CS 4350: Fundamentals of Software Engineering
CS 5500: Foundations of Software Engineering

Supplementary Slides for 2/11/21

Mitch Wand

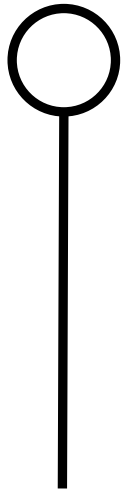
Khoury College of Computer Sciences

URLs vs URIs

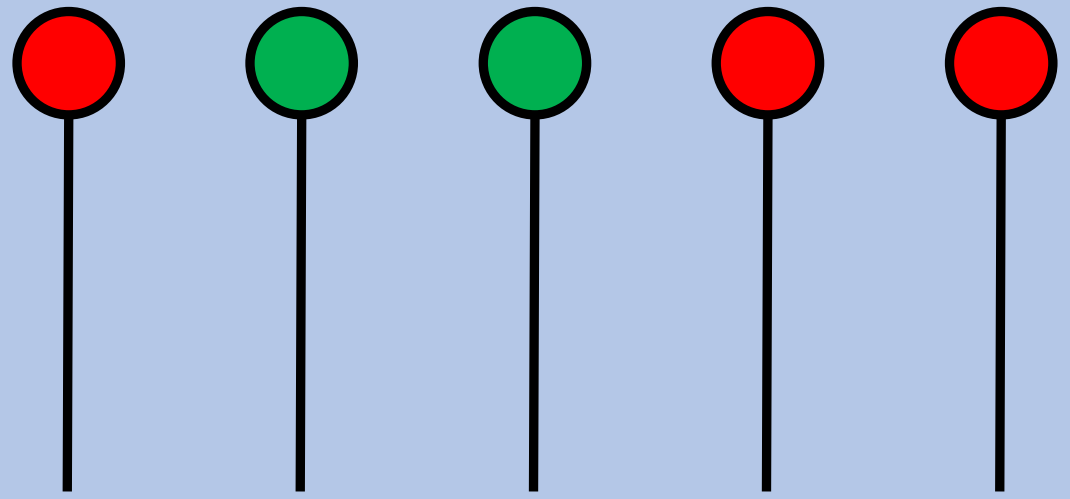
- Short answer: a URI is a name for an object, a URL is an address that tells you how to get to the object.
- A URL will always have an access method, eg http://foo.bar.example ;
- Every URL is a URI; a URL may be located by 0, 1, or more URLs.
- The gory details:
  - https://tools.ietf.org/html/rfc3986 (esp Sec 1.1.2)
  - https://tools.ietf.org/html/rfc8820
- Even shorter answer: don't worry about it.

When the running event handler completes, the scheduler chooses one of the other ready event handlers to execute

The running event handler



The Pool of Waiting event handlers



How can a handler become ready?

- There are roughly 3 ways in which a handler can become ready:
  - it can become ready at a specific time.
  - it can become ready when some input/output event occurs
  - it can become ready when some other handler or handlers complete.

What happens when you create a promise?

- When you say:

```
const p1 = new Promise ...
```

  - A new event handler is created and thrown in the pool.
  - The variable p1 is bound to a representation of that event handler
  - The handler that is creating p1 continues to execute.

What happens when you say "await"?

- When you say:

```
await p1
```

  - the current handler suspends itself.
  - the remainder of the current handler is thrown into the pool, with notation to say that it will be ready to run when p1 completes successfully.
  - the scheduler chooses some ready event to execute.
    - Maybe p1, maybe not.

What happens when you say p1.next{...} ?

- When you say p1.next{....}
  - a new promise is created.
  - the new promise is marked as being ready to execute then p1 completes successfully
  - the current handler continues to execute.

What happens when you say p1.catch{...} ?

- When you say p1.catch{....}
  - a new promise is created.
  - the new promise is marked as being ready to execute then p1 completes ~~successfully~~ by throwing an error.
  - the current handler continues to execute.

What's the difference between a promise and an event handler?

1. A promise is the representation of an event handler in a program.
2. A promise survives even after its handler has finished executing.
   - so, for example, in the following code

```
await p1(...);
p1.next(...)
```

   - the code in the next will be executed.

Better names for the layers in our client

index.ts : contains scripts to be executed.
Calls: getTranscript, getStudentIDs, etc., corresponding to the REST endpoints

client.ts

dataService.ts: provides REST endpoints
exports: getTranscript, getStudentIDs, etc.

endpointService.ts

remoteService.ts : provides http methods
exports: remoteGet, remotePost, etc.

httpService.ts

axios: an npm package that actually does the http work
provides: axios.get, axios.post, etc

You could put the scripts in a separate file, and just call them from client.ts