API CONFERENCE

MAKE NODEJS APIs GREAT WITH TYPESCRIPT

zalando

ABOUT ME

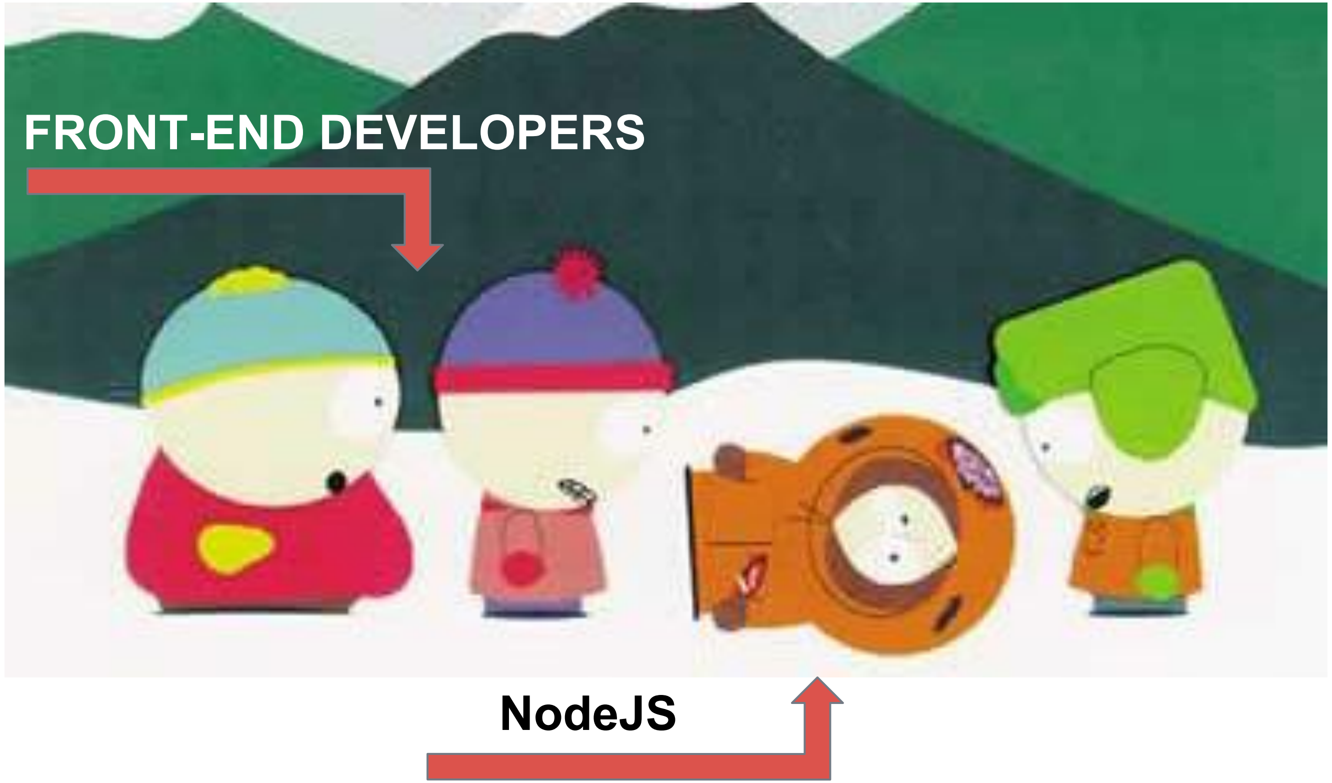
I really like my work, software engineering never makes me bored, always keeps in learning and improving mood.

dmytro.zharkov@gmail.com

http://bit.ly/2FAm3Lr

in http://bit.ly/2FAm3Lr

NODEJS REPUTATION



What's wrong with NodeJS APIs?

- Dynamic VS strong typing
- Pure OOP model of JS
- No of access modifiers
- No hypermedia in RESTful APIs
- Lack of documentation
- Design and structure

zalando

TYPESCRIPT FOR A RESQUE

TYPE SYSTEM

INTERFACES

DECORATORS

ACCESS MODIFIERS

GENERICS

ABSTRACT CLASSES

TYPES TRIPLET

TYPE ALIASES

```
public port: number;
```

INTERFACES

```
protected server: http.Server;
```

CLASSES

```
public article: Article;
```

LET’S COMPARE

TYPE ALISES

- Primitive types (number, string, boolean) and reference types (Object).
- Can’t be extended

INTERFACES

- Reference types only
- Can be extended
- Signature not implementation

CLASSES

- Reference types only
- Can be extended
- Signature and implementation

INTERFACES

```
interface BaseArticle {
 SKU: string,
 name: string,
 type: string,
 price: Price
}

export default BaseArticle;
```

```
interface FashionArticle extends BaseArticle {
 size: Sizes,
 color: Colors
}

export default FashionArticle;
```

```
import { Document } from "mongoose";

interface FashionArticleModel extends FashionArticle, Document {};
export default FashionArticleModel;
```

USING INTERFACES AND GENERICS

```
import { Schema, Model, model} from "mongoose";
import FashionArticleModel from "../interfaces/FashionArticleModel";

const ArticleSchema: Schema = new Schema({
 ....
});



const ArticleModel: Model<FashionArticleModel> =
model<FashionArticleModel>("Article", ArticleSchema);

export {ArticleModel};
```

MORE INTERFACES

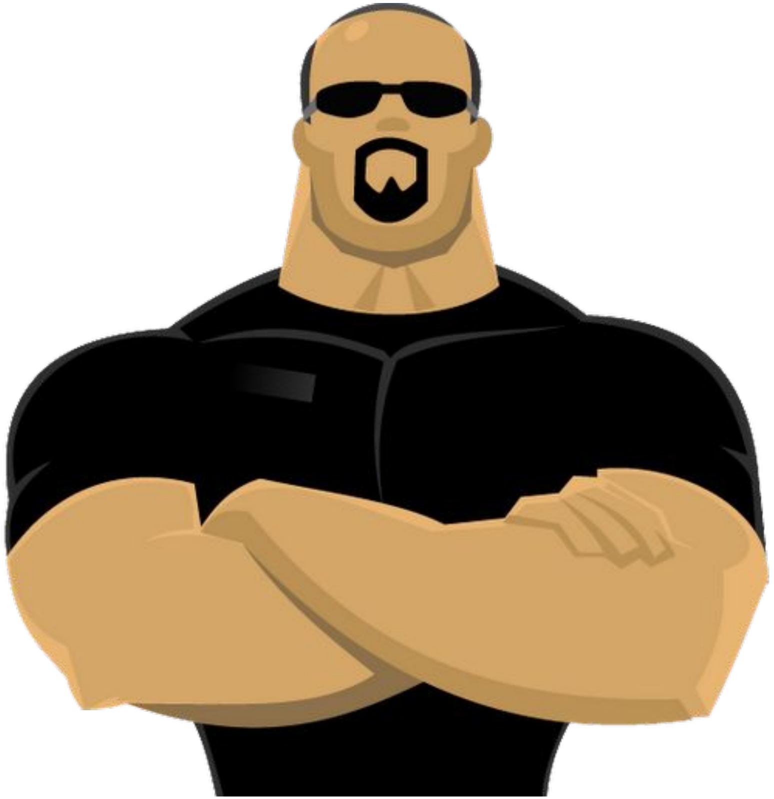
```
export class Server {

 protected app: express.Application;

 protected server: http.Server;

 private db: mongoose.Connection;

 private routes: express.Router[] = [];

}
```

ACCESS MODIFIERS

PUBLIC
Default modifier and can be omitted (better not).

PRIVATE
Restics members visibility to current class only.

PROTECTED
Visible in derived classes.



IN PRACTICE

```
class Server {
 protected app: express.Application;
 protected server: http.Server;
 public port: number;

 constructor(port: number = 3000) {
   this.app = express();
   this.port = port;
   this.app.set("port", port);
   this.app.listen(this.port);
 }
}
```

```
import * as io from "socket.io";

class SocketServer extends Server {
 private socketServer: io.Server;

 constructor(public port: number) {
   super(port);
   this.socketServer = io(this.server);
   ....
 }
}
```

DECORATORS

CLASSES

METHODS

PARAMETERS

FIELDS

EXAMPLE: TRY-CATCH WRAPPER

```
class TestClass {

  @safe
   public doSomething(str: string): boolean {
     return str.length > 0;
   }

}

var safeTest: TestClass = new TestClass();
safeTest.doSomething(null);
safeTest.doSomething("Hello from IJS and API conference");
```

EXAMPLE: TRY-CATCH WRAPPER

```
function safe(target: any, propertyKey: string, descriptor:
TypedPropertyDescriptor<any>): TypedPropertyDescriptor<any> {

   let originalMethod = descriptor.value;
   descriptor.value = function () {
     try {
       originalMethod.apply(this, arguments);
     } catch(ex) {
       console.error(ex);
     }
  };

 return descriptor;
}
```

EXAMPLE: DECLARATIVE ROUTES

Typical ExpressJS route:

```
var express = require('express');
var router = express.Router();

router.get('/', function(req, res, next) {
 res.send('respond with a resource');
});

router.get('/:id', function(req, res, next) {
 res.send('respond with a resource');
});
```

EXAMPLE: DECLARATIVE ROUTES

```
@RouteHandler("/sample-route")
class SampleRoute {
  public router: Router;
  constructor(public app: Server) {}

  @Get()
  public resources(req: Request, res: Response) {
     res.json([]);
  }

  @Get("/:id")
  public resources(req: Request, res: Response) {
     res.json({});
  }
}
```

EXAMPLE: REQUEST VALIDATION

```
@Validate({
 param: "name",
 validate: "required"
})
public createArticle(request: Request, response: Response): void {
	//  create an article
}
```

EXAMPLE: REQUEST VALIDATION

```
export function Validate(params: Array<any>): any {
 return (target: Object, propertyKey: string): TypedPropertyDescriptor<any> => {

const descriptor = Object.getOwnPropertyDescriptor(target, propertyKey);
    const originalMethod = descriptor.value;

    descriptor.value = function () {
      const body = arguments[0].body;
      const response = arguments[1];
        // do some work
    };
    return descriptor;
 };
}
```

EXAMPLE: HATEOAS

Hypermedia allows to decouple client and server to a large extent and allow them to evolve independently.

EXAMPLE: HYPERMEDIA

DEMO

DOCUMENTATION

SWAGGER

TSOA

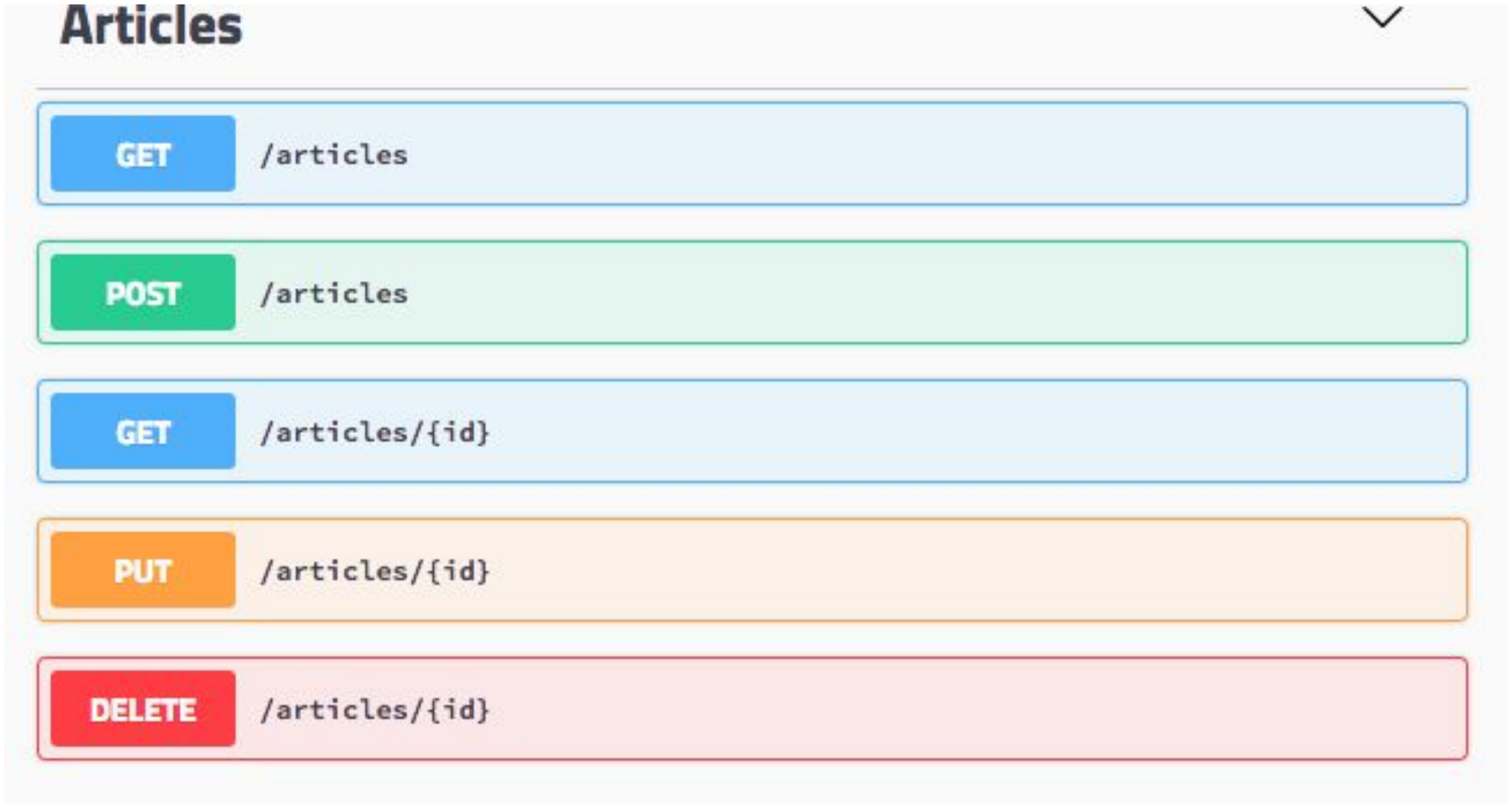
TYPE DOC

AUTO SWAGGER DOCUMENTATION WITH DECORATORS

TSOA

```
@Route("Users")
class ArticlesService {

  @Post()
  public createArticle(@Body() requestBody: FashionArticle): Promise<FashionArticle> {
    //  create an article
  }
}

export default ArticlesService;
```

AUTO SWAGGER DOCS WITH DECORATORS



Articles

GET /articles

POST /articles

GET /articles/{id}

PUT /articles/{id}

DELETE /articles/{id}

UNIT AND INTEGRATION TESTS

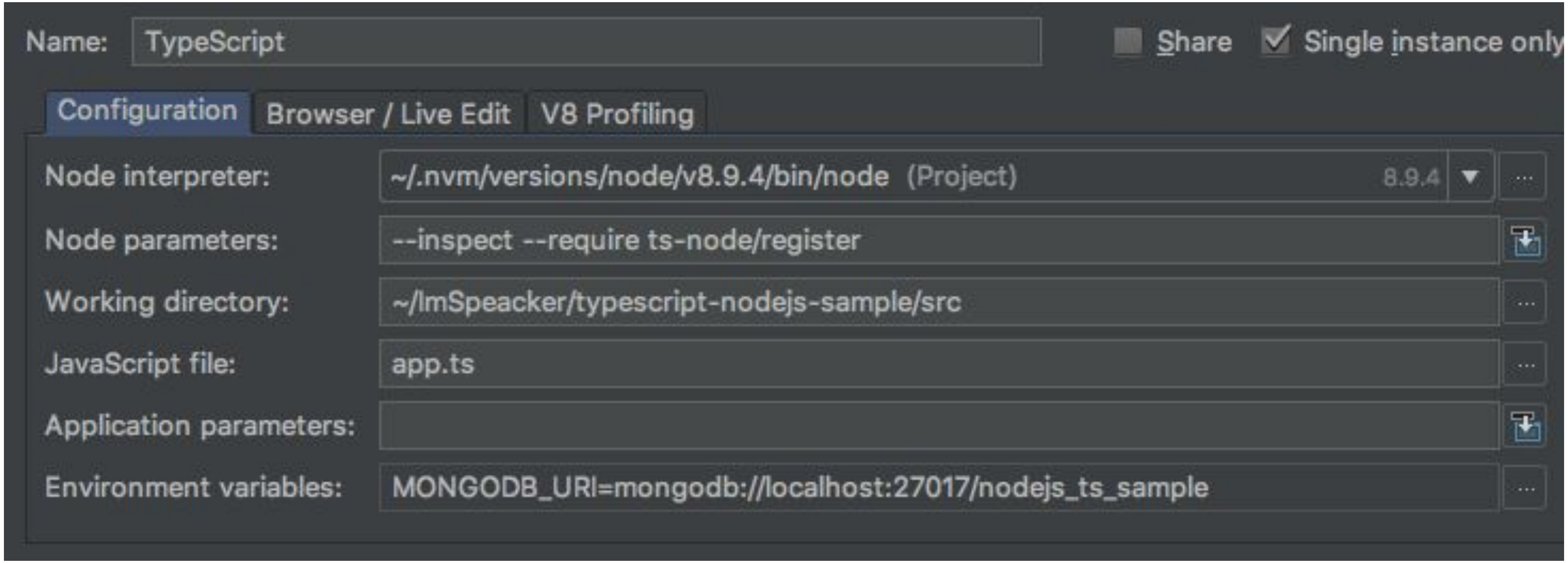
WRITE YOUR TESTS IN TYPESCRIPT

Jest

SUPERTEST

SINON.JS

DEBUG DIRECTLY IN TS



Name: TypeScript

☐ Share ✓ Single instance only

Configuration | Browser / Live Edit | V8 Profiling

Node interpreter:	~/.nvm/versions/node/v8.9.4/bin/node (Project) 8.9.4
Node parameters:	--inspect --require ts-node/register
Working directory:	~/ImSpeacker/typescript-nodejs-sample/src
JavaScript file:	app.ts
Application parameters:	
Environment variables:	MONGODB_URI=mongodb://localhost:27017/nodejs_ts_sample

zalando

API CONFERENCE

DRAWBACKS

Additional build step.

Lack of definition files for npm modules.

Source and compiled code.

MAKE NODEJS APIs GREAT WITH TYPESCRIPT

THANKS FOR ATTENTION

Slides	
NodeJS TypeScript starter	http://bit.ly/2FW64qn
TSOA auto swagger documentation	http://bit.ly/2FW64qn