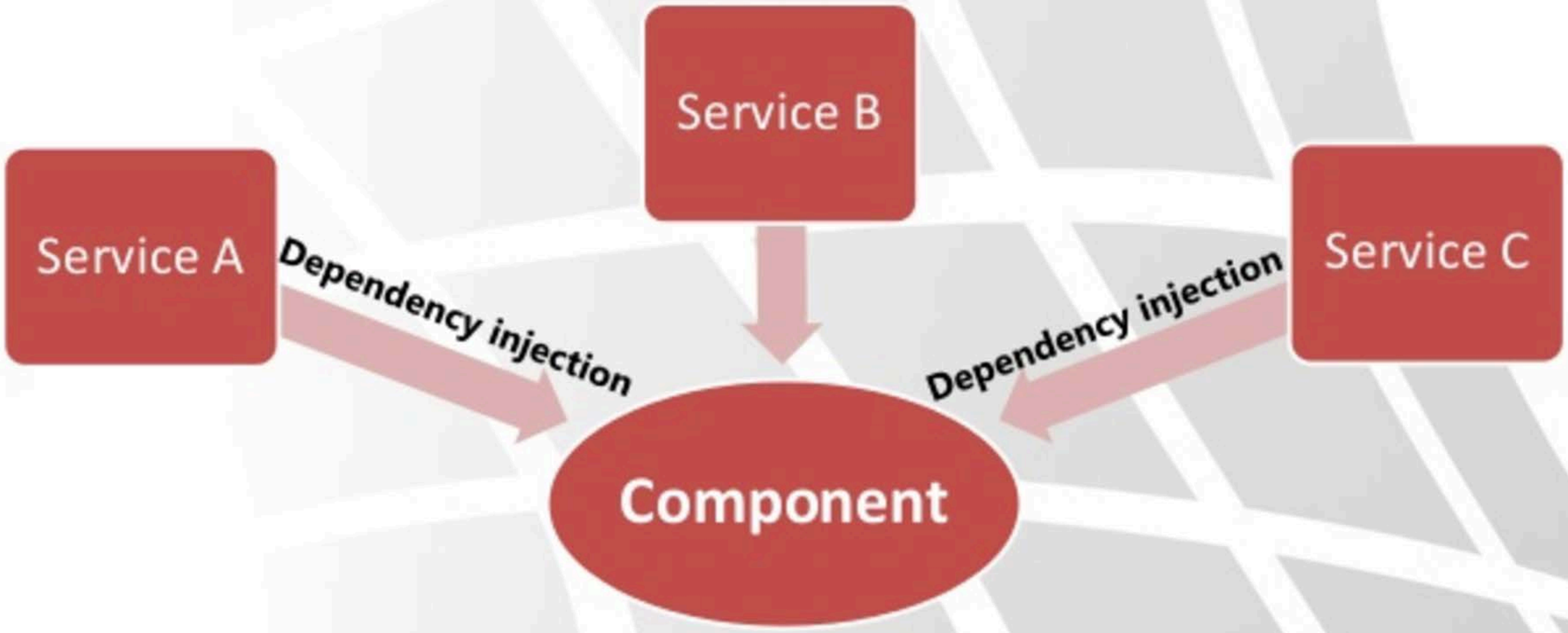
Data is Flowing downwards.



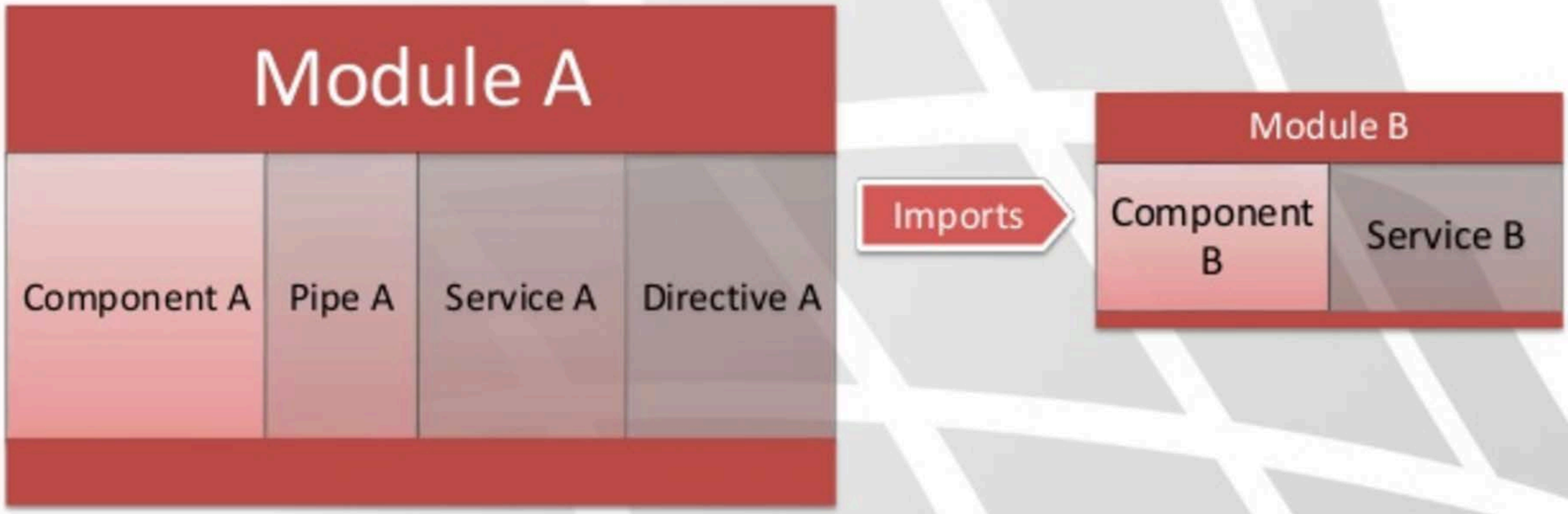
Events are Flowing upwards.

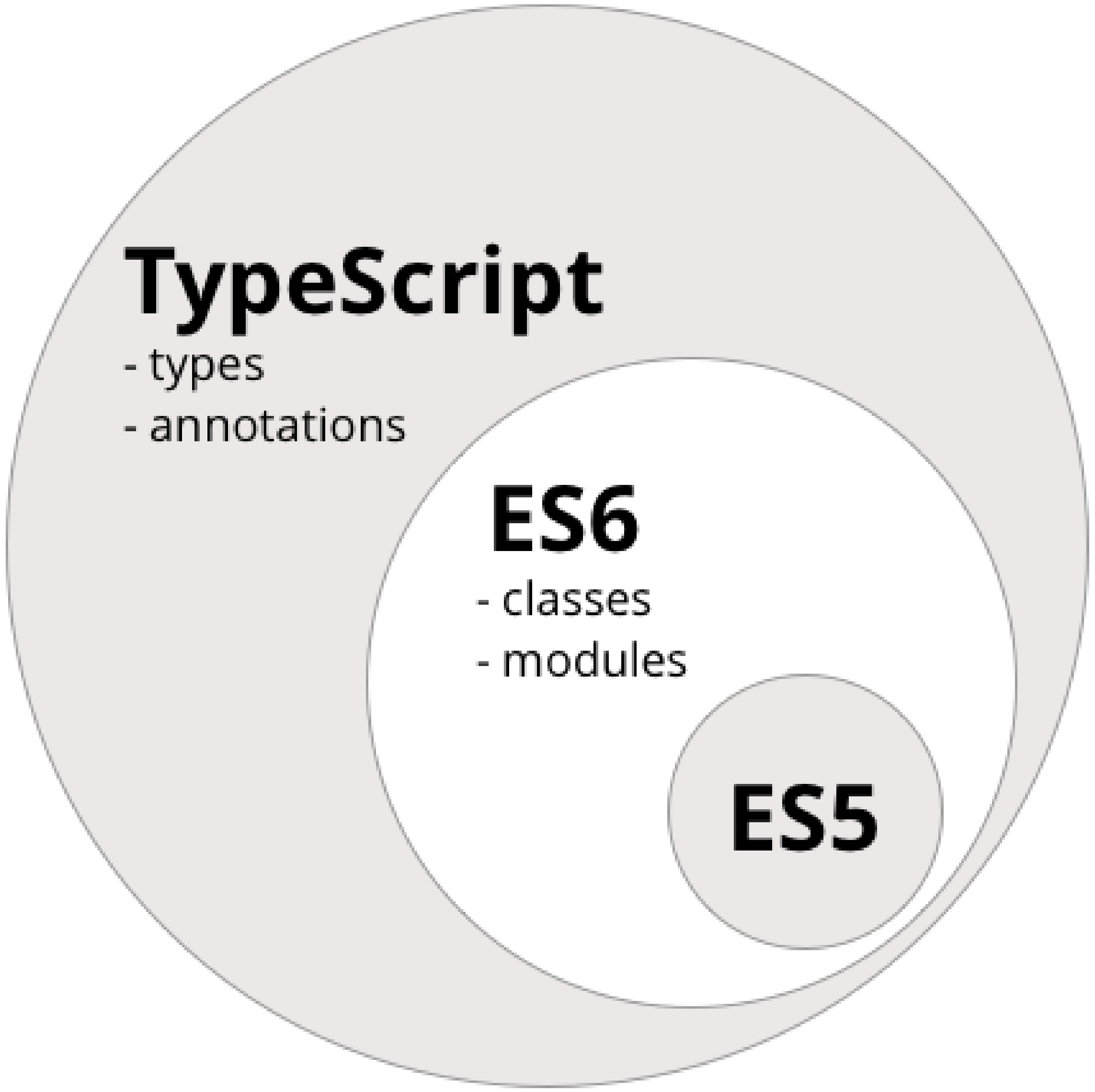


- Each Component Can consume injectable Services.

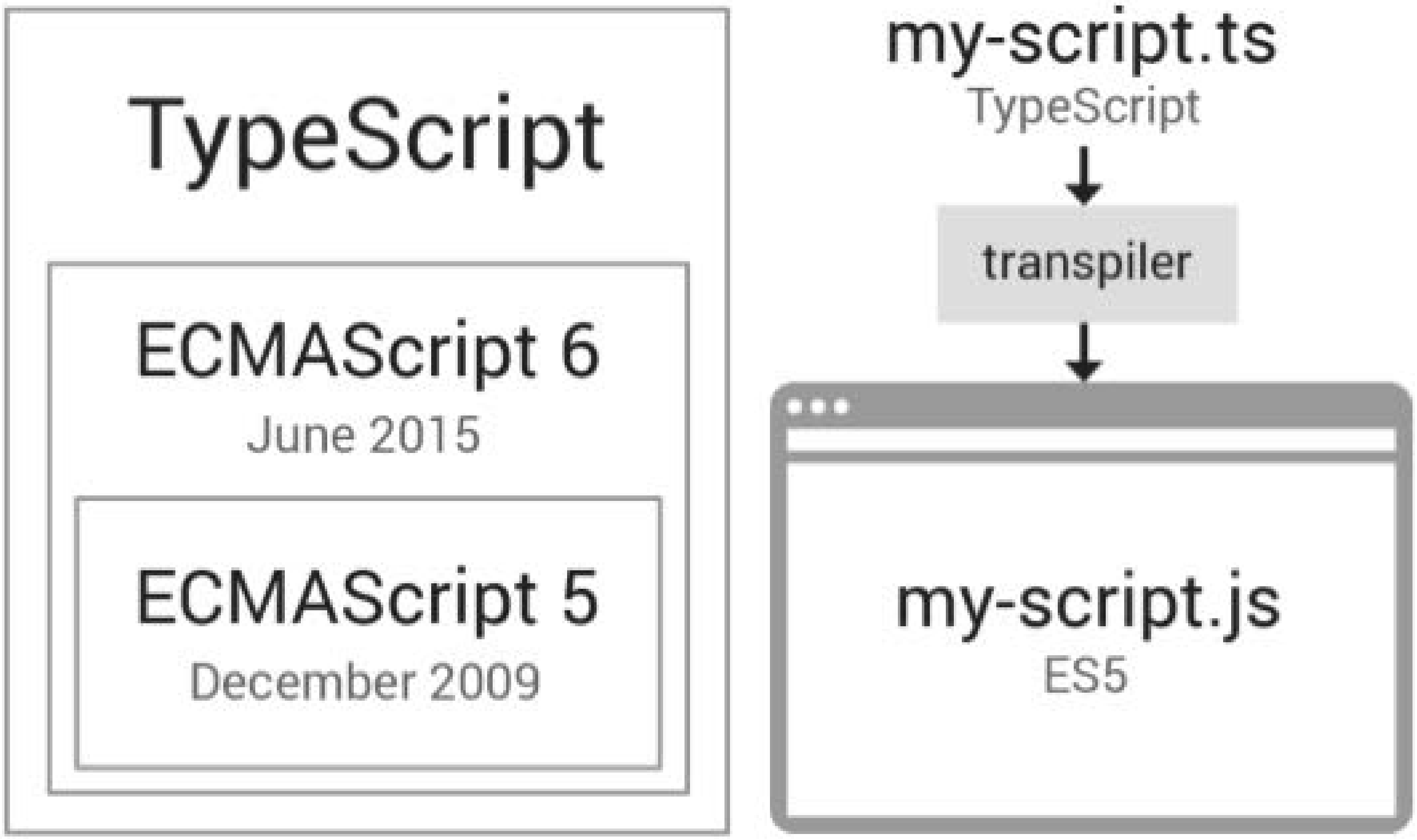


- Components, Services, Directives and Pipes are all defined inside Angular Modules



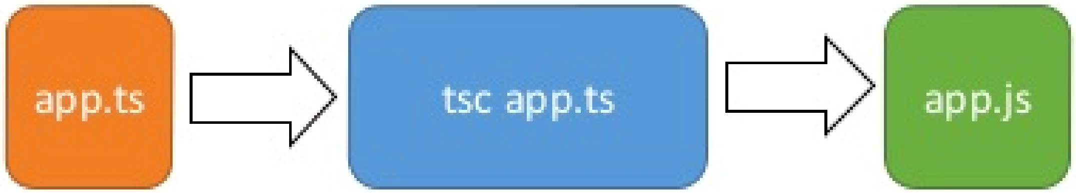

TypeScript
- types
- annotations
ES6
- classes
- modules
ES5

TypeScript


TypeScript
ECMAScript 6
June 2015
ECMAScript 5
December 2009
my-script.ts
TypeScript
transpiler
my-script.js
ES5

TSC - the TypeScript compiler

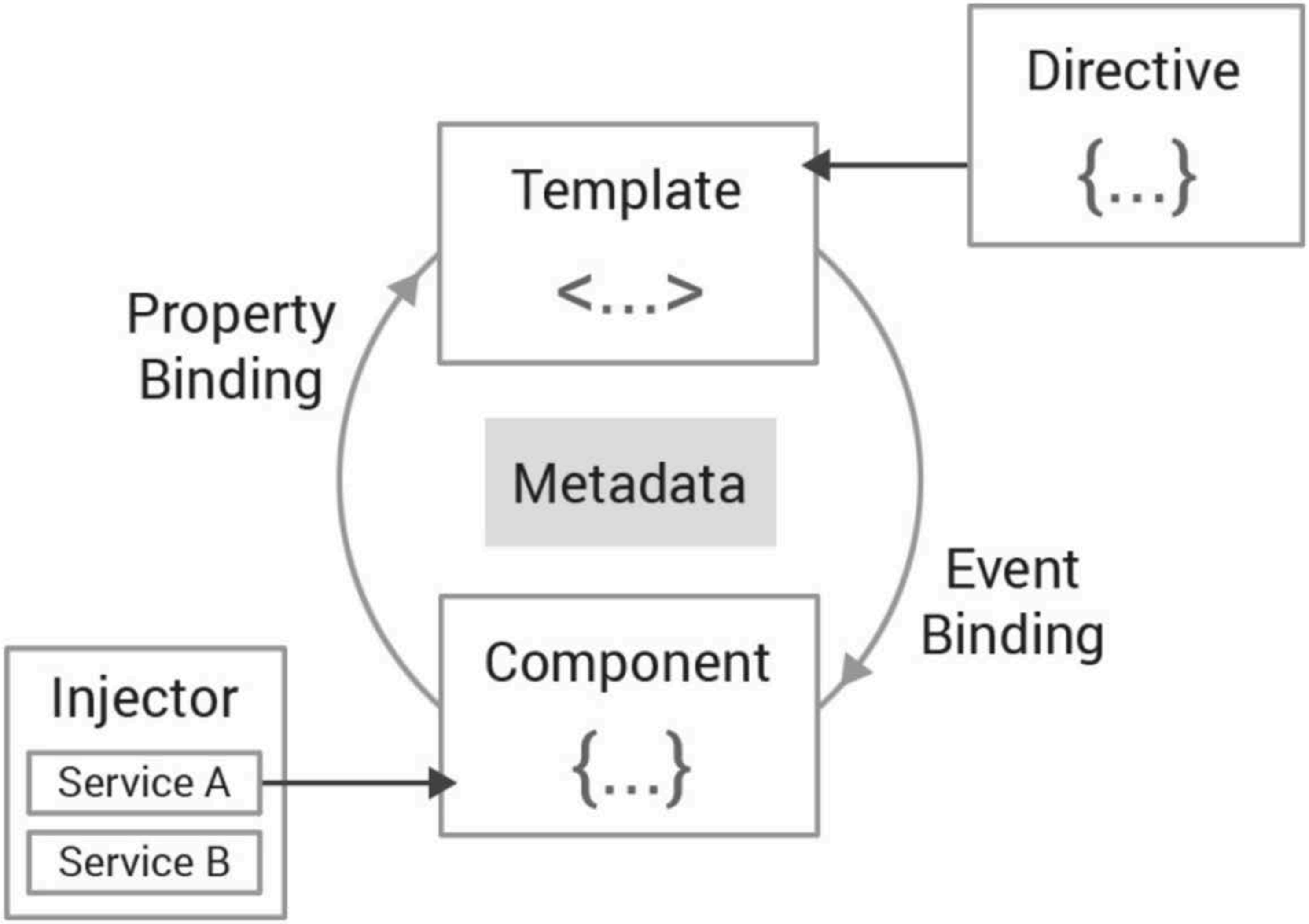
TSC is a source-to-source compiler (a transpiler).



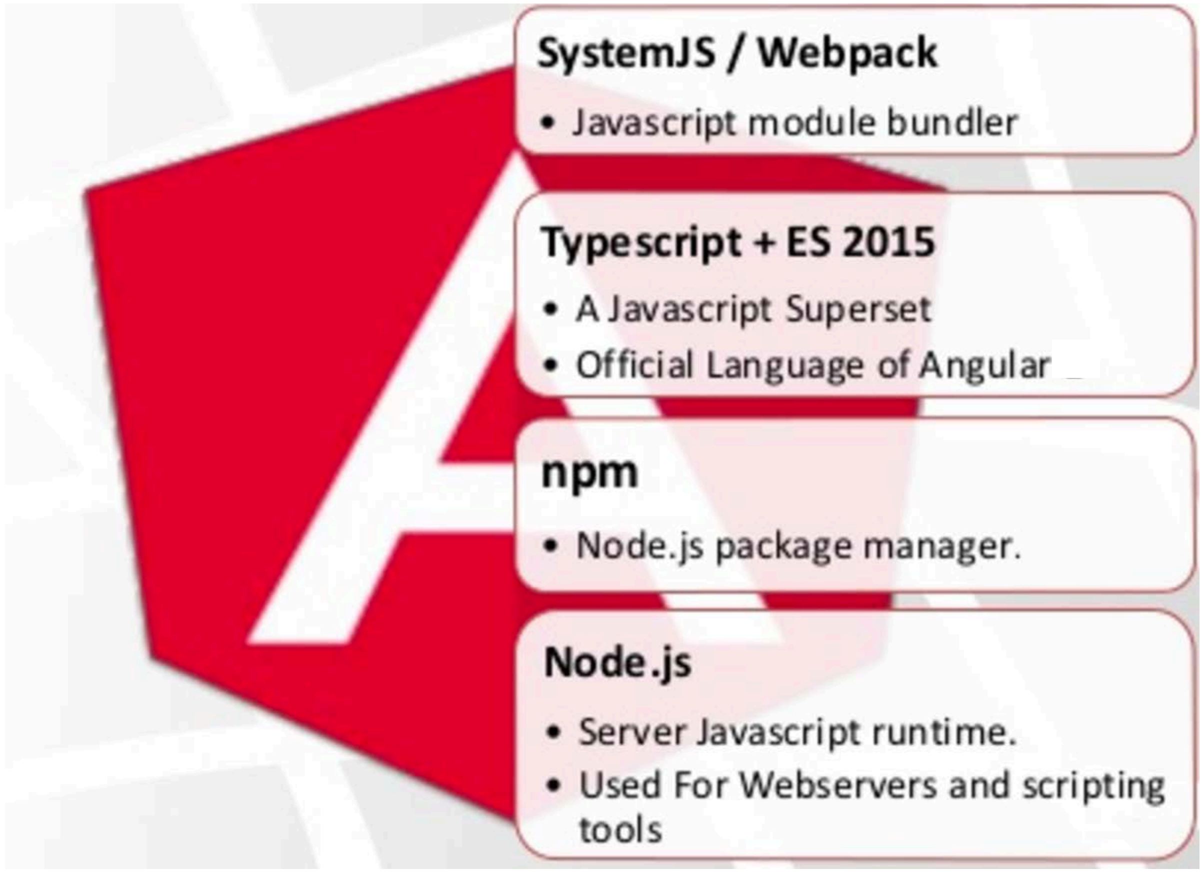
There are lots of options that allow you to:
- concatenate different files in a single output file.
- generate sourcemaps.
- generate module loading code (node.js or require.js).

You can play with the TypeScript playground or setup your environment to see it in action.

Architecture



ANGULAR DEVELOPMENT ENVIRONMENT



SystemJS / Webpack

- Javascript module bundler

Typescript + ES 2015

- A Javascript Superset
- Official Language of Angular

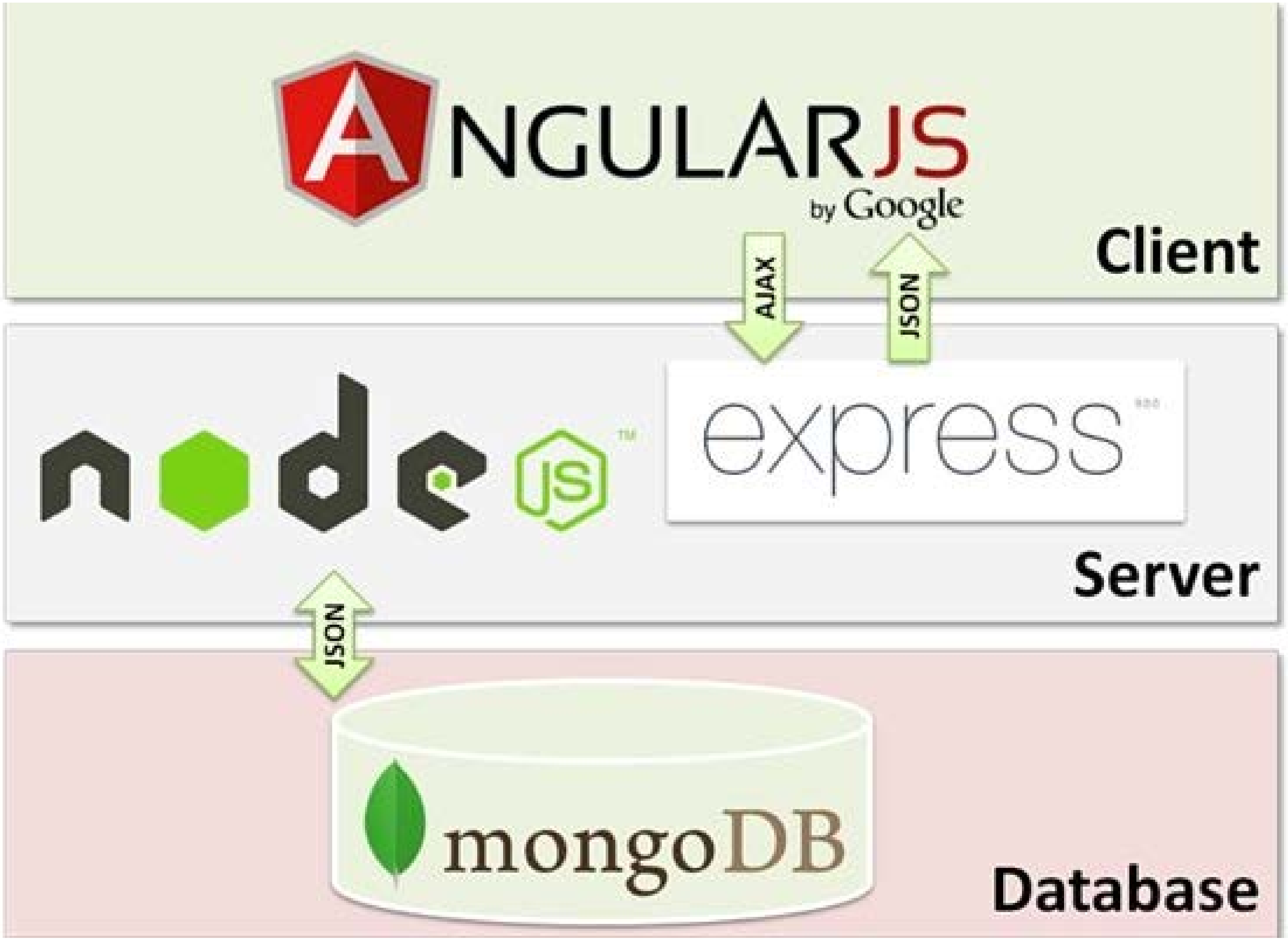
npm

- Node.js package manager.

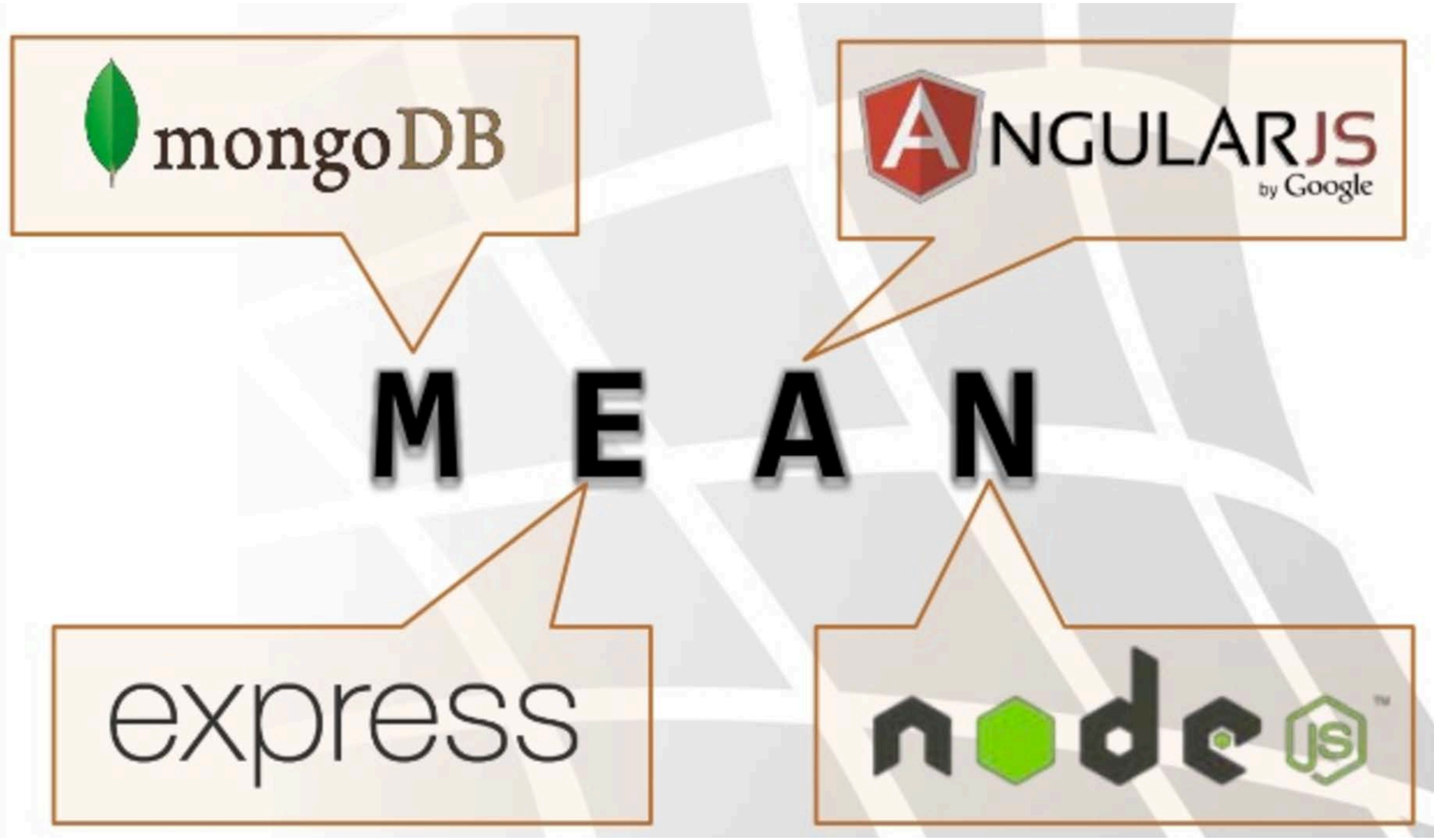
Node.js

- Server Javascript runtime.
- Used For Webservers and scripting tools

MEAN STACK


ANGULARJS by Google
Client
AJAX
JSON
node JS
express
Server
JSON
mongoDB
Database

MEAN STACK



THANK YOU

AND HAVE A GOOD TRAINING!