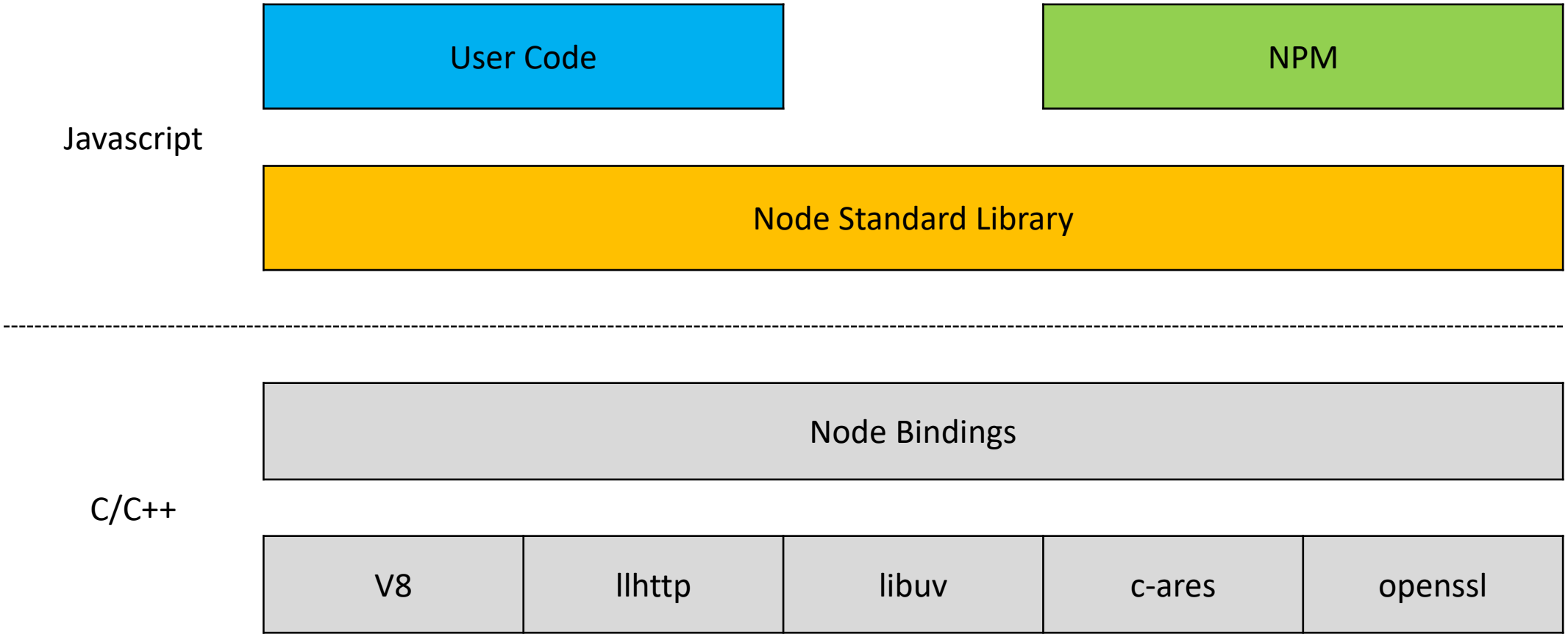
Node.js

MEDT4

htl Hollabrunn

Architektur



Installation

- Projekte werden mit unterschiedlichen Node-Versionen erstellt
- eingesetzte Entwicklungswerkzeuge benötigen eventuell spezifische Node-Versionen

- es ist daher von Vorteil, wenn alle Node-Versionen zur Verfügung stehen und schnell umgeschalten werden kann

→ Node Version Manager (nvm)

Installation

1. Stelle sicher, dass Node nicht installiert ist:

```
> node -v

node: The term 'node' is not recognized…
```

2. Installiere `nvm-setup.exe` https://github.com/coreybutler/nvm-windows/releases

   - Alternativ für Linux: https://github.com/nvm-sh/nvm

3. Überprüfe die Installation

```
> nvm

Running version 1.1.10.
Usage:…
```

htl Hollabrunn

Installation

4. Aktuellste Node-Version installieren

```
> nvm install latest

Downloading node.js version 19.8.1 (64-bit)...
```

5. LTS-Version installieren

```
> nvm install lts

Downloading node.js version 18.15.0 (64-bit)...
```

htl Hollabrunn

Node Version Manager - nvm

- erlaubt den Wechsel zwischen verschiedenen Node-Versionen
- dabei wird das Installationsverzeichnis (z.B. C:\Program Files\nodejs) als Symlink angelegt.

```
PS C:\Program Files> Get-ChildItem -Filter nodejs

    Directory: C:\Program Files

Mode  LastWriteTime     Length Name
----  -------------     ------ ----
l---  28/10/2022 14:40         nodejs -> C:\Users\marti\AppData\Roaming\nvm\v19.0.0
```

Node Version Manager - nvm

- Initiale Versionsauswahl/ Versionswechsel: `nvm use 19.0.0`
  (*setzt den Symlink-Pfad)*
- Installierte Versionen anzeigen: `nvm list`
- Aktuelle Auswahl anzeigen: `nvm current`
- Spezifische Version installieren: `nvm install x.x.x`

htl Hollabrunn

npm – Node Package Manager

`npm install <modul/package>`

`-g`	installiert das Modul „Global“, dadurch ist es in jedem node-Projekt verfügbar **(handle with care)**
`npm init`	erstellt Package in aktuellem Verzeichnis → `package.json`
`-y`	ohne Usereingabe (defaults)

htl Hollabrunn

npm – Node Package Manager

```
npm uninstall <modul/package>
```

`npm --version`	gibt aktuelle Version von npm
`npm list`	listet alle installierten Packages auf
`npm list --depth=2`	Darstellung der Abhängigkeiten

npm – Node Package Manager

`npm outdated`	zeigt Module an, die nicht in der aktuellsten Version installiert sind
`npm update`	aktuallisiert alle Module auf die neueste Version (ausgenommen fixierte Versionen)

package.json

- name
  - <= 214 Zeichen
  - keine Großbuchstaben
  - Name muss URL-fähig sein, daher nur „URL-safe characters“
- scripts
  - start - Ausführendes skript (npm start)
- dependencies
  - Listet installierte module auf
- type: module (Aktivierung Modulsystem ECMAScript)
- private: true
  - npm publish → Error EPRIVATE … verhindert versehentliches publishen nach npmjs.org

Read-Eval-Print-Loop

- Starten

```
> node
```

- Read – liest Kommando ein z.B. `> http.STATUS_CODES`
- Eval – führt Kommando aus
  - Kern-Module müssen nicht erst geladen werden
- Print – gibt Ergebnis (Rückgabewert) aus

- Beenden
  - Strg+C x2
  - Strg+D
  - `.exit`

Read-Eval-Print-Loop

- Hilfe für wichtigsten Kommandos

```
> .help
```

- Sessions sind „flüchtig“

```
> .save session.js // Speichert Input einer Session
> .load session.js // Lädt den Fileinhalt in die Session
```

Hollabrunn

Ausführung Node-Apps

```
node index.js
```

- übersetzt js-Code in Byte-Code und führt diesen aus

```
npm install -g nodemon
nodemon index.js
```

- Hot-Reloading
  ändert sich der Inhalt eines Files im aktuellen Verzeichnis, startet nodemon den node-Prozess neu

ab Version >= 18.1 ist Hot-Reloading auch via Node-Parameter `--watch` verfügbar:

```
node index.js --watch
```

Hollabrunn

Debugger

```
>node --inspect index.js
```

oder

```
>node --inspect-brk index.js
// brk … break (on first line)
```

```
Debugger listening on
ws://127.0.0.1:9229/95a56c06-41ce-47bf-97ff-
9f49ab995b46
For help, see:
https://nodejs.org/en/docs/inspector
```

Debugger

js-Code:

```
break    // Haltepunkt wenn debugging aktiv
```

Coding Demo - nodejs-basics

Modulsystem - CommonJS

- Ursprüngliches Node.js-Modulsystem
- Exportieren
  - `module.exports`
  - `exports`
- Importieren – keine Dateinamenerweiterung erforderlich
  - `require('./filename')`
  - `require('./dirname')`
    - automatischer Import eines index-Files, falls vorhanden
- beliebige Strukturen können importiert/exportiert werden
- Konstanten __dirname und __filename

Modulsystem - CommonJS

- index.js
  ```
  require(`./user`);
  ```
  Module wird importiert

- user.js
  ```
  class User {…}
  module.exports = User;    // exports = User;
  ```
  User wird als Module exportiert

Modulsystem - ECMAScript

- Aktivierung
  - Dateityp
    - `.mjs`
  - package.json
    - `type: module`
  - Commandline Argument
    - `node –input-type=module`

Modulsystem - ECMAScript

- Exportieren
  - Named: `export class User {…}`
  - Default: `export default class User {…}`
- Importieren
  - Named:
```
import { User } from `./user.mjs`
import { User as CustomName } from `./user.mjs`
```
  - Default:
```
import User from `./user.mjs`
import CustomName from `./user.mjs`
```

Filesystem fs

https://nodejs.org/api/fs.html

3 Varianten:

- Synchron
- Asynchron
- Promises

Coding Demo – Filesystem-Module

Filesystem fs - synchron

```
// import fs module
import { readFileSync } from 'node:fs';


// read from input.txt synchronous into fileContent
const fileContent = readFileSync('./input.txt');


// log fileContent to console
console.log('FileContent: ${fileContent}');
```

Filesystem fs - asynchron

```
import { readFile } from 'node:fs'

readFile('./input.txt', (error, data) => {
    if (error !== null) {
        console.error(error.message);
    }
    console.log(data.toString());
});
```

Filesystem fs - asynchron

```
import { readFile } from 'node:fs'

readFile('./input.txt', (error, data) => {
    if (error !== null) {
        console.error(error.message);
    }
    console.log(data.toString());
});
```

- verarbeitet das Ergebnis einer asynchronen Funktion
- kann auch zur Erweiterung bestehender Funktionen dienen (wird gerne in Libraries angeboten)

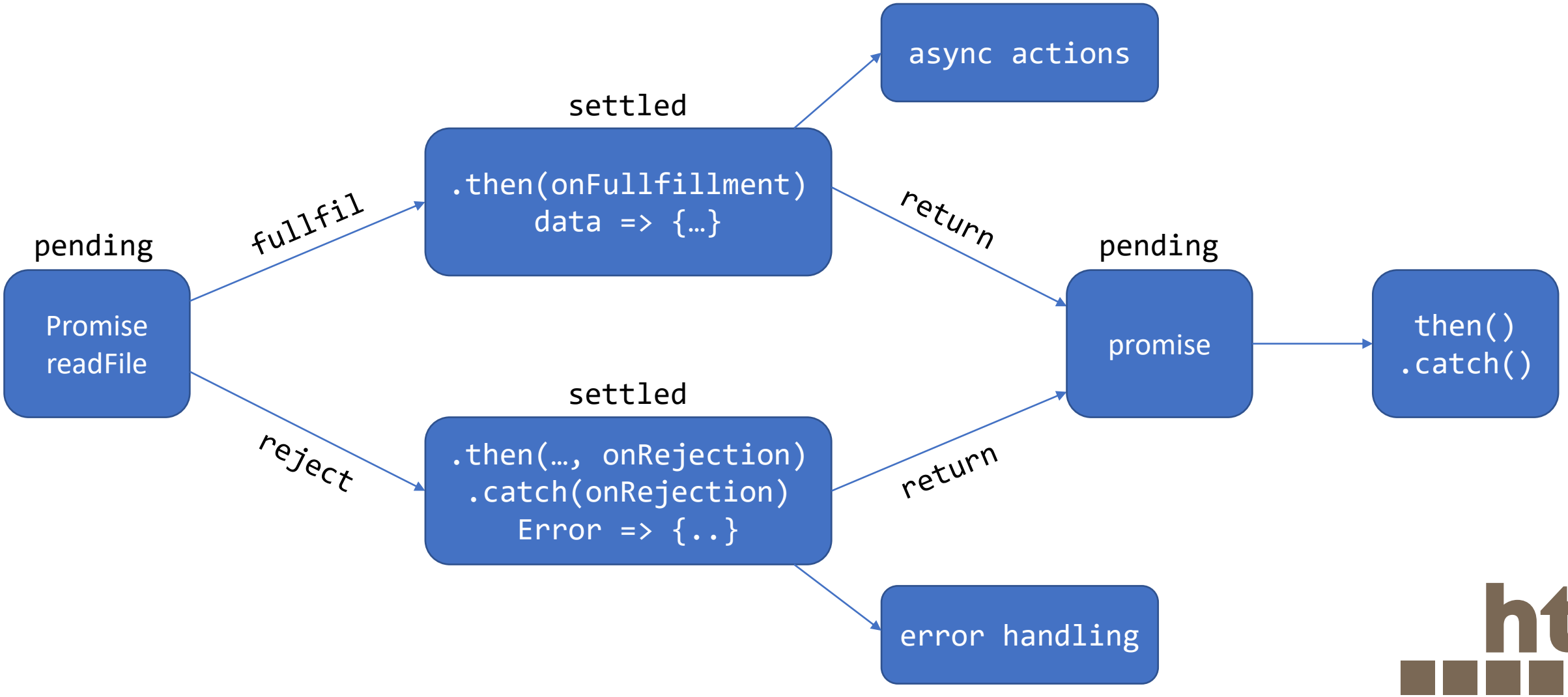
Hollabrunn

Promises *(übersetzt Versprechen, Zusagen)*

- Ein Promise ist ein Konstrukt in Javascript, das festlegt, was passieren soll, wenn eine zeitgesteuerte Operation stattgefunden hat. Sie steuert ein Versprechen bei, um die Abläufe asynchroner Anweisungen zu vereinfachen.

- Ein Promise wird nur einmal aufgelöst und mehrere Versprechen können einfach miteinander verkettet werden.

Promises *(übersetzt Versprechen, Zusagen)*



Promises *(übersetzt Versprechen, Zusagen)*

- Ein Promise gilt als "`fulfilled`" wenn `promise.then(f)` die Funktion "`f`" frühestmöglich aufruft.
- Ein Promise gilt als "`rejected`" wenn `promise.then(undefined, r)` die Funktion "`r`" frühestmöglich aufruft.
- Ein Promise gilt als "`settled`" wenn er entweder "`fulfilled`" oder "`rejected`" ist.
- Ein Promise gilt als "`pending`" wenn er weder "`fulfilled`" noch "`rejected`" ist.

Filesystem fs - promise

```
import { readFile } from 'node:fs/promises'

readFile('./input.txt', 'utf8')
    .then((data) => {
        console.log(data);
    })
    .catch((error) => console.error(error));
```

Filesystem fs – promise (async & await)

```
import { readFile } from 'node:fs/promises'

try {
    const data = await readFile('./input.txt', 'utf8');
    console.log(data);
} catch (error) {
    console.error(error);
}
```

Coding Task 

- implementiere Web-Anfragen an https://randomuser.me/api/ mittels **fetch-API** https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

- verwende dazu die latest-Version von node
  - diese beinhaltet die fetch-API
  - alternativ kann node-fetch in node-Versionen < 18 installiert werden

- implementiere die Requests in allen Promises-Varianten:
  - `then().catch()`
  - `async/await`

Coding Task 

- die Ausgabe sollte wie folgt aussehen (Vorname und Nachname sind durch die empfangen Daten zu ersetzen):

```
1.) Promise using then().catch(): Vorname Nachname
2.) Promise using Async/Await: Vorname Nachname
```

Express

- Web-Framework für node.js
- schnelle Entwicklung von Web-APIs (REST)

```
npm install express
```