Node.js

Patrick Kraaij @ Zoom - deTesters / TestCoders

House rules

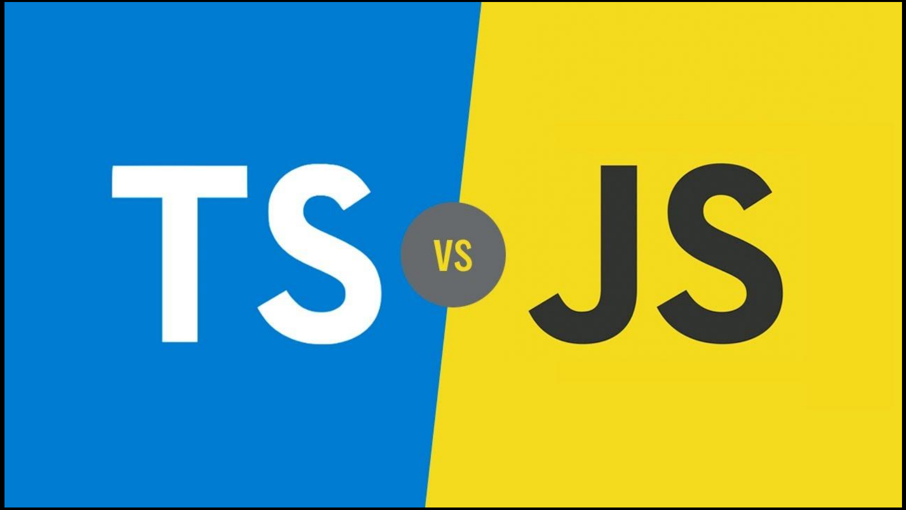
- Put yourself on mute
- Put your question in the chat
- Get a drink / take a bio-break whenever you like

Agenda

- 09:15 - Introduction
- 09:30 - TypeScript
- 11:30 - Lunch
- 12:30 - Node.js
- 15:30 - Closing ceremony

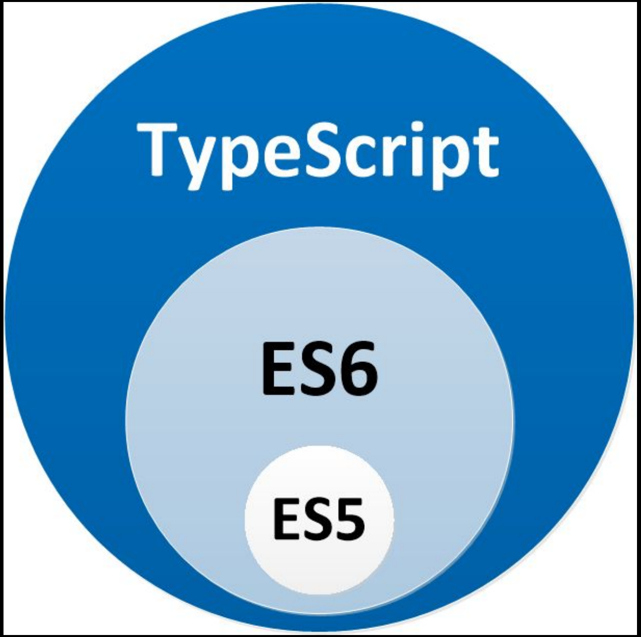
TypeScript // Intro

- Created by Microsoft
- “Strong typed JavaScript”
- Predictable
- Preferred language in Angular
- tsc



TypeScript // What is it

- Superset of ES



TypeScript // Variables

In terms of variables and constants, keyword var is hoisted and let and const does not allow hoisting.

TypeScript // Variables (2)

```
function greet() {
    b = 'hello';
    console.log(b); // hello
    var b;
}

greet(); // hello
console.log(b); // Uncaught ReferenceError: b is not defined
```

TypeScript // Variables (3)

If a variable is used with the let keyword, that variable is not hoisted.

```
a = 5;

console.log(a);

let a; // Uncaught ReferenceError: Cannot access 'a' before

initialization
```

While using let, the variable must be declared first.

TypeScript // Variables (4)

```
let [first, ...rest] = [1, 2, 3, 4];
console.log(first); // outputs 1
console.log(rest); // outputs [ 2, 3, 4 ]

let personObject = { name: "Patrick ", age: 34 };
let { name, age } = personObject;
```

TypeScript // Basic Types

- boolean `// const isEnabled = true;`
- number `// const age = 30;`
- string `// const name = ‘John’;`
- T[] | Array<T> `// const list: Array<number> = [1, 2, 3];`
- [string, number] `// const mixed = [‘John’, 30]`
- enum `// enum Color { Red, Green, Blue }; let c: Color = Color.Green;`
- void
- null
- undefined
- unknown
- any
- never

TypeScript // Interfaces

```
interface IName {
   name: string;
}
function getName(): IName {
   return { name: 'John' }
}

let personName;
function setName(input: string): void {
   personName = input;
}
```

TypeScript // Interfaces: Optional keys

```
interface ICountry {
    countryCode: string;
    telephonePrefix?: string;
}

const NL: ICountry = { countryCode: "NL", telephonePrefix: '+31' }
const BE: ICountry = { countryCode: "BE" }
```

Assignment Time! (~20m)

Assignment // Interfaces and types

- Go to nodejs/exercise-types/interfaces-and-types.ts
- Try to write those functions, add interfaces!
- Change your directory to: cd nodejs
- Check your output with npm run exe1

Recap

TypeScript // Utility Types

- Partial<Type>
- Promise<Type>
- Readonly<Type>

TypeScript // Utility Types (Partial)

```
interface Todo {
   title: string;
   description: string;
}

function updateTodo(todo: Todo, fieldsToUpdate: Partial<Todo>) {
   return { ...todo, ...fieldsToUpdate };
}

const todo1 = {
   title: "organize desk",
   description: "clear clutter",
};

const todo2 = updateTodo(todo1, {
   description: "throw out trash",
});
```

CODE ZILLA

TypeScript // Utility Types (Promise)

```
async function countries(): Promise<string> {
    ...
}
```

TypeScript // Utility Types (Readonly)

```typescript
interface Todo {
  title: string;
}
const todo: Readonly<Todo> = {
  title: "Delete inactive users",
};
todo.title = "Hello";
```

TypeScript // Interfaces: Readonly keys

```
interface IName {
   readonly name: string;
   readonly age: number;
 }
 let person: IName = { name: 'Patrick', age: 34 };
 person.name = 'Rory' // cannot assign
```

Assignment Time! (~20m)

Assignment // Interfaces and types

- Go to nodejs/exercise-types/advanced-types.ts
- Try to write those functions
- Change your directory to: cd nodejs
- Check your output with npm run exe2

Recap

Node.js

Express // Intro

Fast, unopinionated, minimalist web framework for Node.js

Express // Uses in real world

- Web applications
- BFF
- API’s
- Redirect service

Express // Basic example

```
import * as express from 'express';
const app = express();
const port = 4000;

app.get('/', function (req, res) {
   res.send('Hello World!');
});

app.listen(port, function() {
 console.log(`Example app listening at http://localhost:${port}`);
});
```

Express // Routing

```
app.METHOD(PATH, HANDLER)
```

- app: Instance of express
- Method: http request method
- Path: points to a path on the server
- Handler(s): a function which is executed when the path matches

Express // Routing: Request and Response

```
app.get('/hello', function (req, res) {
    res.send('Hello World!');
});


OR


router.get('/hello, function (req, res) {
    res.send('Hello World!');
});
```

Express // Routing: Request and Response (2)

```
router.get('/hello/:name', function (req, res) {
    res.send(`Hello ${req.params.name}`);
});
```

Express // Middleware

```
import * as bp from 'body-parser';

const { Router } = express;

app.use(bp.json());

app.use('/', Router);
```

Express // Error handling

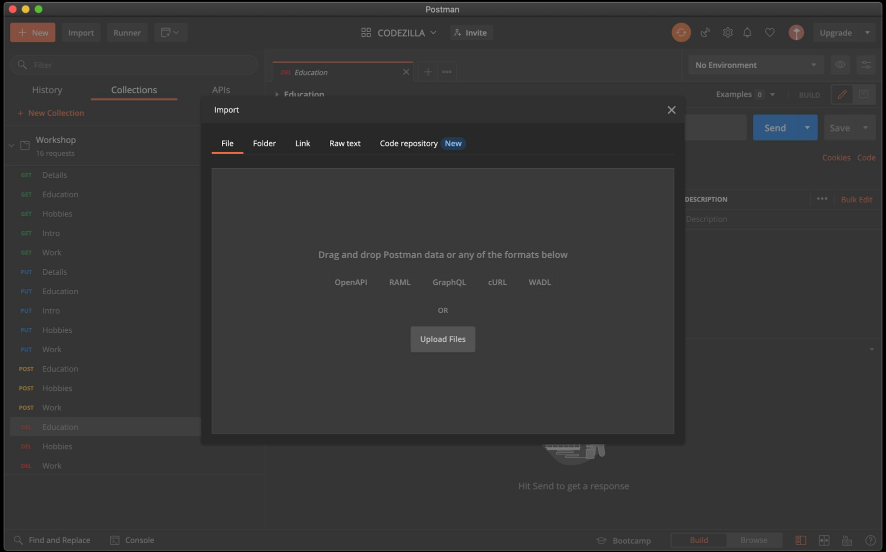
```
app.use(bodyParser.json())
app.use(logErrors)
app.use(clientErrorHandler)
app.use((error, req, res, next) => {
   res.sendStatus(503);
});
```

Express // Error handling (2)

```
app.get('/', function (req, res, next) {
   fs.readFile('/file-does-not-exist', function (err, data) {
      if (err) {
         next(err); // Pass errors to Express.
      } else {
         res.send(data);
      }
   });
});
```

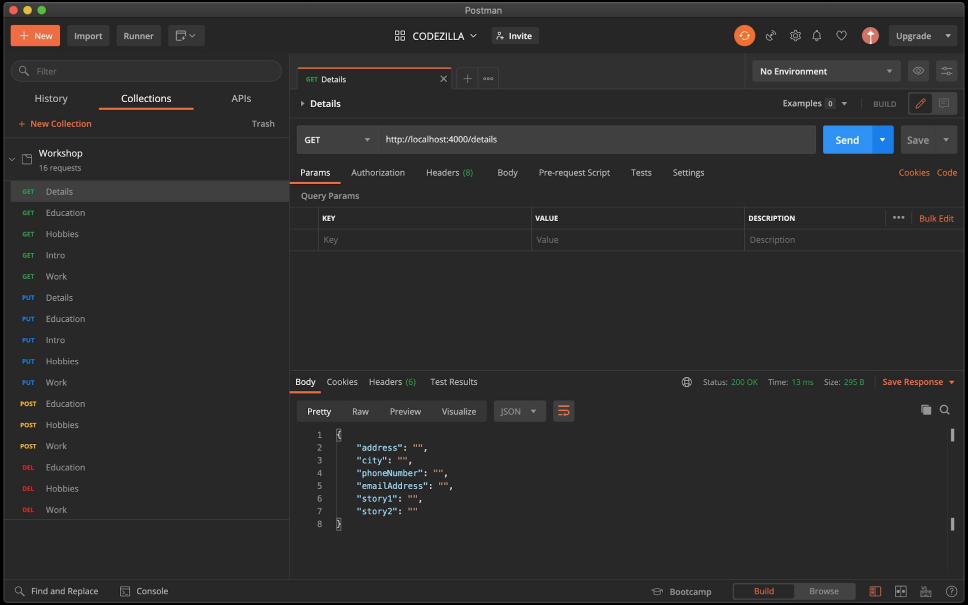
Postman // Import collection

Import nodejs/Workshop.postman_collection.json



Assignment Time! (~30m)

Assignment // Write GET functions



GET http://localhost:4000/details

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 13 ms Size: 295 B

```
{
    "address": "",
    "city": "",
    "phoneNumber": "",
    "emailAddress": "",
    "story1": "",
    "story2": ""
}
```

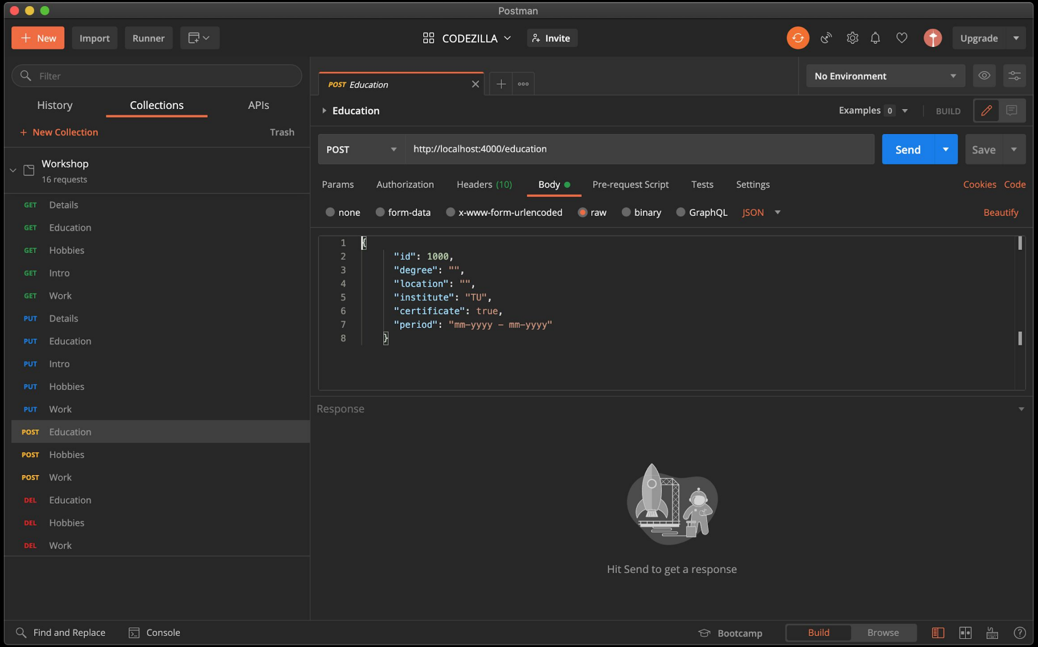
Assignment // GET Example

```
import { Request, Response, NextFunction } from 'express';
import * as fs from 'fs';
import * as path from 'path';

const fileLocation = path.join(__dirname, '../../../../reactjs/src/exercise/data/intro.json');
const file = JSON.parse(fs.readFileSync(fileLocation, 'utf-8'));

export interface Iintro {
   aboutMe: string;
   age: number;
   description: string;
   welcomeMessage: string;
   goal: string;
}

export async function intro(req: Request, res: Response, next: NextFunction): Promise<Response<Iintro>> {
  return res.status(200).json(file);
};
```

Recap

Assignment Time! (~30m)

Assignment // Write POST functions



POST http://localhost:4000/education

```
{
    "id": 1000,
    "degree": "",
    "location": "",
    "institute": "TU",
    "certificate": true,
    "period": "mm-yyyy – mm-yyyy"
}
```

CODE ZILLA

Assignment // POST Example

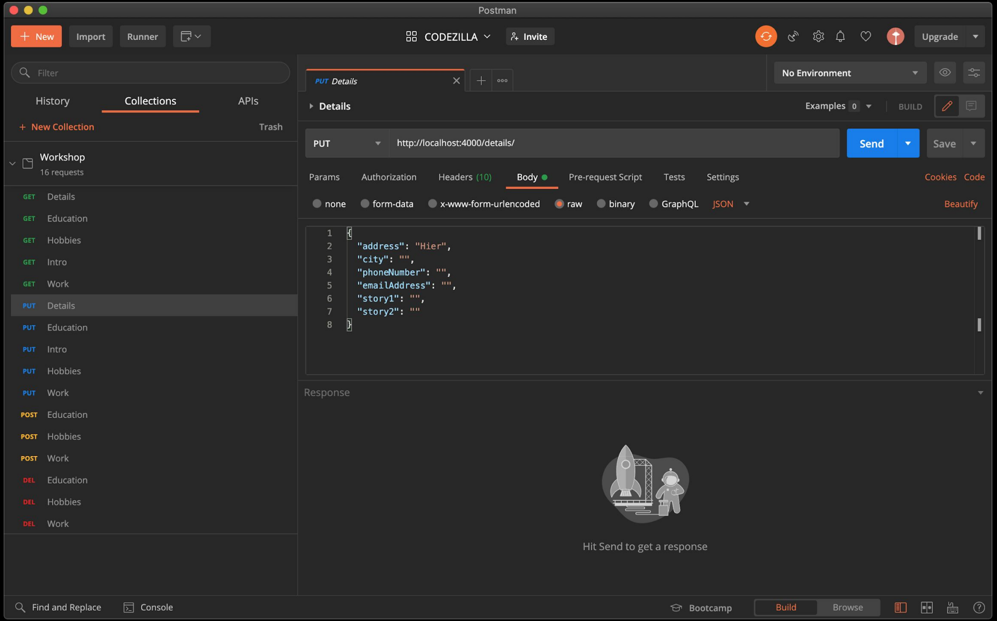
```
export async function addEducation(req: Request, res: Response, next: NextFunction) {
    // add new content to JSON
    file.items.push(req.body);

    // save JSON
    fs.writeFileSync(fileLocation, JSON.stringify(file.items));

    // send success message
    res.status(200).send();
}
```

CODE ZILLA

Recap

Assignment Time! (~20m)

Assignment // Write PUT functions



PUT http://localhost:4000/details/

Body: [✓ raw] JSON

```
{
  "address": "Hier",
  "city": "",
  "phoneNumber": "",
  "emailAddress": "",
  "story1": "",
  "story2": ""
}
```

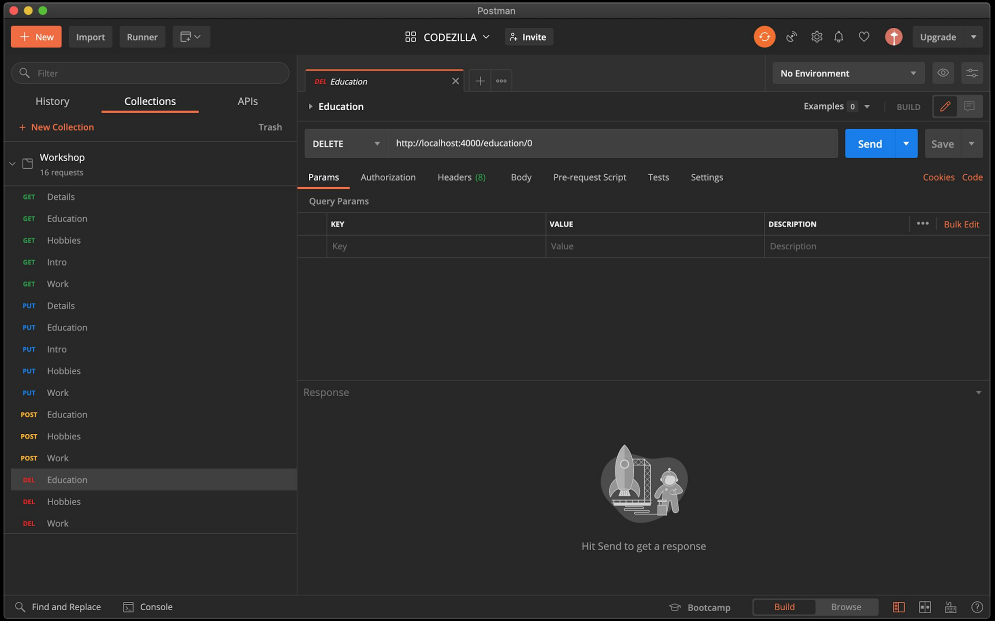
Assignment // PUT Example

```
export async function updateIntro(req: Request, res: Response, next: NextFunction): Promise<void> {
   const body: Iintro = req.body;

   file.aboutMe = body.aboutMe;
   file.age = body.age;
   file.description = body.description;
   file.welcomeMessage = body.welcomeMessage;
   file.goal = body.goal;

   fs.writeFileSync(fileLocation, JSON.stringify(file));
   res.status(200).send();
}
```

Recap

Assignment Time! (~20m)

Assignment // Write DELETE functions



DELETE http://localhost:4000/education/0

Assignment // DELETE Example

```
export async function deleteEducation (req: Request, res: Response, next: NextFunction) {
    // get the id from the url
    const id = req.params.id;

    // retrieve all items expect the one you want to delete
    const allEducationsExpectTheRemovedOne  = file.items.filter(item => {
        return item.id != id; // strict check does not remove the given id
    });

    // save JSON
    fs.writeFileSync(fileLocation, JSON.stringify({ items: allEducationsExpectTheRemovedOne }));

    // send success message
    res.status(200).send();
}
```

Recap

Thanks for your attention!

Are there any questions?