JavaScript and TypeScript

© ORIFLAME COSMETICS AG,2018

Today Topics

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

1. Single Page Applications
2. JS Ecosystem
   I. Node
   II. Npm
      I. Init project
      II. Dependencies
      III. Global dependencies
      IV. Lock file
      V. Carrets and tildes
      VI. Upgrading packages
3. Javascript ES6+ & Typescript
   I. Differences between JS&TS
   II. Variables using let&const
   III. String templates, null asserts
   IV. Arrow functions
   V. Modules, Exports and default exports
   VI. Classes, abstract classes, interfaces
   VII. Spread and rest operator, object destructuring

© ORIFLAME COSMETICS AG,2018

4. Basic functions

I. Array iterations

II. Fetching data

III. Parsing JSON

IV. Working with dates

© ORIFLAME COSMETICS AG,2018

Chapter 1

Single Page Applications

© ORIFLAME COSMETICS AG,2018

What is SPA?

Most logic living on FE

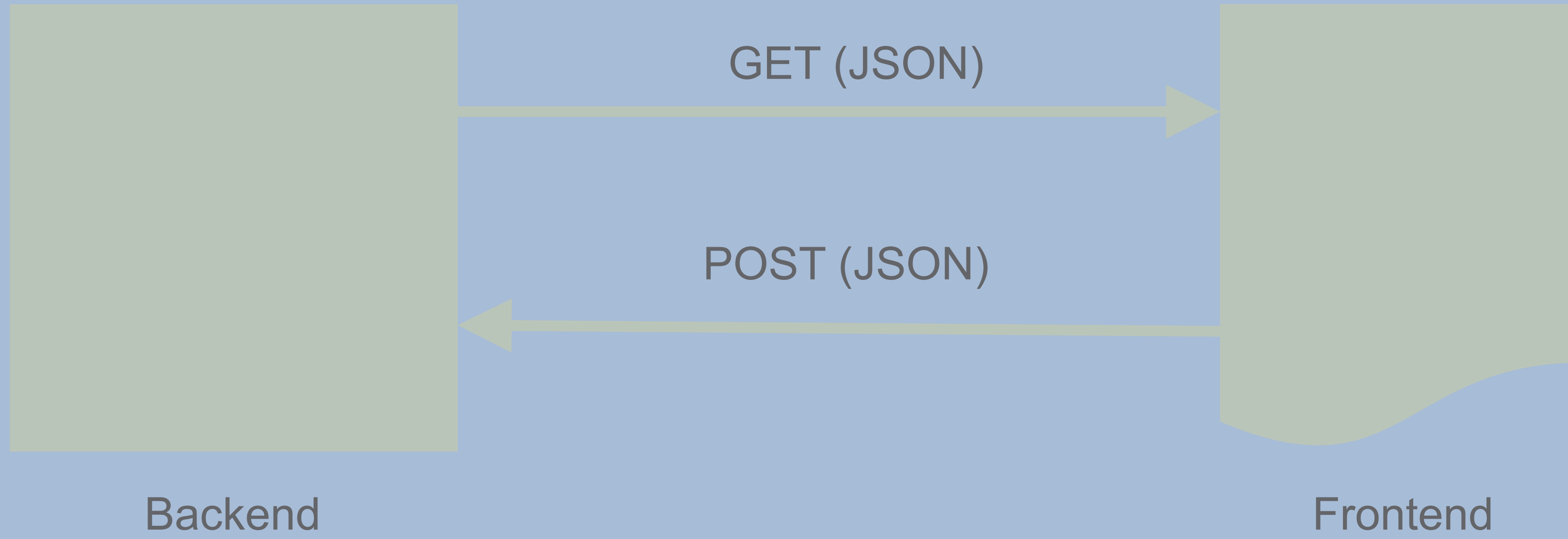
Written or transpiled to JS

Reactive application with no blick

High level of control over app

Decoupled from backend

© ORIFLAME COSMETICS AG,2018



© ORIFLAME COSMETICS AG,2018

Chapter 2

Javascript Ecosystem

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Node.js

As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications.

© ORIFLAME COSMETICS AG,2018

npm

is the package manager for JavaScript and the world's largest software registry.

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Initiate npm project

```
npm init
```

- Creates package.json
- Package.json contains:
  - Project description
  - Dependencies
  - Runnable commands

© ORIFLAME COSMETICS AG,2018

Initiate npm project

```
npm init
```

- Creates package.json
- Package.json contains:
  - Project description
  - Dependencies
  - Runnable commands

© ORIFLAME COSMETICS AG,2018

Depencencies

```
npm install --save-dev typescript

npm i -D typescript
```

- *Dev packages only available during develepoment*

```
npm install --save react react-dom

npm i -S typescript react react-dom
```

- *Regular dependencies also in dist package*

- Adds entry to package.json
- Creates/updates lock file

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Global depencencies

```
npm install --global --save typescript
```

- Installed in node folder
- Available as cmd app
- In most cases not recommended

© ORIFLAME COSMETICS AG,2018

Lock file

- When packege was installed one day in version 1 on other machine next day could be installed in version 2
- Lock prevents it
- Also guarants installation order. Npm packages are not necessarily installed in the same order and this caused problems sometimes.

```
npm install package-a@3.0.0 --package-lock-only
```

© ORIFLAME COSMETICS AG,2018

Carret and tilde

```
"*"
"^1.4.3" -> match any major 4.x.x
"3.4.1"
"~4.2.1" -> match any minor 4.2.x
```

- You can you `install` or `update` for updating packages
- Please not update is not updating dev dependencies unless you add –dev flag

```
npm install express@latest --save --force
```

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Carret and tilde

```
{ "name": "my-project", "version": "1.0", // install update
"dependencies": { // ------------------
"already-installed-versionless-module": "*", // ignores "1.0" -> "1.1"
"already-installed-semver-module": "^1.4.3" // ignores "1.4.3" -> "1.5.2"
"already-installed-versioned-module": "3.4.1" // ignores ignores
"not-yet-installed-versionless-module": "*", // installs installs
"not-yet-installed-semver-module": "~4.2.1" // installs installs
"not-yet-installed-versioned-module": "2.7.8" // installs installs
} }
```

© ORIFLAME COSMETICS AG,2018

Chapter 3

Javascript ES6+ & Typescript

© ORIFLAME COSMETICS AG,2018

What‘s the difference? (JS vs TS)

Typescript is superset of Javascript.
It compiles Javascript but also give you option to use strong typing, additional syntax sugar and transpiles to supported JS.

© ORIFLAME COSMETICS AG,2018

Using let and const

```
console.log(a); // what value is a here?
a = 5;
var a;
```

- vars are hoisted and therefore should be avoided
- is replaced by
  - mutable **let**
  - immutable **const**

© ORIFLAME COSMETICS AG,2018

Using let and const

- let and const both keep their scope since declaration

```
const a = 5;
let b = 6;
```

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

String templates (string interpolation)

- you don‘t have to concatenate strings if you need to add variable content into the text
- you can add any JS expression
- you can use also quotes there

```
let text = `string text ${expression} string text`;
```

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Null assertion

- there is nice syntax to check for null and undefined and get value

```
function(input: any) { // we expect some object with value prop
   let num: number? = input && input.value;
}
```

© ORIFLAME COSMETICS AG,2018

Arrow functions („lambda“)

- shorter function syntax
- no problems with `this` and binding it

```
function(input: number) {
   return input * 5;
}

// can be rewritten to
(input: number): number => ({ input * 5});
```


© ORIFLAME COSMETICS AG,2018
ORIFLAME SWEDEN

Modules

- basic structuring concept of JavaScript code
- you can use function from other files or packages – `import` / `export`

```
import { ZipCodeValidator as ZCV } from "./ZipCodeValidator";


// or later
const validator = await import { ZipCodeValidator } from "./ZipCodeValidator";
```

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Exports

- Export allows to use a function in another file
- default export gives you fallback or exports a single function

```
export function add(a: number, b: number): number {
   return a + b;
}
export default function substract(a: number, b: number): number {
   return a - b;
}

// or export default {add};
```

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Classes

- Shall I introduce you to basic OOP concepts?

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

© ORIFLAME COSMETICS AG,2018

Abstract classes

- they can‘t be instantiated.

```
abstract class Greeter {
  // same code as previous example
}
let greeter = new Greeter("world"); // can‘t do this!!!

class MyGreeter extends Greeter {
  // some additional code
}
let greeter = new MyGreeter("world"); // possible now
```

Interfaces

- You define variables and DECLARE functions.
- Used either for polymorphism or as a „structures“

```
interface IGreeter {
        greeting: string;
        greet();
}
I
class Greeter implements IGreeter { }
```

© ORIFLAME COSMETICS AG,2018

Abstract or interface?

Mostly use interfaces

Avoid abstract classes

One has functional code other not

© ORIFLAME COSMETICS AG,2018

Class inheritance?

Try to avoid it

Also avoid base classes in most cases

Try to use aggregates

© ORIFLAME COSMETICS AG,2018

Spread operator

- Used on arrays, objects or functions
- Copy object, change props immutably, merge arrays…

```
var obj1 = { foo: 'bar', x: 42 };
var obj2 = { foo: 'baz', y: 13 };

var clonedObj = { ...obj1 };
// Object { foo: "bar", x: 42 }


var mergedObj = { ...obj1, ...obj2 };
// Object { foo: "baz", x: 42, y: 13 }
```

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Destructuring

- Same operator as spread
- It gets variables from same structure/object

```
var o = {p: 42, q: true};
var {p: foo, q: bar} = o;
console.log(foo); // 42
console.log(bar); // true
```

```
var {a = 10, b = 5} = {a: 3};
console.log(a); // 3
console.log(b); // 5
```

© ORIFLAME COSMETICS AG,2018

Chapter 4

Basic functions

© ORIFLAME COSMETICS AG,2018

ORIFLAME
SWEDEN

Iterating arrays

- Use `map()` / `every()` / `filter()` / `find()` / `findIndex()` / `reduce()` / `some()` / ... to iterate over arrays, and `Object.keys()` / `Object.values()` / `Object.entries()` to produce arrays so you can iterate over objects.

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN

Before we fetch some data

What are promisses?

How do we handle them?

© ORIFLAME COSMETICS AG,2018

Fetching data

```
fetch('http://example.com/movies.json‘)
        .then(function(response) { return response.json(); })
                .then(function(myJson) { console.log(JSON.stringify(myJson));
});
```

Fetching data

```
fetch(url, {
        method: "POST", // *GET, POST, PUT, DELETE, etc.
        mode: "cors", // no-cors, cors, *same-origin cache:
        "no-cache", // *default, no-cache, reload, force-cache, only-if-cached
        credentials: "same-origin", // include, same-origin, *omit
        headers: { "Content-Type": "application/json; charset=utf-8",
        // "Content-Type": "application/x-www-form-urlencoded", },
        redirect: "follow", // manual, *follow, error
        referrer: "no-referrer", // no-referrer, *client
        body: JSON.stringify(data), // body data type must match "Content-Type" header
}) .then(response => response.json()); // parses response to JSON
```

© ORIFLAME COSMETICS AG,2018

ORIFLAME SWEDEN