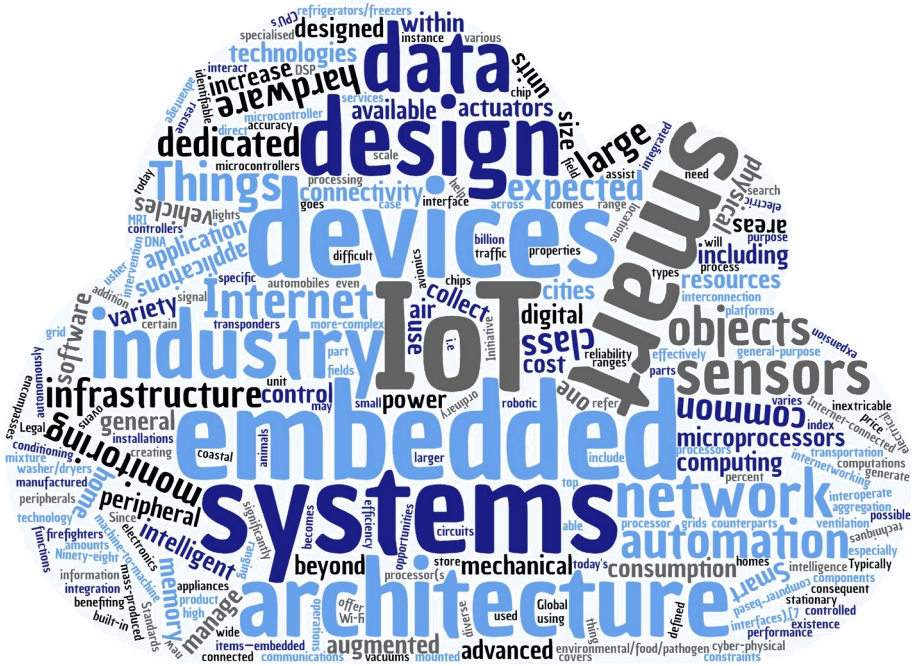
Associate Professorship of Embedded Systems and Internet of Things
Department of Electrical and Computer Engineering
Technical University of Munich

Lecture
IoT Remote Lab

04 – Node.js

Ege Korkan



Zoom Guidelines

- The lectures and tutor sessions happen on Zoom meetings following the link sent to you via email.
- The lecture will be recorded and uploaded in a way that is accessible only to course participants
- The recording will be paused when a student speaks.
- Tutor sessions will not be recorded.
- Participatition to Zoom sessions is optional
- You can choose a random string for your name
- The Zoom chat will not be recorded and we will not save the chat.
- All the participants except the lecturer is muted. You must go to participants, click the hand icon to raise your hand and then unmute yourself.
- You can also do other things, like asking me to go slower. I have a separate window where I look at the requests from the participants.
- You can ask quick questions in Zoom chat or use the Tweedback link provided in each session.

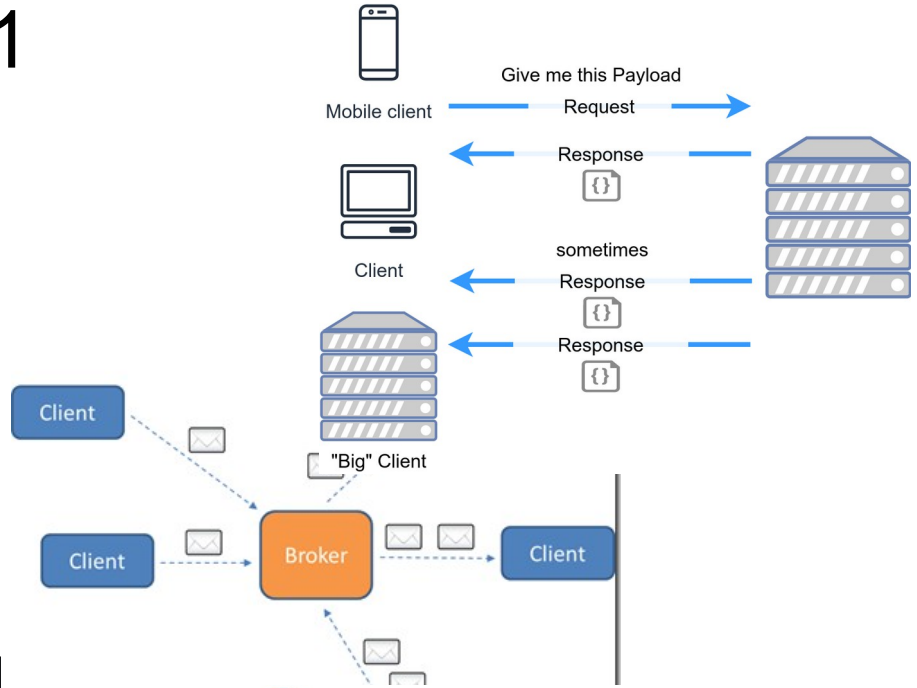
Tweedback for Real-time Q&A During the Lecture

https://tum.tweedback.de/pk81

Recap: Season 1

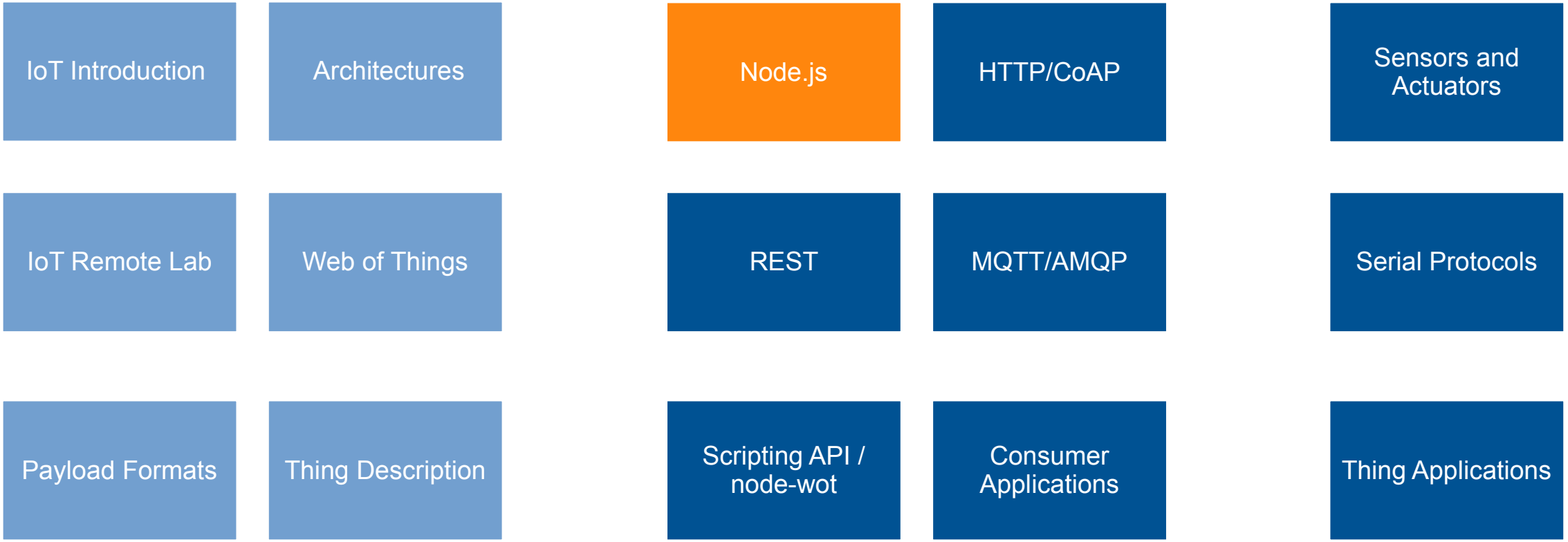
```
{
  "productId": 1,
  "productName": "An ice sculpture",
  "price": 12.50,
  "tags": [ "cold", "ice" ],
  "dimensions": {
    "length": 7.0,
    "width": 12.0,
    "height": 9.5
  },
  "warehouseLocation": {
    "latitude": -78.75,
    "longitude": 20.4
  }
}
```

```
{
  "$schema": "http://json-schema.org/draf
  "$id": "http://example.com/product.sche
  "title": "Product",
  "description": "A product from Acme's c
  "type": "object",
  "properties": {
    "productId": {
      "description": "The unique identifi
      "type": "integer"
    },
    "productName": {
      "description": "Name of the product
      "type": "string"
    },
    "price": {
      "description": "The price of the pr
      "type": "number",
      "exclusiveMinimum": 0
    },
```




```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLampThing",
    "securityDefinitions": {
        "basic_sc": {"scheme": "basic", "in":"header"}
    },
    "security": ["basic_sc"],
    "properties": {
        "status" : {
            "type": "string",
            "forms": [{
                "href": "https://mylamp.example.com/status",
                "htv:methodName":"GET"
            }]
        }
    },
    "actions": {
        "toggle" : {
            "forms": [{
                "href": "https://mylamp.example.com/toggle",
                "htv:methodName":"POST"
            }]
        }
    },
    "events":{
        "overheating":{
            "data": {"type": "string"},
            "forms": [{
                "href": "https://mylamp.example.com/oh",
                "htv:methodName":"GET",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

Course Contents



Deliverable 1

- You manage your own git repository and submit via git bundle to Moodle
- Deadline of Deliverable 1 is **27.05.2020 23:59 (11:59 pm)**.
- Late submissions will not be accepted at all, you simply lose 40% of your grade. Better to submit an empty repository.
- You can use GitHub (https://github.com/), GitLab from TUM (https://gitlab.lrz.de/), or any other Cloud-based git provider. You can also self manage a repository.

Deliverable 1: Part 1

- Part 1 of Deliverable 1 is available on Moodle.
- *I have updated the pdf **again** to make it clear*
  - Thank you for all the feedbacks, give me more!

Deliverable 1: Part 2

- Part 2 of Deliverable 1 will be available on Moodle tonight
- It will focus on writing Thing Descriptions
  - Validation:
    - TD Playground : http://plugfest.thingweb.io/playground/
    - TD Validation Schema : https://github.com/w3c/wot-thing-description/blob/master/validation/td-json-schema-validation.json
  - You will write small parts of TDs as well, which are not valid on their own.

Teaching a Programming Language

- Is a difficult job!
  - You will actually learn by doing
  - So many aspects to teach

- Let‘s not teach the language syntax but the parts that make it a ``framework``.
- Make sure to install nodejs, npm, typescript for your OS
- Official references for Node.js different versions: https://nodejs.org/en/docs/

Node.js

- It is actually a framework, Javascript (or ECMAScript) is the language.
  - So when you have a question to google on the framework related questions, use Node.js and for simple language specific questions, use Javascript (like how to get the 4th element of an array)
- A bit of history:
  - JavaScript was created for Web browsers, to do *small* programs that would help with animations, submitting forms, etc. Around 1995 were the baby steps
  - The processing capabilities of the browsers increased, web pages got bigger, browsers became more of a business → Importance and use of client-side processing (via scripting) increased
  - The JavaScript language got mature enough → JS engines got mature enough
  - Node.js was born in 2009. It allows running JS outside of browsers

Basic Examples

```
console.log("Hello World")
// → Hello World
```

```
if (!Number.isNaN(theNumber)) {
  console.log("Your number is the square root of " +
              theNumber * theNumber);
}
```

```
for (let i = 0; i < JOURNAL.length; i++) {
  let entry = JOURNAL[i];
  // Do something with entry
}
```

Basic Examples

```
const power = function(base, exponent) {
  let result = 1;
  for (let count = 0; count < exponent; count++) {
    result *= base;
  }
  return result;
};

const power = (base, exponent) => {
  let result = 1;
  for (let count = 0; count < exponent; count++) {
    result *= base;
  }
  return result;
};
```

Node.js

- Some features you may not be used to:
  - Async
    - Callbacks
    - Promises
    - Async/Awaits
  - Packages and local environments
    - Understanding package.json and npm (package manager)

Callback Examples

```
bigOak.readStorage("food caches", caches => {
  let firstCache = caches[0];
  bigOak.readStorage(firstCache, info => {
    console.log(info);
  });
});


let caches = bigOak.readStorage("food caches")
let firstCache = caches[0];
// Not defined
```

Promises Examples

```
function storage(nest, name) {
  return new Promise(resolve => {
    nest.readStorage(name, result => resolve(result));
  });
}

storage(bigOak, "enemies")
  .then(value => console.log("Got", value));
```

Async Await Examples

```
async function myFunction() {

  let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("done!"), 1000)
  });

  let result = await promise; // wait until the promise resolves (*)

  console.log(result); // "done!"
}

myFunction();
```

package.json

- If you are building a Node.js project and want it to be used by others who should be able to:
  - Find it: Name, tags
  - Install it
  - Install its dependencies
  - Get further information on it
    - Version
    - Human readable description
    - Homepage
    - License
    - Author and contributors
- This is done via package.json file for Node.js projects. More info: https://docs.npmjs.com/files/package.json

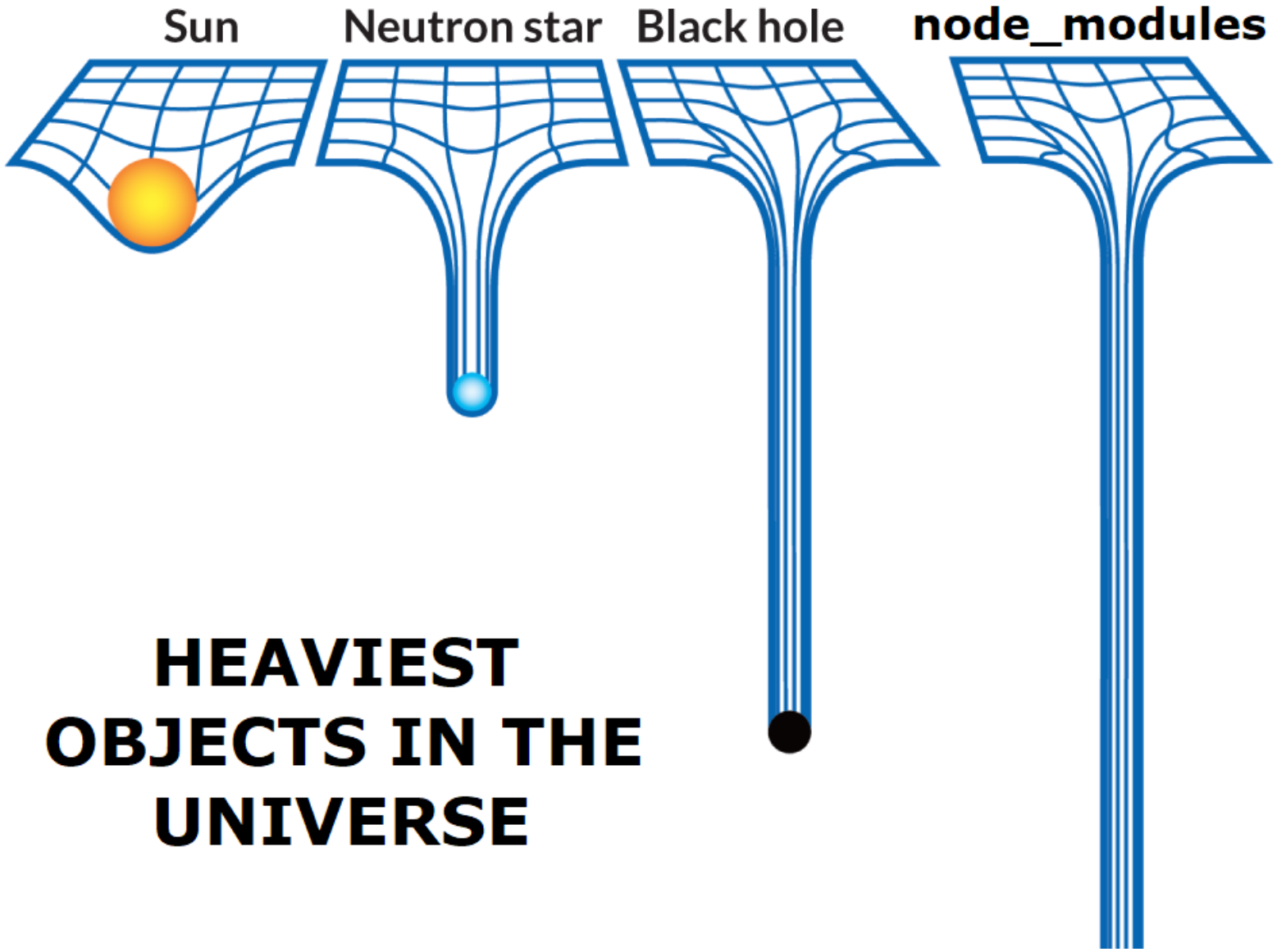
```
{
    "name": "wot-testbench",
    "version": "1.0.1",
    "repository": {
        "type": "git",
        "url": "https://github.com/tum-esi/testbench.git"
    },
    "license": "MIT",
    "dependencies": {
        "@node-wot/binding-coap": "0.6.2-SNAPSHOT.1",
        "@node-wot/core": "0.6.2-SNAPSHOT.1",
        "ajv": "6.10.2",
        "chai-http": "^4.3.0",
        "del": "5.1.0",
        "json-schema-faker": "0.5.0-rc15",
        "mkdirp": "0.5.1",
        "xmlhttprequest": "^1.8.0",
        "inquirer": "^7.0.0"
    },
    "devDependencies": {
        "mocha": "^6.2.0",
        "chai": "^4.2.0",
        "typescript": "3.3.1"
    },
    "scripts": {
        "build": "tsc",
        "start": "node dist/wot-test-bench.js",
        "clean": "node clean.js",
        "test": "mocha --exit",
    }
}
```

package.json

- If you are familiar with pip and python but not sure how it happens there, a nice guide is available at https://medium.com/@joel.barmettler/how-to-upload-your-python-package-to-pypi-65edc5fe9c56
- You will see that it is a lot more work in python and not really standardized

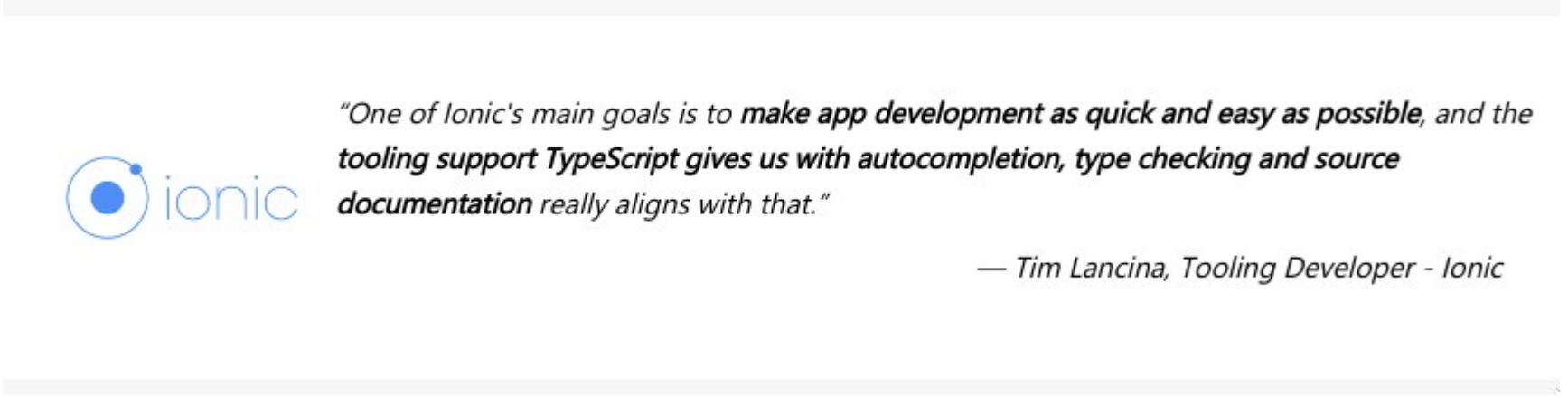
npm

- npm is a Node.js specific package manager, comparable to Linux package managers or PyPi (pip)
- It stores your JS files for others to find
- It relies on package.json files
- Can and should be installed in your system, along with Node.js
- Conventions:
  - `npm install my-package` installs the package called `my-package`. Unless you do `npm install -g my-package` , the package is installed locally in the folder you run it from.
  - `npm install` installs the dependencies based on the package.json in the current directory. The installed files go into `node_modules` folder of the current directory. Their dependencies will also be installed and go into the same folder.


Sun
Neutron star
Black hole
node_modules
HEAVIEST OBJECTS IN THE UNIVERSE

Typescript

- We had said that JavaScript was created „to do *small* programs that would help with animations, submitting forms, etc.“
  - What if the project gets big? Multiple contributors, different abstraction layers
    - Class definitions, interfaces
    - Type safety
    - Verification of the source code for code style (**not** the amount of spaces)



Examples

```
function greeter(person: string) {
    return "Hello, " + person;
}

let user = [0, 1, 2];

console.log(greeter(user));
```

```
function greeter(person: string) {
    return "Hello, " + person;
}

let user: string = "Joe";

console.log(greeter(user));
```

Examples

```
1 class Student {
2     fullName: string;
3     constructor(public firstName: string, public middleInitial: string, public lastName:
  string {
4         this.fullName = firstName + " " + middleInitial + " " + lastName;
5     }
6 }
7
8 interface Person {
9     firstName: string;
10    lastName: string;
11 }
12
13 function greeter(person: Person) {
14    return "Hello, " + person.firstName + " " + person.lastName;
15 }
16
17 let user = new Student("John", "M.", "Doe");
18
19 console.log(greeter(user));
20
```

Examples

```
1 "use strict";
2 class Student {
3 }
4 function greeter(person) {
5     return "Hello, " + person.firstName + " " + person.lastName;
6 }
7 let user = new Student("John", "M.", "Doe");
8 console.log(greeter(user));
```

Tutorials

- JavaScript:
  - https://eloquentjavascript.net/ : Very reputable source for JS programming
- Typescript:
  - Natural way (https://news.ycombinator.com/item?id=14648064)

    ```
    changed all my .js files to .ts, turned on all of TypeScript's
    checks, and started adding types to my files.
    ```

  - Typescript‘s official 5 minutes tutorial: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html
  - You can learn TS and JS from the Typescript Playground: https://www.typescriptlang.org/play/

Tutorials

You can also do something more practical. Here is a small task that involves many aspects of Node.js

You want a script that can go to GitHub repository based on a configuration file, find the contributor with 3rd most commits and log a congratulations message. This means:

- Reading a file from the filesystem (fs is a standard library)
- Validating it against a JSON Schema (use the library ajv for this)
- Understand the GitHub API (https://developer.github.com/v3/) to build the required request
  - There are many libraries to do the request, use as you wish
- Choose a repository with at least 3 contributors
- Parse the response and get to the contributor with 3rd most commits
- Log a message with his/her name

Wrap-Up

- Node.js is a framework for the JavaScript language that allows it to run in non-browser environments
- It has some unusual features to handle asynchronous calls
- Programming in JavaScript is being replaced with TypeScript which has more safety related features