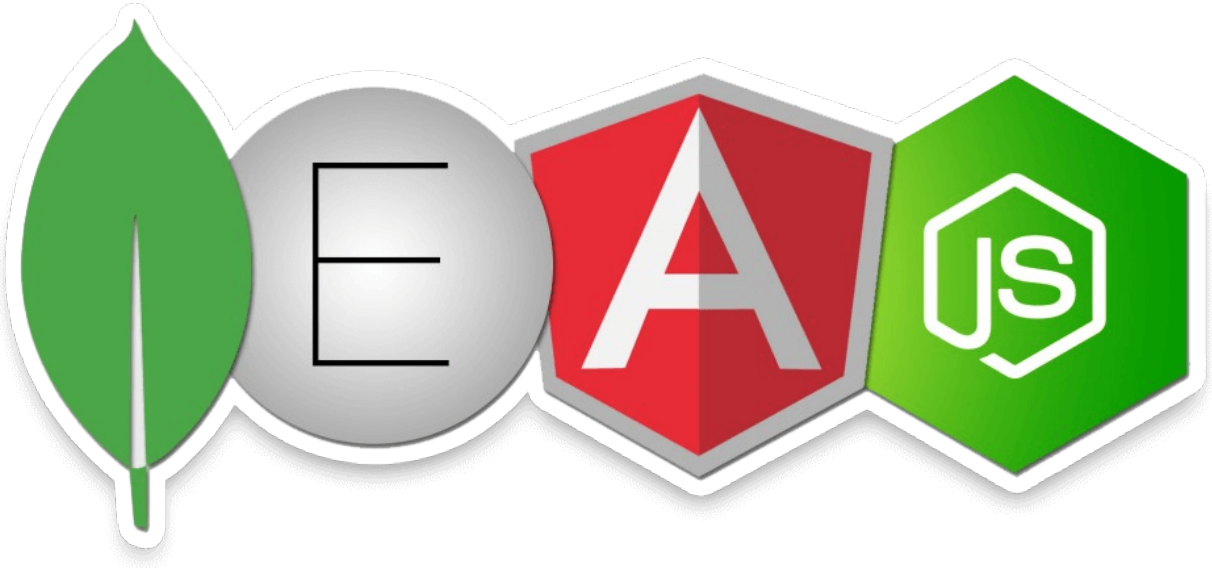
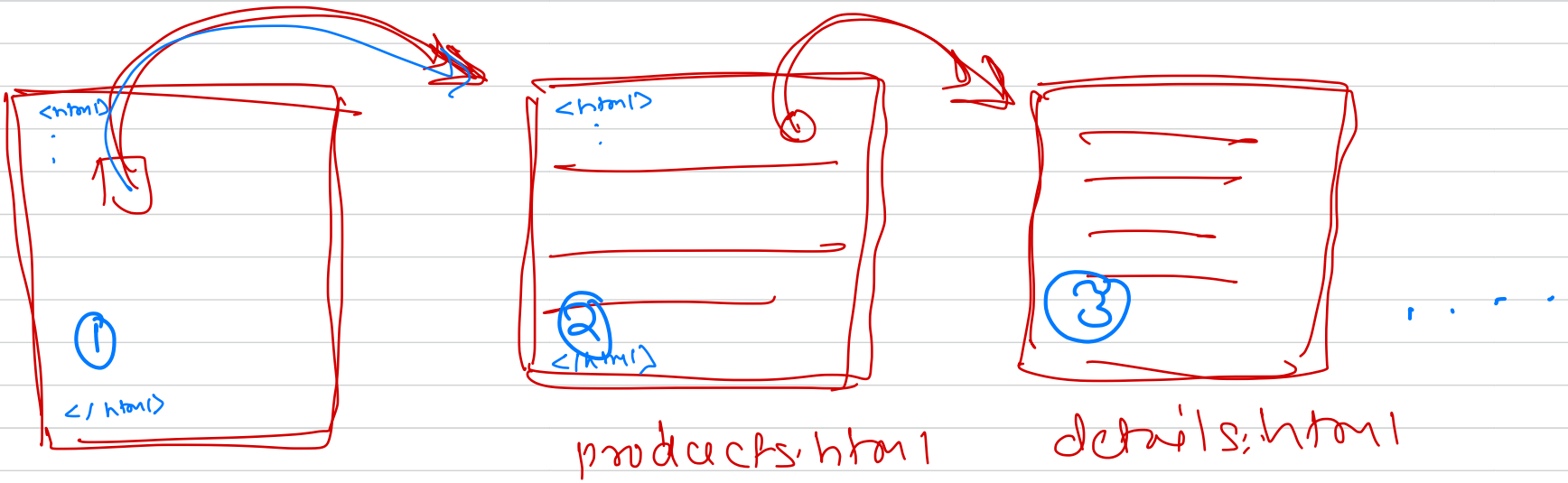
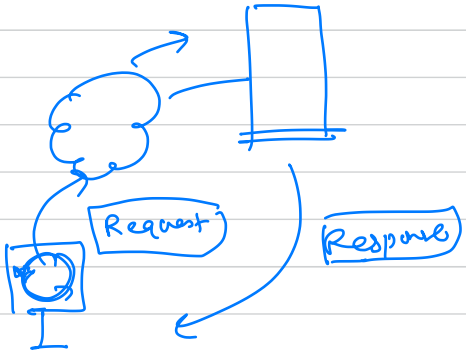
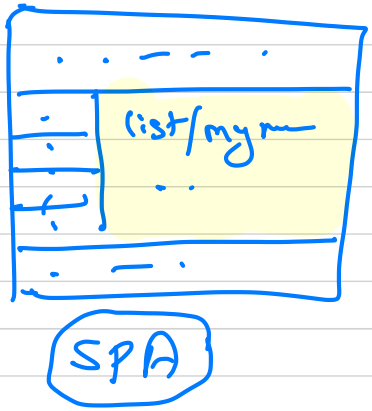

E
A
JS


1
2
3
index.html
products.html
details.html

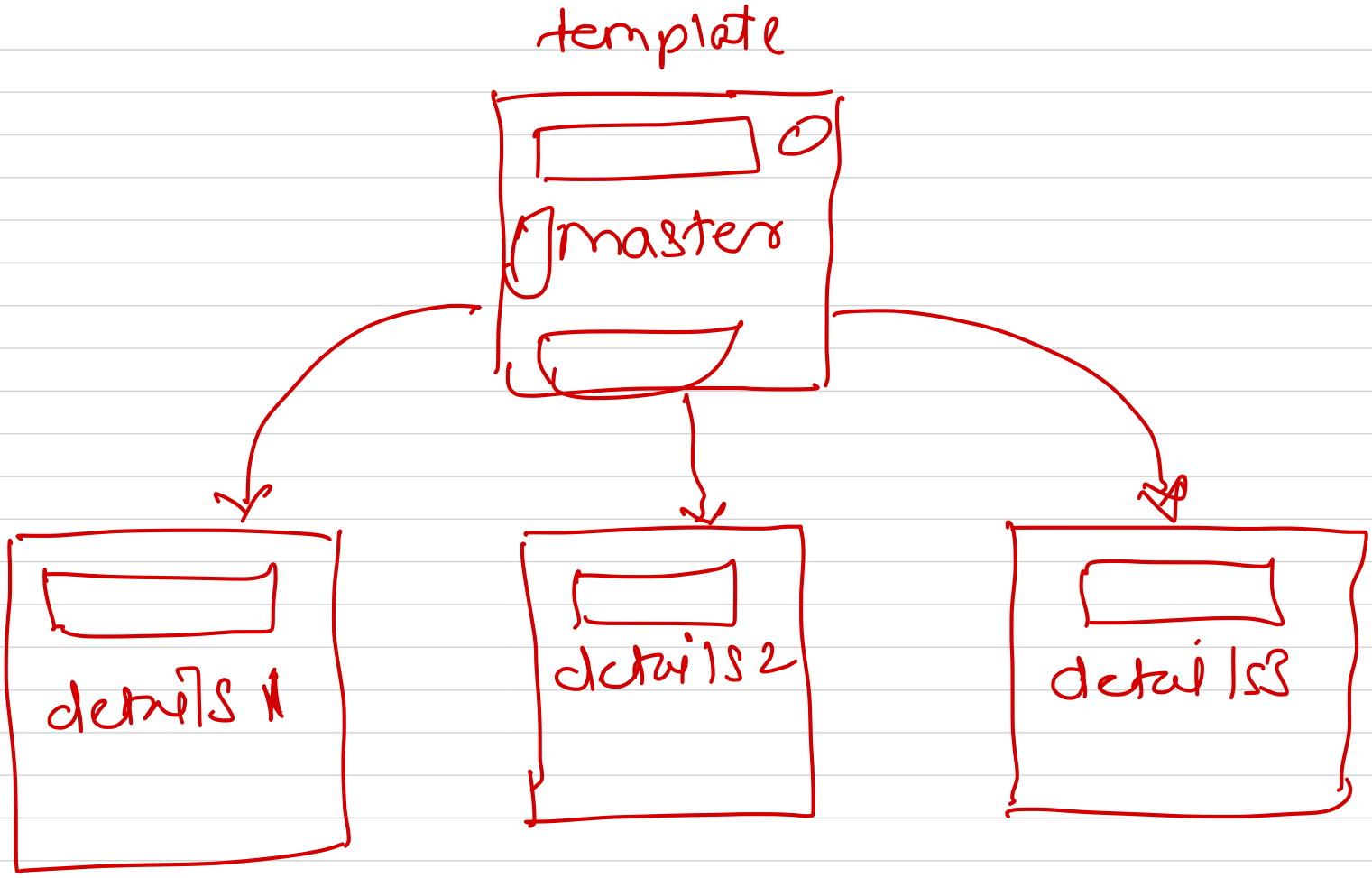
MPA


Request
Respone


list/my

SPA

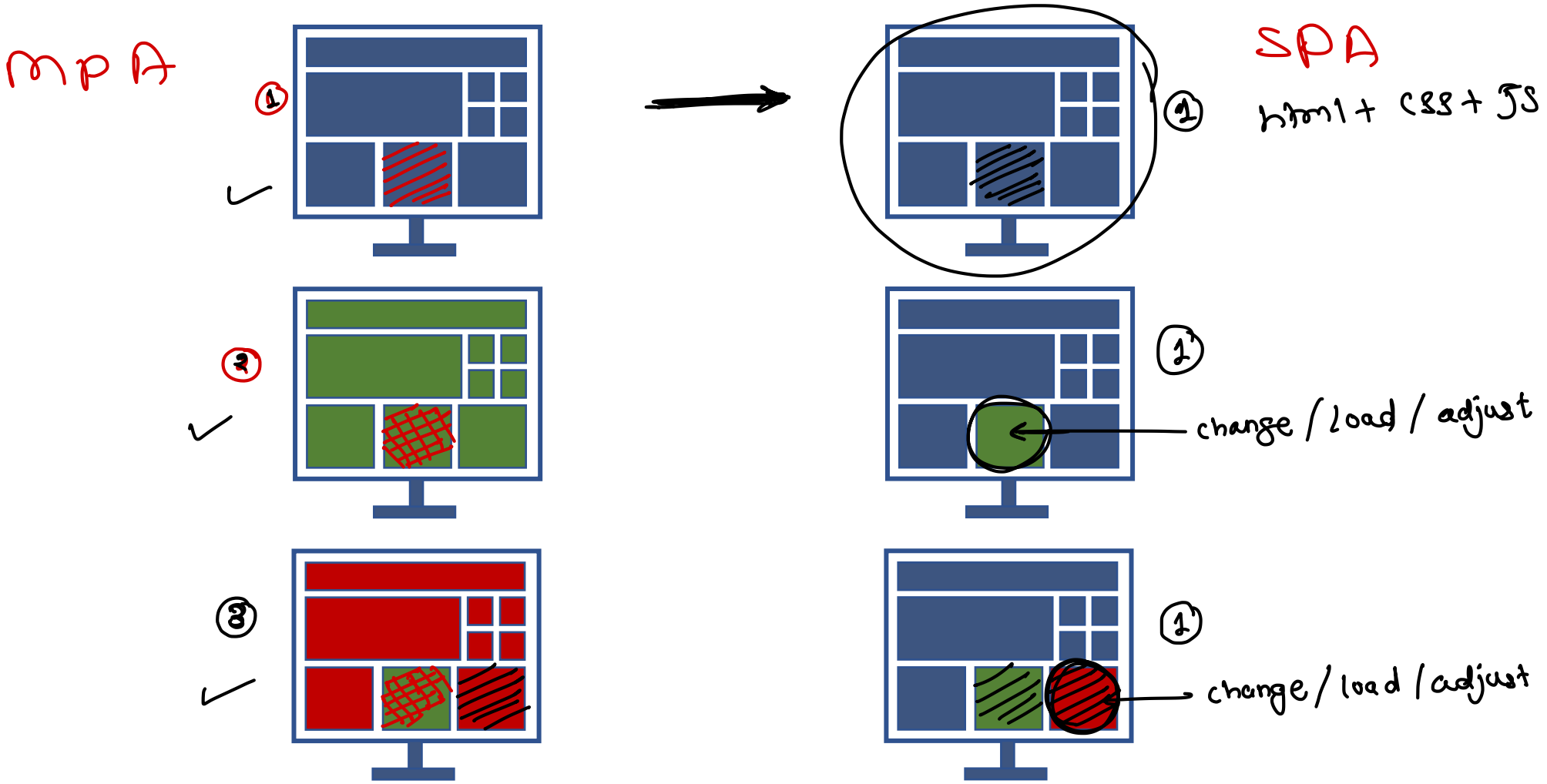
MPA



Single Page Application (SPA)

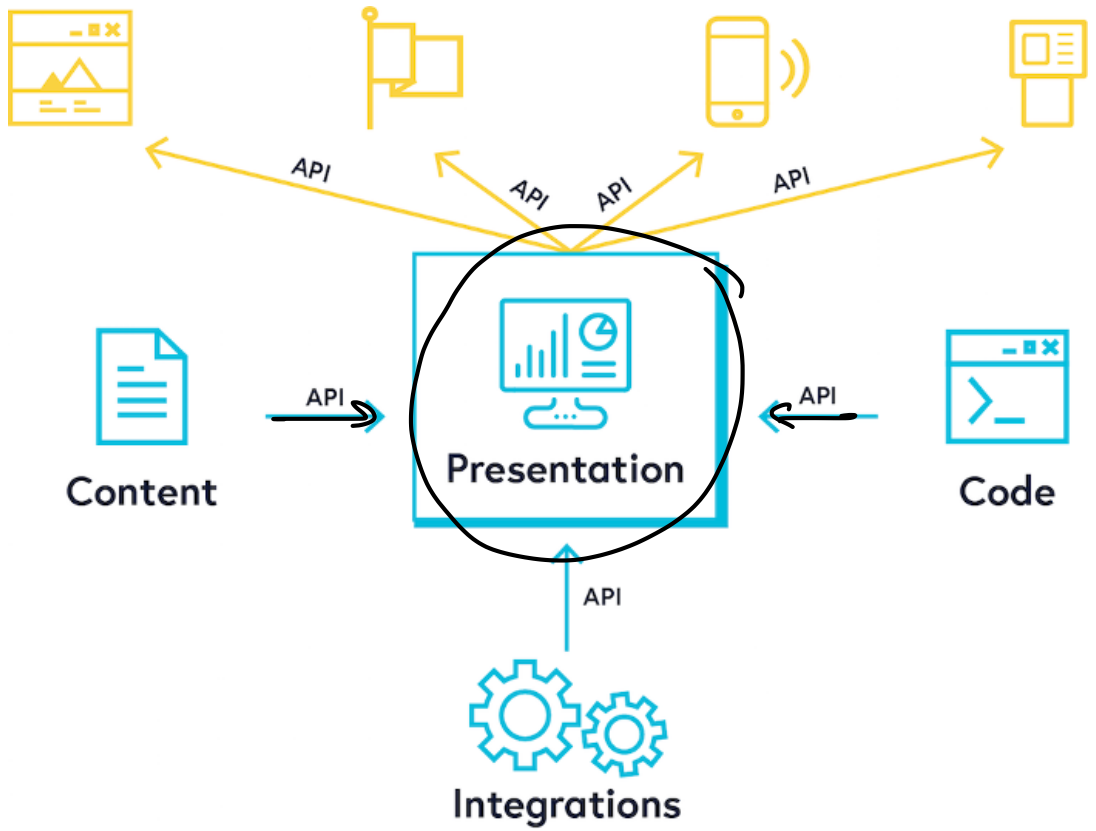
- The single page application is a web application or website that interacts with the user by dynamically rewriting the current page, rather than loading entire new pages from the server
- This approach voids interruption of the user experience between successive pages, making the application behave more like a desktop application
- Some of it stays the same no matter where the user goes (headers, footers, logos, navigation bar, etc), some of it is constant in just a certain section (filter bars, banners), and there are many repeating layouts and templates (blogs, self-service, the google mail setup mentioned above)
- **Single Page Applications take advantage of this repetition**

MPA vs SPA



Advantages of SPA

- Single Time File Load Each of HTML, CSS, JS
- No Extra Queries to Server
- Fast and Responsive Front-end Built
- Enhanced User Experiences



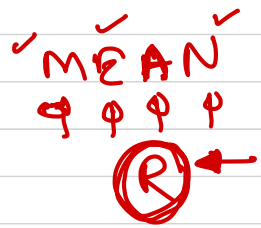
MPA

1. PHP
2. Perl
3. .net
4. Java
5. html + CSS + JS

SPA

1. Angular ← Google
2. React ← facebook
3. Vuejs ← alibaba

MEAN



MPA + SPA

1. React

What is Angular ?

- Angular is a modern web application platform that promises to provide developers with a comprehensive set of tools and capabilities to build large, robust applications
- The core value proposition of Angular is to make it possible to build applications that work for nearly any platform - whether mobile, web, or desktop
- Angular is a platform and framework for building single-page client applications using HTML and TypeScript
- Angular is written in TypeScript
- It implements core and optional functionality as a set of TypeScript libraries that you import into your apps
-

Why to choose Angular?

- Inspired by web standards, enhanced by modern capabilities
- Development tooling included, customizations available
- Powerful ecosystem with a large community
- Sponsored by Google, open source community driven

Setup

① toolchain
↳ Angular CLI
↳ ng

① to create a new project
> ng new <app name>

② run the application
> cd appl
> ng serve

Angular Project Hierarchy

e2e	End-to-end testing folder, contains a basic stub test
node_modules	Standard NPM modules directory, no code should be placed here
src	Source directory for the application
.editorconfig	Editor configuration defaults
.angular.json	Configuration file for the CLI about this project
karma.conf.js	Karma configuration file for unit test runner
package.json	Standard NPM package manifest file
protractor.conf.js	Protractor configuration file for e2e test runner
README.md	Standard readme file, contains starter information
tsconfig.json	Default configuration file for TypeScript compiler
tslint.json	TSLint configuration file for TypeScript linting rules

Angular Project Hierarchy

app	Contains the primary App component and module
assets	Empty directory to store static assets like images
environments	Environment configurations to allow you to build for different targets, like dev or production
favicon.ico	Image displayed as browser favorite icon
index.html	Root HTML file for the application
main.ts	Entry point for the web application code
polyfills.ts	Imports some common polyfills required to run Angular properly on some browsers
styles.css	Global stylesheet
test.ts	Unit test entry point, not part of application
tsconfig.app.json	TypeScript compiler configuration for apps
tsconfig.spec.json	TypeScript compiler configuration for unit tests
typings.d.ts	Typings configuration

Booting

① Reads angular.json
- → main : src/main.ts :→ bootstraping / startup / initialization code
- → index : src/index.html :→ User interface

② starts executing main.ts
- → gets browser object
- → bootstrapModule (AppModule) : loads a module named AppModule
  - NgModule [collection of different parts of app]
- → starts catching errors (if any)

③ AppModule starts loading
- → loads metadata of module
- → loads AppComponent

④ AppComponent loads metadata
- → selector : app-root
- → templateUrl : app.component.html
- → styleUrls : app.component.css

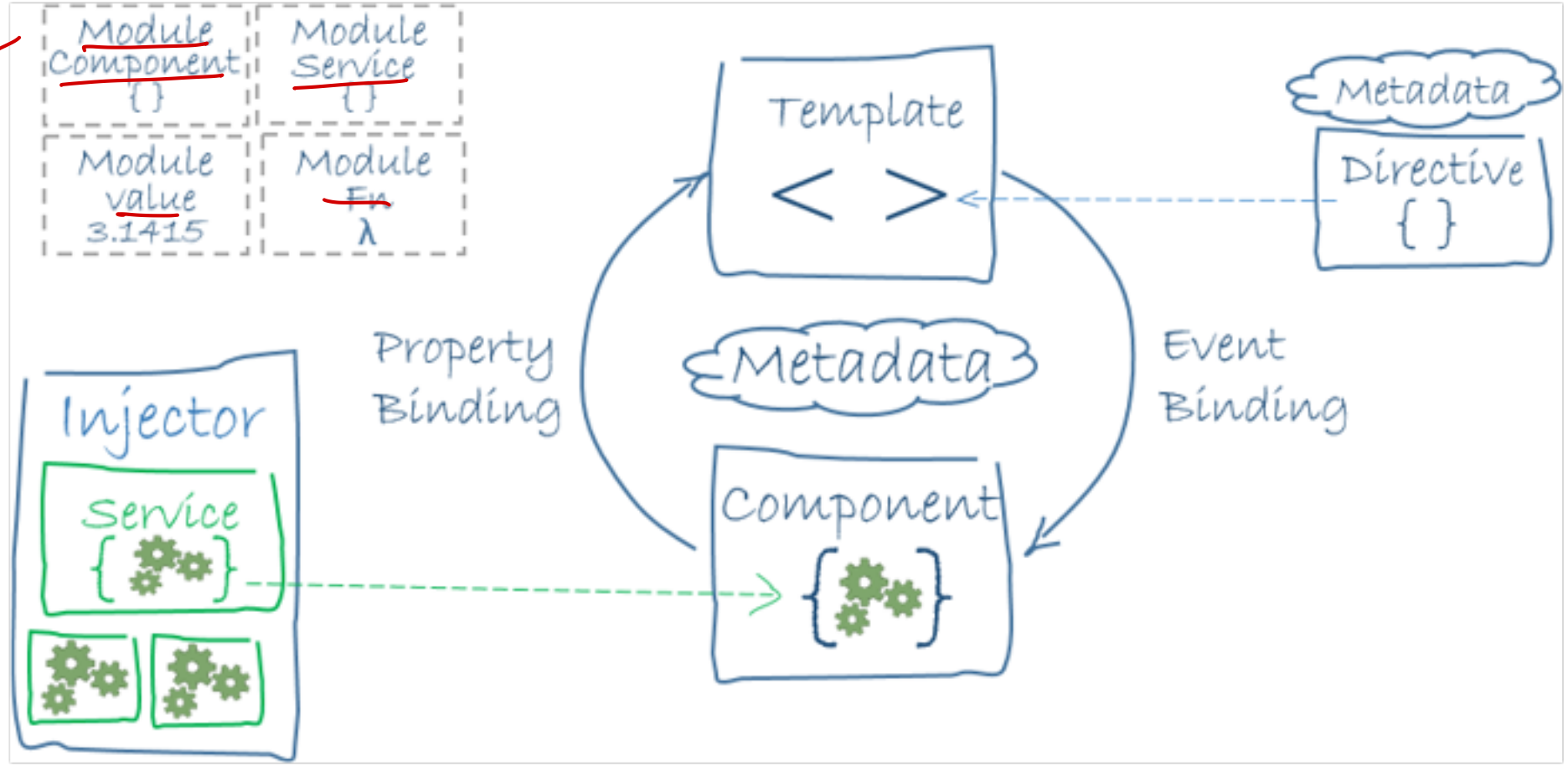
→

Selector	Component	template	styles
app-root	AppComponent	app.component.html	app.component.css
app-first	FirstComponent	first.comp...html	first.com...css

table

⑤ loads index.html

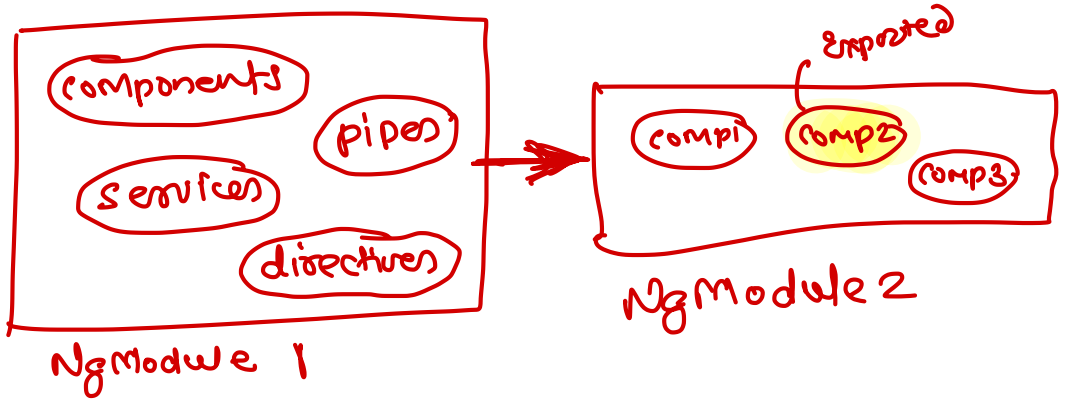
Basic Concepts

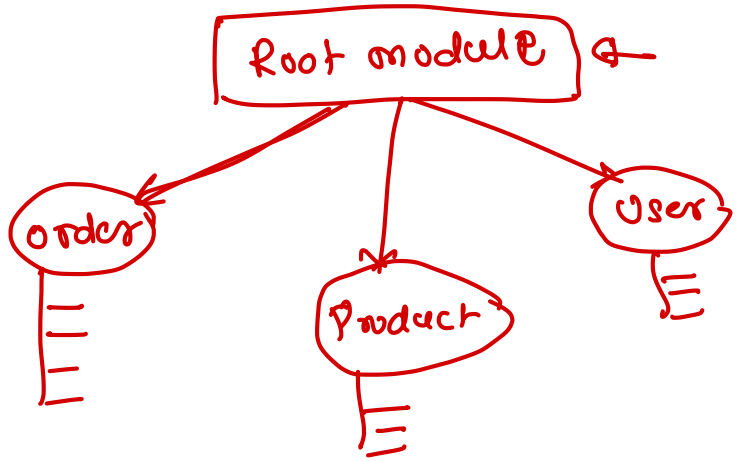


Module

Introduction

- Angular apps are modular and Angular has its own modularity system called *NgModule*
- NgModules are containers for a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities
- They can contain components, service providers, and other code files whose scope is defined by the containing NgModule
- They can import functionality that is exported from other NgModules, and export selected functionality for use by other NgModules





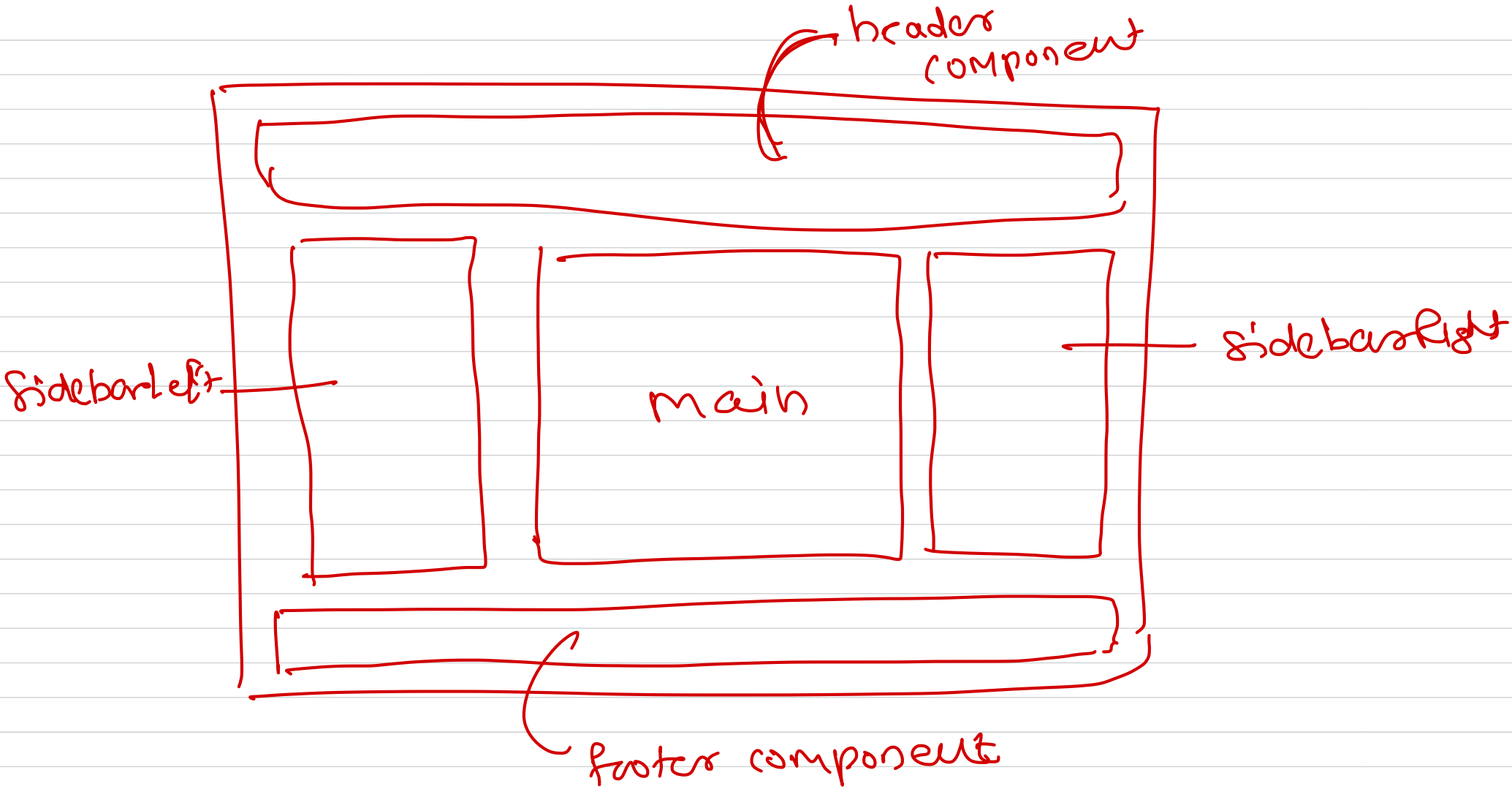
NgModule metadata

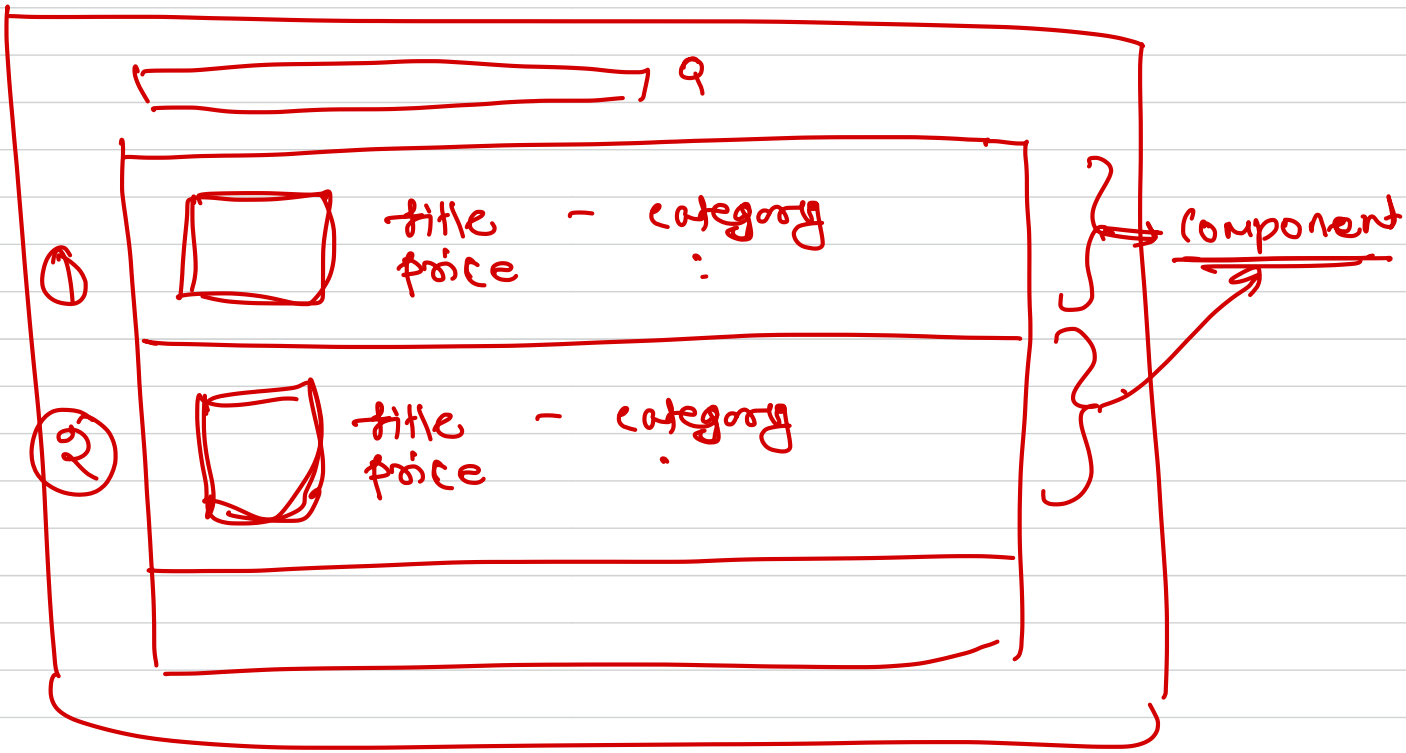
- An NgModule is defined by a class decorated with @NgModule()
- The @NgModule() decorator is a function that takes a single metadata object, whose properties describe the module
- The most important properties are as follows.
  - **declarations**:
    - The components, *directives*, and *pipes* that belong to this NgModule
  - **exports**:
    - The subset of declarations that should be visible and usable in the *component templates* of other NgModules
  - **imports**:
    - Other modules whose exported classes are needed by component templates declared in *this* NgModule
  - **providers**:
    - Creators of services that this NgModule contributes to the global collection of services; they become accessible in all parts of the app(You can also specify providers at the component level)
  - **bootstrap**:
    - The main application view, called the *root component*, which hosts all other app views. Only the *root NgModule* should set the bootstrap property

NgModules and JavaScript modules

- The NgModule system is different from and unrelated to the JavaScript (ES2015) module system for managing collections of JavaScript objects
- These are *complementary* module systems that you can use together to write your apps
- In JavaScript each *file* is a module and all objects defined in the file belong to that module
- The module declares some objects to be public by marking them with the export key word
- Other JavaScript modules use *import statements* to access public objects from other modules

Components


header component
sidebarleft
main
sidebarright
footer component

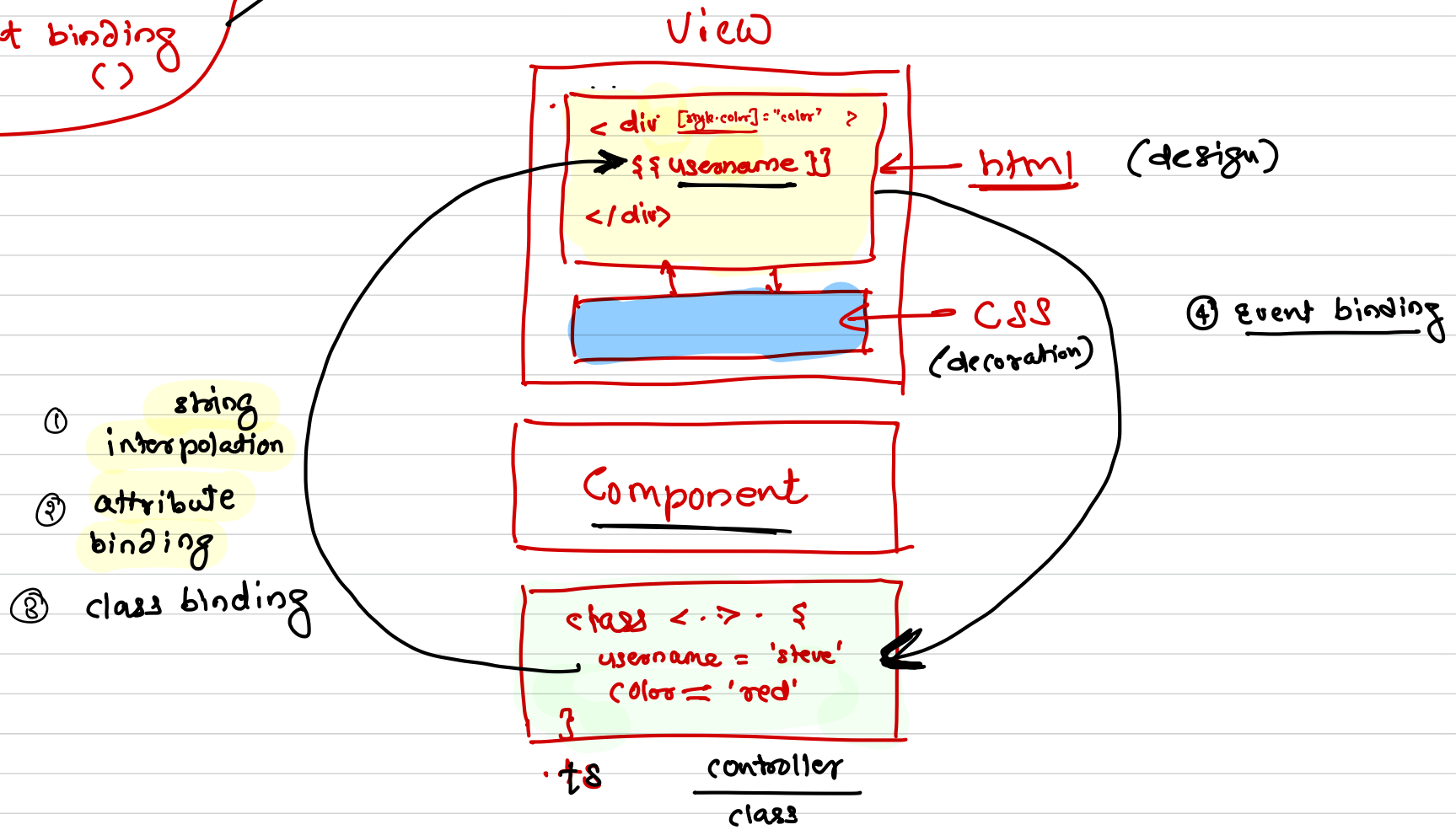

q
title - category
price
1
title - category
price
2
component

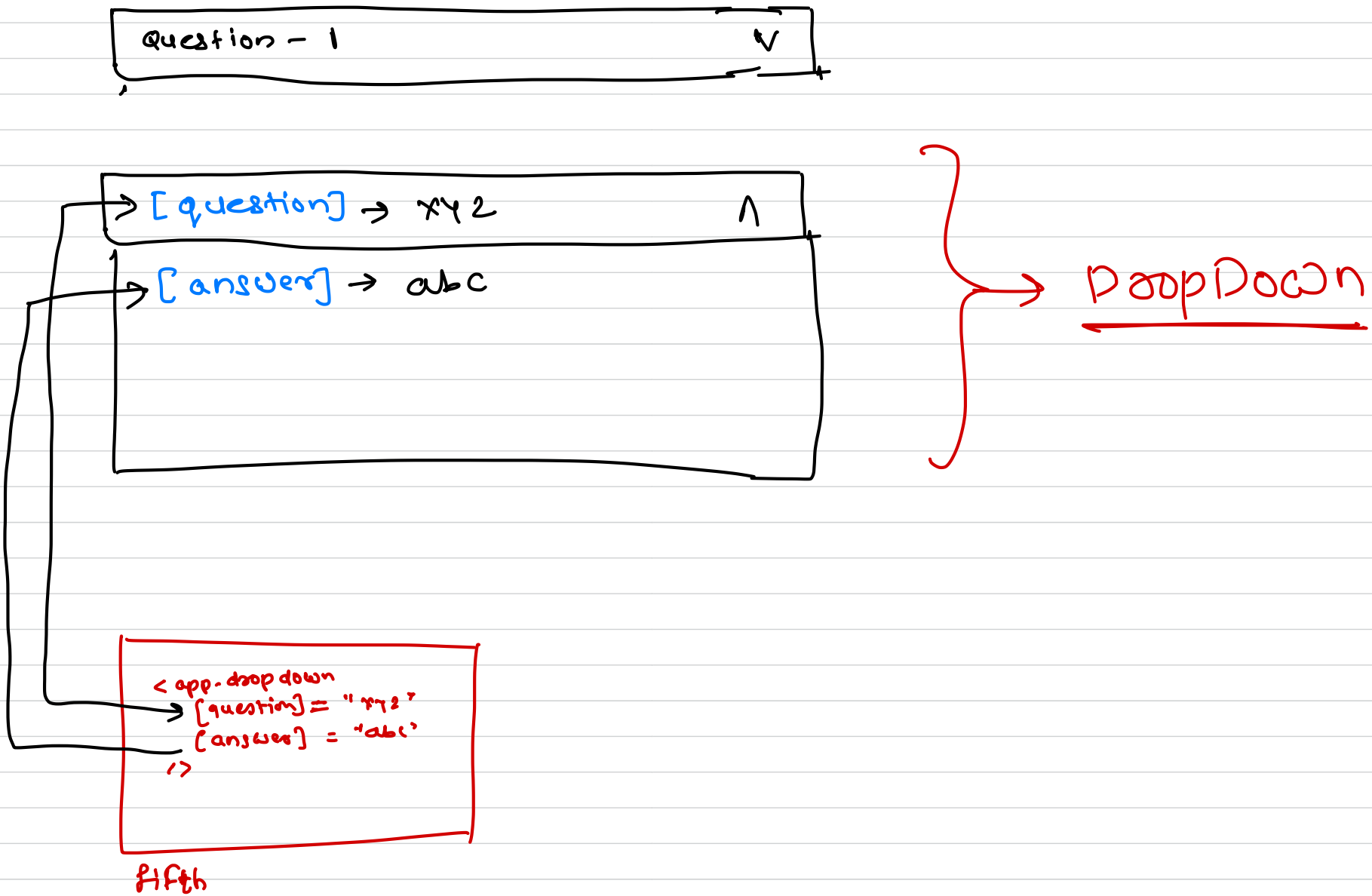
Introduction

- Components are the main building block for Angular applications
- Each component consists of:
  - An HTML template that declares what renders on the page
  - A Typescript class that defines behavior
  - A CSS selector that defines how the component is used in a template
  - Optionally, CSS styles applied to the template
- To create a component using the Angular CLI
  - From a terminal window, navigate to the directory containing your application
  - Run the ng generate component <component-name> command,
    - where <component-name> is the name of your new component

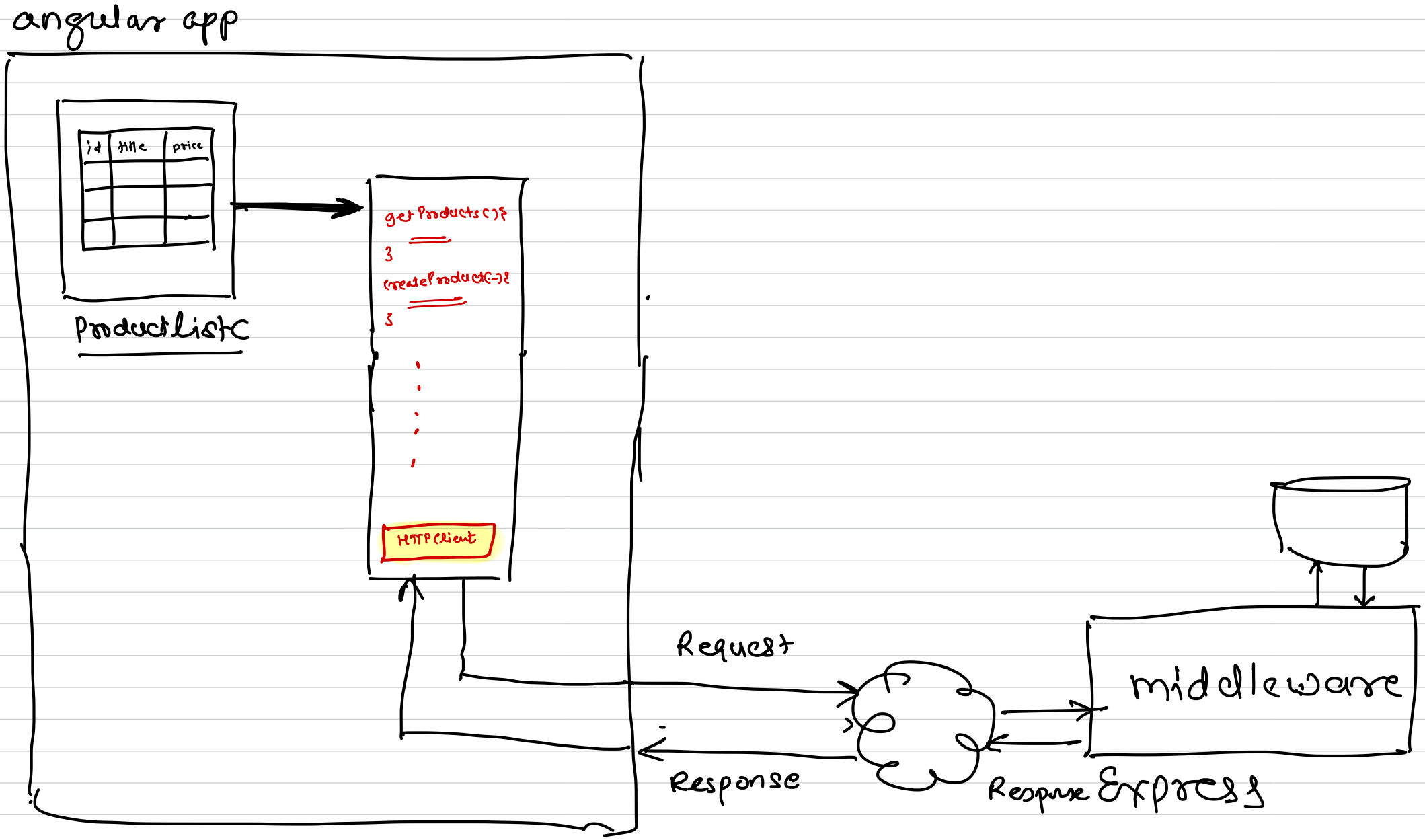
Binding

- string interpolation {{ }}
- attribute binding [ ]
- class binding [ ]
- event binding ( )




Question - 1
[question] → xyz
[answer] → abc
DropDown
<app-dropdown
[question] = "xyz"
[answer] = "abc"
/>

Fifth


angular app
id
title
price
ProductListC
getProducts(){
}
createProduct(){
}
HTTPClient
Request
Response
middleware
Respnse Express