

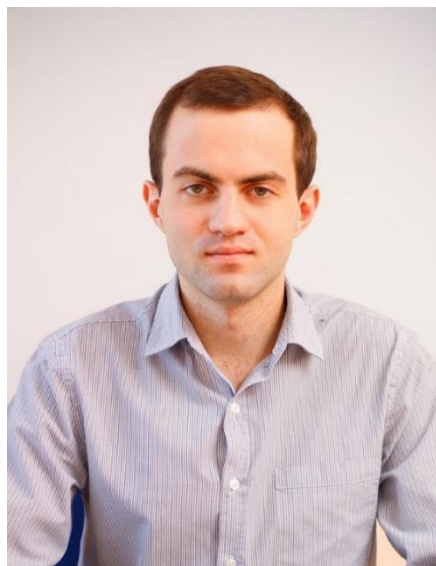


# Angular 2 Essential

Введение в Angular 2

# Angular 2 Essential

## Introduction



Охрименко Дмитрий  
MCT

 \_okhrimenko

 dmitriy.okhrimenko

 <https://goo.gl/eeTdMv>



MCID: 9210561

# Angular 2 Essential

Тема урока

Введение в Angular 2

# Angular 2 Essential

## План урока

- Angular 1.x и Angular 2
- Инструменты для разработки
- Hello World приложение
- Структура проекта
- Введение в TypeScript
  - Типы
  - Классы
  - Интерфейсы
  - Декораторы
- Общая архитектура Angular 2

# Angular 2 Essential

## Особенности Angular 2

- **Более простой в освоении**
  - Современный подход
  - Хорошо налаженный фреймворк
- **TypeScript**
  - Типизация
  - Удобные инструменты для разработки
- **Некоторое сходство с Angular 1.x**
  - Angular 2 полностью переписан но имеет сходства с предыдущей версией
- **Производительность**
  - Разработан для мобильных устройств
  - Улучшенная производительность
  - Интеграция с NativeScript
- **Архитектура и легкость сопровождения**
  - ES2015 module system



*Построен на основе отзывов сообщества разработчиков, собранных в течении 5-ти лет*

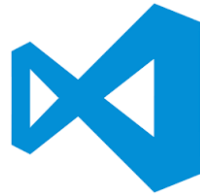
# Angular 2 Essential

## Отличия Angular 1 и Angular 2

	Angular 1.x	Angular 1.x Best Practice	Переходная архитектура	Angular 2
Вложенные scope (\$scope, watchers)	Активно используются	Рекомендуется избегать	Рекомендуется избегать	Нет
Directives vs Controllers	Используются как альтернативы	Используются вместе	Использование компонентов	Компоненты
Реализация контроллеров и сервисов	Function	Function	ES6 class	ES6 class
Модули	Angular's module	Angular's modules	ES6 modules	ES6 modules
Transpiler	Нет	Нет	TypeScript	TypeScript

# Angular 2 Essential

## IDE



**Visual Studio Code**

<http://code.visualstudio.com/>



**IntelliJ IDEA**

<http://code.visualstudio.com/>



<https://www.jetbrains.com/webstorm/>

# Angular 2 Essential

## NodeJS и npm

Для работы примеров необходимо установить **Node** v 4.x.x или выше и также **npm** v 3.x.x или выше.



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

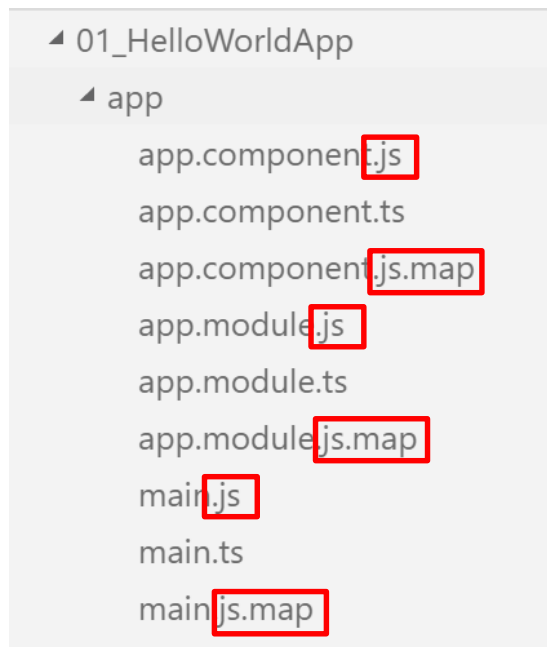
Инструкции по установке <http://blog.npmjs.org/post/85484771375/how-to-install-npm>



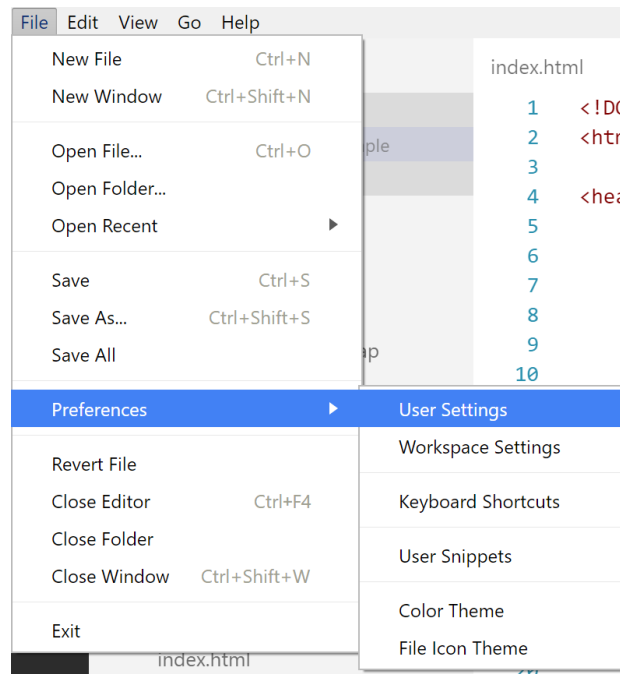
# Angular 2 Essential

## Скрытие .map и .js файлов в VS Code

1



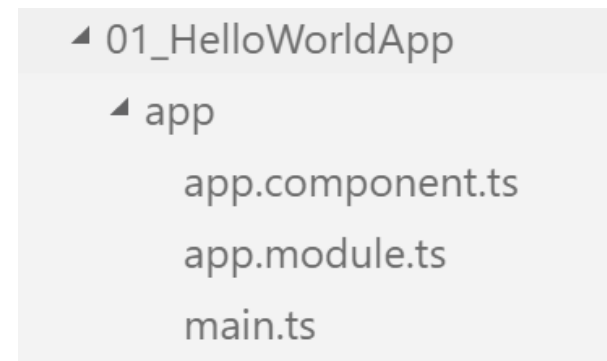
2



3

```
{
  "files.exclude": {
    "**/.git": true,
    "**/.DS_Store": true,
    "**/*.js.map": true,
    "**/*.js": {
      "when": "$(basename).ts"
    }
  }
}
```

4



Код для настроек редактора можно найти в описании к уроку (ст. 4)

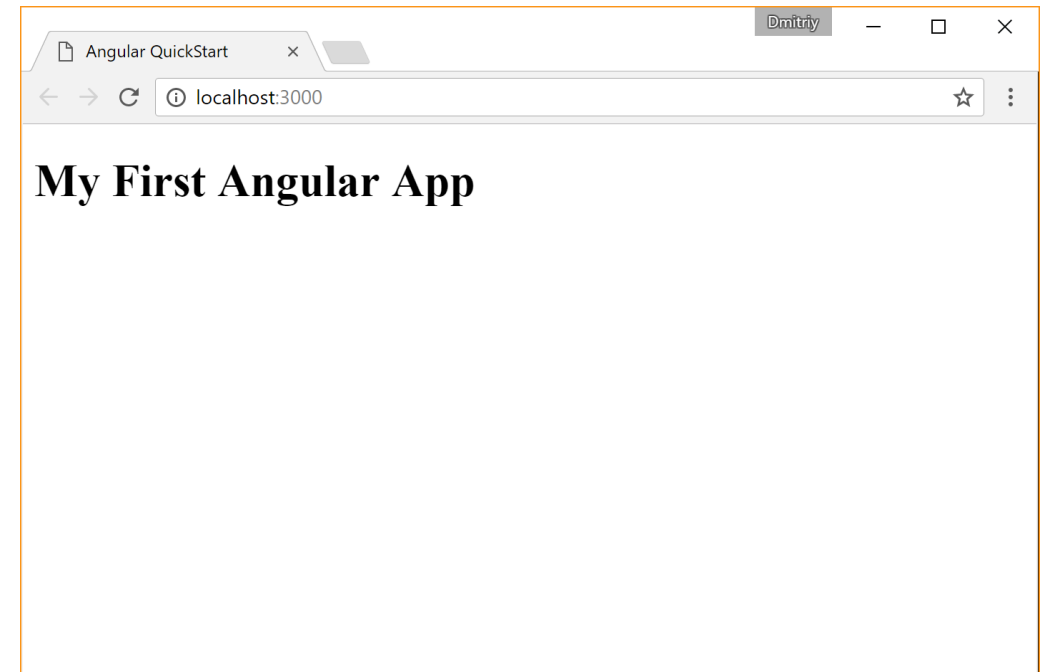
# Angular 2 Essential

## Запуск lite-server

Установка: **npm install lite-server -g**

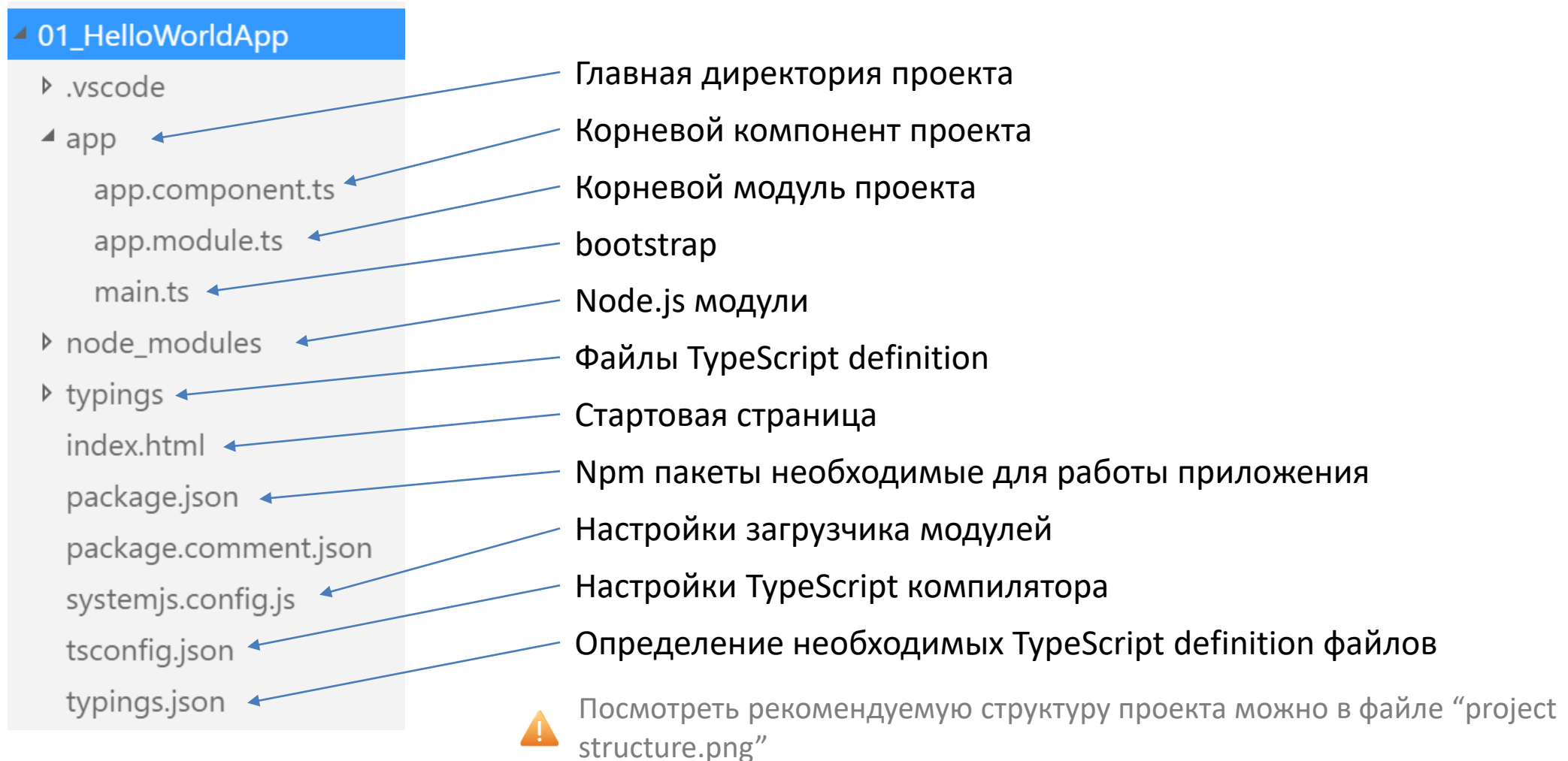
Запуск: **lite-server** (запуск сервера для текущей директории, порт 3000)

```
lite-server
d:\Lessons_Development\Angular2 Essential\Lessons\01_Introduction\01_HelloWorldApp>lite-server
Did not detect a `bs-config.json` or `bs-config.js` override file. Using lite-server defaults...
** browser-sync config **
{ injectChanges: false,
  files: [ './**/*.html,css,js' ],
  watchOptions: { ignored: 'node_modules' },
  server: { baseDir: './', middleware: [ [Function], [Function] ] } }
[BS] Access URLs:
-----
Local: http://localhost:3000
External: http://192.168.88.19:3000
-----
UI: http://localhost:3001
UI External: http://192.168.88.19:3001
-----
[BS] Serving files from: ./
[BS] Watching files...
16.10.07 12:49:57 304 GET /index.html
16.10.07 12:49:57 304 GET /node_modules/core-js/client/shim.min.js
16.10.07 12:49:57 304 GET /node_modules/zone.js/dist/zone.js
16.10.07 12:49:57 304 GET /node_modules/reflect-metadata/Reflect.js
16.10.07 12:49:57 304 GET /node_modules/systemjs/dist/system.src.js
16.10.07 12:49:57 304 GET /systemjs.config.js
16.10.07 12:49:57 304 GET /app/main.js
16.10.07 12:49:57 304 GET /node_modules/@angular/platform-browser-dynamic/bundles/platform-browser-dynamic.umd.js
16.10.07 12:49:57 304 GET /app/app.module.js
16.10.07 12:49:57 304 GET /node_modules/@angular/compiler/bundles/compiler.umd.js
16.10.07 12:49:57 304 GET /node_modules/@angular/core/bundles/core.umd.js
16.10.07 12:49:57 304 GET /node_modules/@angular/platform-browser/bundles/platform-browser.umd.js
```



# Angular 2 Essential

## Структура Hello World проекта



# Angular 2 Essential

## Что такое TypeScript

**TypeScript** – язык программирования, который является надстройкой над JavaScript. TypeScript компилируется в JavaScript и является обратно совместимым.

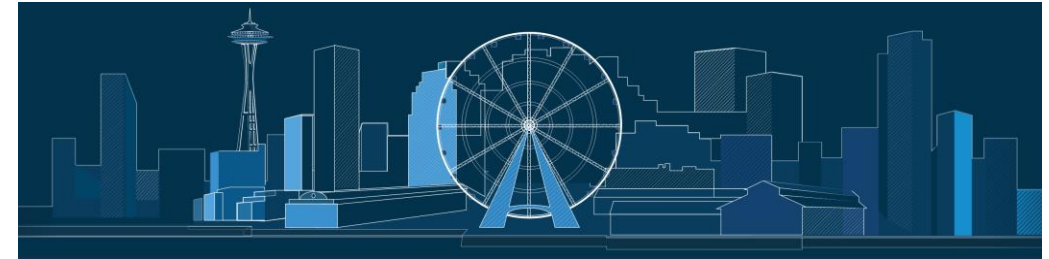
- Явная типизация
- Классы, интерфейсы, перечисления
- Модули

Установка TypeScript

```
npm install -g typescript
```

Компиляция файла test.ts

```
tsc test.ts
```



tsconfig.json x

```
1  {
2    "compilerOptions": {
3      "target": "es5",
4      "module": "commonjs",
5      "moduleResolution": "node",
6      "sourceMap": true,
7      "emitDecoratorMetadata": true,
8      "experimentalDecorators": true,
9      "removeComments": false,
10     "noImplicitAny": false
11   }
12 }
```

tsconfig.json - файл с настройками TSC

# Angular 2 Essential

## Типы данных в TypeScript



- Boolean
- String
- Number
- Array
- Tuple
- Enum
- Any
- Void
- Null
- Undefined
- Never

<https://www.typescriptlang.org/docs/handbook/basic-types.html>

Функция, с указанием типов аргументов и возвращаемого значения

```
function Add(a: number, b: number): number {  
    let sum: number = a + b;  
    return sum;  
}
```

# Angular 2 Essential

## Классы в TypeScript

Определение класса

Свойство

Конструктор

Метод

```
class Greeter {  
  greeting: string;  
  constructor(message: string) {  
    this.greeting = message;  
  }  
  greet() {  
    return "Hello, " + this.greeting;  
  }  
}  
  
let greeter = new Greeter("world");
```

# Angular 2 Essential

## Интерфейсы в TypeScript

Интерфейсы дают возможность определить контракты между объектами.

В TypeScript используется «утиная типизация» (*границы использования объекта определяются его текущим набором методов и свойств*)

```
function printLabel(labelledObj: { label: string }) {  
    console.log(labelledObj.label);  
}  
  
let myObj = {size: 10, label: "Size 10 Object"};  
printLabel(myObj);
```

```
interface LabelledValue {  
    label: string;  
}  
  
function printLabel(labelledObj: LabelledValue) {  
    console.log(labelledObj.label);  
}  
  
let myObj = {size: 10, label: "Size 10 Object"};  
printLabel(myObj);
```

# Angular 2 Essential

## Декораторы

**Декораторы** – особый вид определений, который позволяет добавить к классам, методам, свойствам, методам доступа и аргументам **метаданные**.

@sealed

```
class Greeter {  
  greeting: string;  
  constructor(message: string) {  
    this.greeting = message;  
  }  
  greet() {  
    return "Hello, " + this.greeting;  
  }  
}
```

```
class Greeter {  
  greeting: string;  
  constructor(message: string)  
    this.greeting = message;  
  }  
  @enumerable(false)  
  greet() {  
    return "Hello, " + this.g  
  }  
}
```

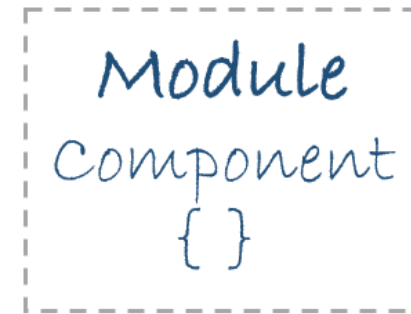
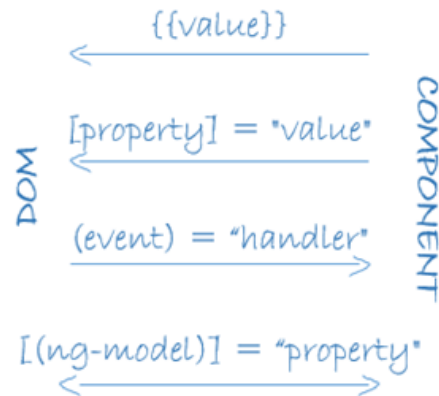
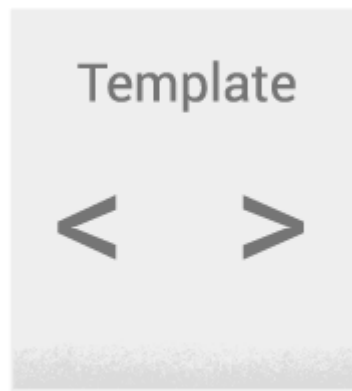
```
class Point {  
  private _x: number;  
  private _y: number;  
  constructor(x: number, y: number) {  
    this._x = x;  
    this._y = y;  
  }  
  @configurable(false)  
  get x() { return this._x; }  
  @configurable(false)  
  get y() { return this._y; }  
}
```



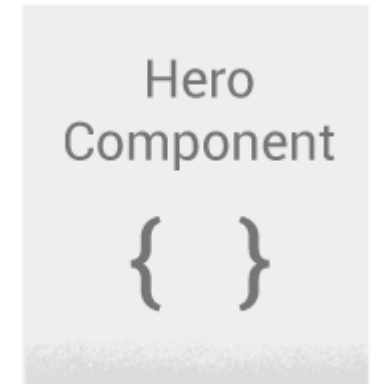
# Angular 2 Essential

## Общая архитектура – главные строительные блоки

- Модули (Modules)
- Компоненты (Components)
- Шаблоны (Templates)
- Метаданные (Metadata)
- Привязка данных (Data Binding)
- Директивы (Directives)
- Сервисы (Services)
- Внедрение зависимостей (Dependency Injection)



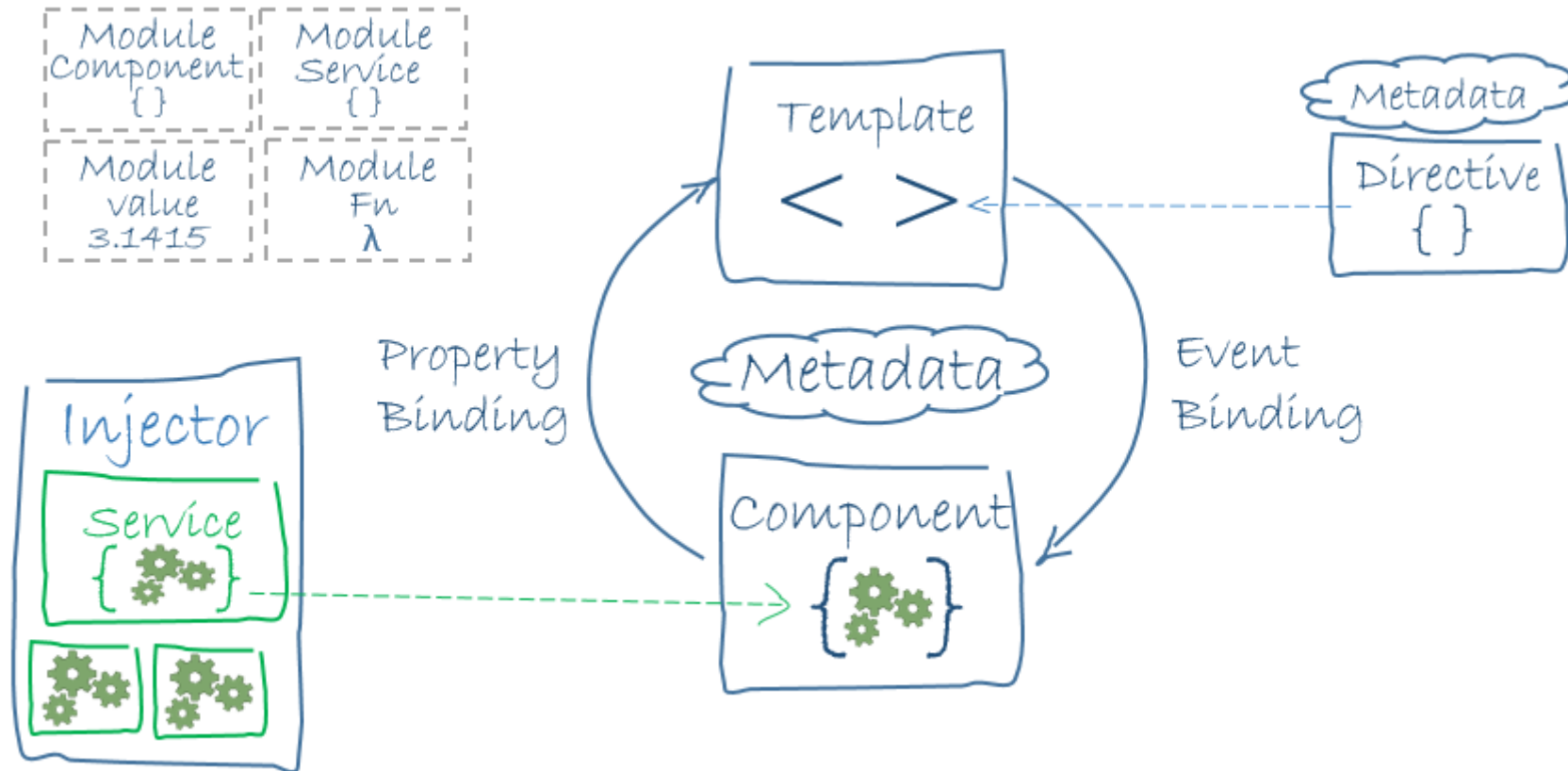
Metadata



Metadata

# Angular 2 Essential

## Общая архитектура



# Angular 2 Essential

## Модули

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

**Модуль** – коллекция связанных элементов Angular приложения.

- **declarations** – компоненты, директивы и pipes, которые принадлежат модулю
- **exports** – компоненты, которые будут видимы для других модулей
- **imports** – модули, которые необходимы этому модулю для работы
- **providers** – сервисы, которые данный модуль добавляет в глобальную коллекцию сервисов
- **bootstrap** – главное представление приложения, также называется AppComponent. Только root module должен устанавливать свойство bootstrap

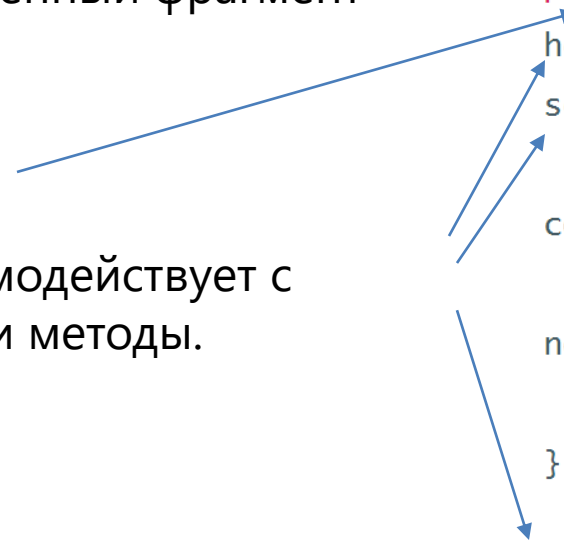
# Angular 2 Essential

## Компонент

**Компонент** контролирует определенный фрагмент интерфейса приложения.

Компонент представлен классом

Пользовательских интерфейсов взаимодействует с классом через открытые свойства и методы.



```
export class HeroListComponent implements OnInit {  
  heroes: Hero[];  
  selectedHero: Hero;  
  
  constructor(private service: HeroService) { }  
  
  ngOnInit() {  
    this.heroes = this.service.getHeroes();  
  }  
  
  selectHero(hero: Hero) { this.selectedHero = hero; }  
}
```

# Angular 2 Essential

## Шаблоны

Пользовательских интерфейсов приложения определяется с помощью **шаблона (template)**.

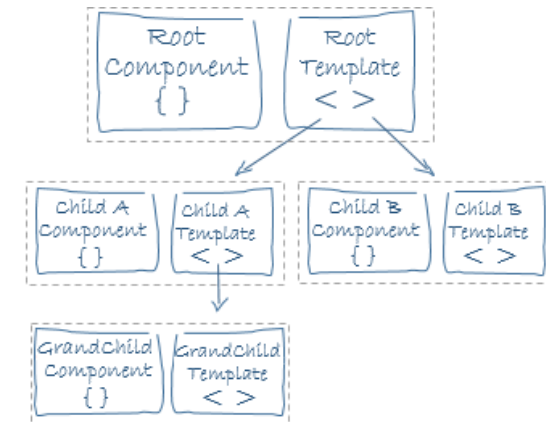
```
<h2>Hero List</h2>
<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>
<hero-detail *ngIf="selectedHero" [hero]="selectedHero"></hero-detail>
```

Директива

Обработчик события

Привязка данных

Дочерний компонент



# Angular 2 Essential

## Метаданные

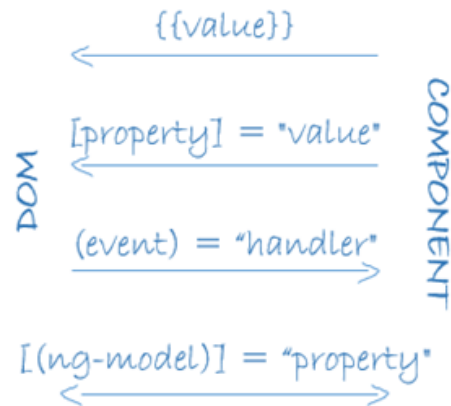
Метаданные определяют, как Angular должен обрабатывать класс. При использовании TypeScript метаданные добавляются к классу с помощью декораторов.

```
@Component({
  moduleId: module.id,
  selector: 'hero-list',
  templateUrl: 'hero-list.component.html',
  providers: [ HeroService ]
})
export class HeroListComponent implements OnInit {
  /* . . . */
}
```

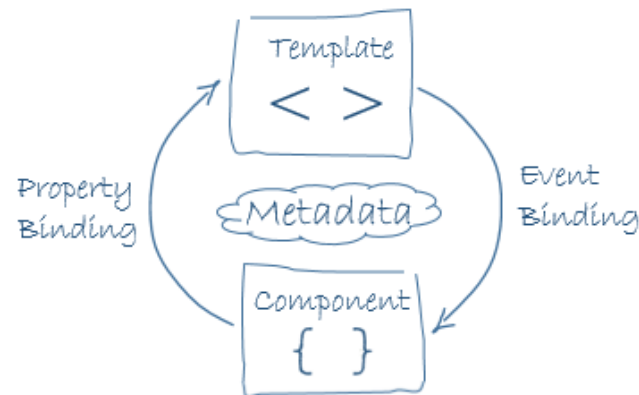
- **moduleId** – устанавливает источник для базового адреса для создания путей относительно модуля, например, для свойства `templateUrl`
- **selector** – CSS селектор, который определяет какой HTML элемент необходимо инициализировать как КОМПОНЕНТ.
- **tempalteUrl** – путь к HTML шаблону
- **providers** – массив провайдеров для внедрения зависимостей.

# Angular 2 Essential

## Привязка данных



Тип привязки	пример
Интерполяция	<code>{{ value }}</code>
Привязка свойств	<code>[property]="value"</code>
Привязка событий	<code>(event)="handler"</code>
Двунаправленная привязка	<code>[(ng-model)]="property"</code>



# Angular 2 Essential

## Директивы

**Директива** – класс со специальными метаданными для изменения структуры DOM дерева или определения специального поведения или стиля для отдельных элементов DOM дерева.



- Structural
- Attribute

```
<li *ngFor="let hero of heroes"></li>
```

```
<hero-detail *ngIf="selectedHero"></hero-detail>
```

```
<input [(ngModel)]="hero.name">
```



# Angular 2 Essential

## Сервисы



Сервис – класс, который предоставляет функции или значения необходимые приложению.

Примеры сервисов:

*Сервис для логирования*

*Сервис с бизнес логикой*

*Сервис для получения данных с сервера*

*Сервис конфигурации*

Создание сервиса (обычный класс)

```
export class Logger {  
  log(msg: any) { console.log(msg); }  
  error(msg: any) { console.error(msg); }  
  warn(msg: any) { console.warn(msg); }  
}
```

Получение сервиса

```
constructor(  
  private backend: BackendService,  
  private logger: Logger) { }
```

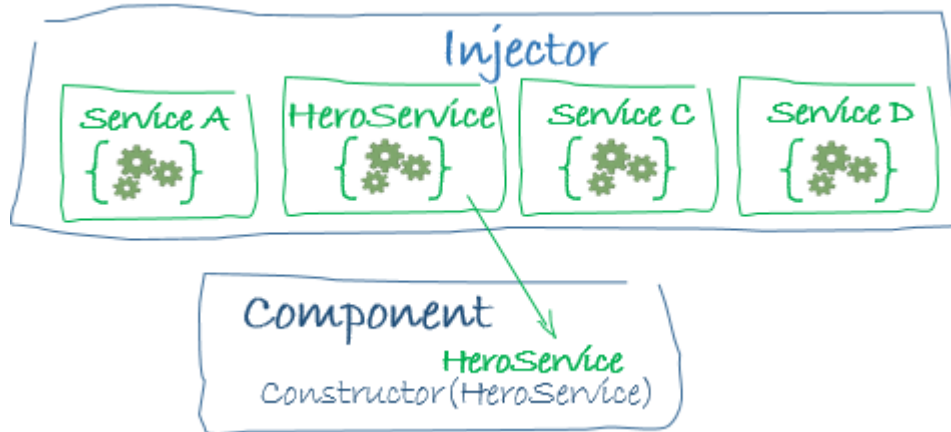
# Angular 2 Essential

## Dependency Injection



**Внедрение зависимостей** – способ создания и установки экземпляра класса, который является зависимостью для другого класса.

Для того, чтобы определить зависимости Angular проверяет типы аргументов конструктора создаваемого компонента (или другого элемента).



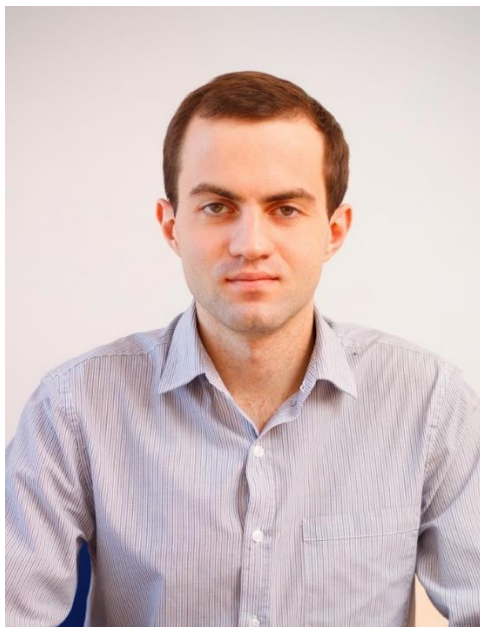
ЗАВИСИМОСТЬ



```
constructor(private service: HeroService) { }
```

# Angular 2 Essential

Спасибо за внимание! До новых встреч!



Охрименко Дмитрий  
МСТ



MCID: 9210561

# Информационный видеосервис для разработчиков программного обеспечения

