Assignment 3

Due: Wed 13.11.2019; 18:00h (1 Week)

Goals

After doing these exercises, you know have a more understanding of JavaScript regarding ...

- EventLoop
- TypeScript

Task 1: Event Loop

Difficulty: Easy

In the tutorial, you learnt the concept of event loop in JavaScript. An event loop has one or more task queues. More precisely, the task queues in the event loop involves two types of tasks: macrotask and microtask. The event loop will have exactly one task being processed from the microtask queue. After a microtask has finished, all available microtasks are processed, they can queue even more microtasks, which will all be run one by one, until the microtask queue is exhausted. For example, the following tasks:

- XMLHttpRequest callback
- Event callback (onClick)
- setTimeout/setInterval
- history.back

are macrotasks. Instead, the following tasks will be pushed to microtask queue:

- Promise.then
- MutationObserver
- Object.observe

Given the following code snippets:

```
setTimeout(() => {
  console.log('timer')
  Promise.resolve().then(() => console.log('promise1'))
}, 0)

Promise.resolve().then(() => {
  console.log('promise2')
  Promise.resolve().then(() => console.log('promise3'))
})
console.log('script')
```

Please figure out its output and explain the reason of the output. Write your answer in a .txt file and include it in your submission in a folder “task01”.

Task 2: Abstraction in TypeScript. Difficulty: Easy

Imaging two type of news categories:

1. Political news has properties: title(string), content(string), author(string), country(string), and publishedAt(number)
2. Entertainment news includes title(string), content(string), url(string), publishedAt(number)
3. The country field in political news may be empty, and url field in entertainment news may also be empty.

Please write an appropriate TypeScript interface that abstracts above news categories. Include your answer in a .ts file and include the file in your submission in a folder “task02”.

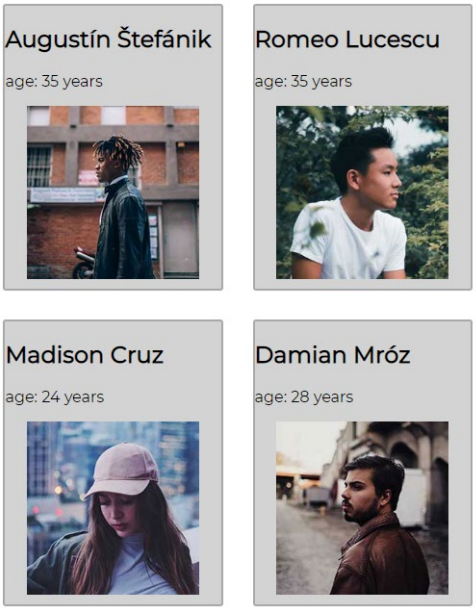
Task 3: (Fictional) User Importer in TypeScript Difficulty: Medium

If you open the webpage *omm-user-import* folder from our GitHub repository, you will find a page that loads some dummy user data from the internet (another example for an interesting API ☺), and then “imports” them in its (fictional) database.

If the import was successful, the webpage will display users’ profile cards, (see Figure 1). You should be able to compile the typescript via `npm run build` (then you can open `index.html`) .

Unfortunately, some parts of this page’s code are incomplete. As a fresh TypeScript expert, now it’s your turn! Complete code in *omm-user-import* folder, make the code work properly as described in above (Look for the according TODO comment).

Include all code files in your submission in a folder “task03”.



Submission

Please turn in your solution as ZIP file via Uni2Work. You can form groups of up to three people.

We encourage you to sign up for Slack! All you need is a CIP account and an email address that ends in “@cip.ifi.lmu.de”. Ask us if you don’t know how to get them.

If you have questions or comments before the submission, please contact one of the tutors. They are on Slack @Aleksa and @Andre, remember that they also want to enjoy their weekends ☺

It also makes sense to ask the question in our #omm-ws1920 channel. Maybe fellow students can help or benefit from the answers, too!