Valderi Leithardt, Dr.

IPW (Fifth week class notes)

Summary

- Review of previous class *
- HTTP
- Exercises

Programação assíncrona (*Review previous class)

Conceitos Promises

- É importante: Promises são objetos. Consequentemente, sempre que instanciarmos uma nova Promise, ela deverá estar acompanhada da palavra new.

- Utilizamos Promises para processamento assíncrono.
- Promisses recebem duas funções callback, que são funções que chamam outras funções: a resolve e a reject.
- Usamos essas funções, respectivamente, quando a Promise é resolvida e quando ela é rejeitada, ou ocorre erro. Qualquer uma das duas situações chama uma função correspondente.

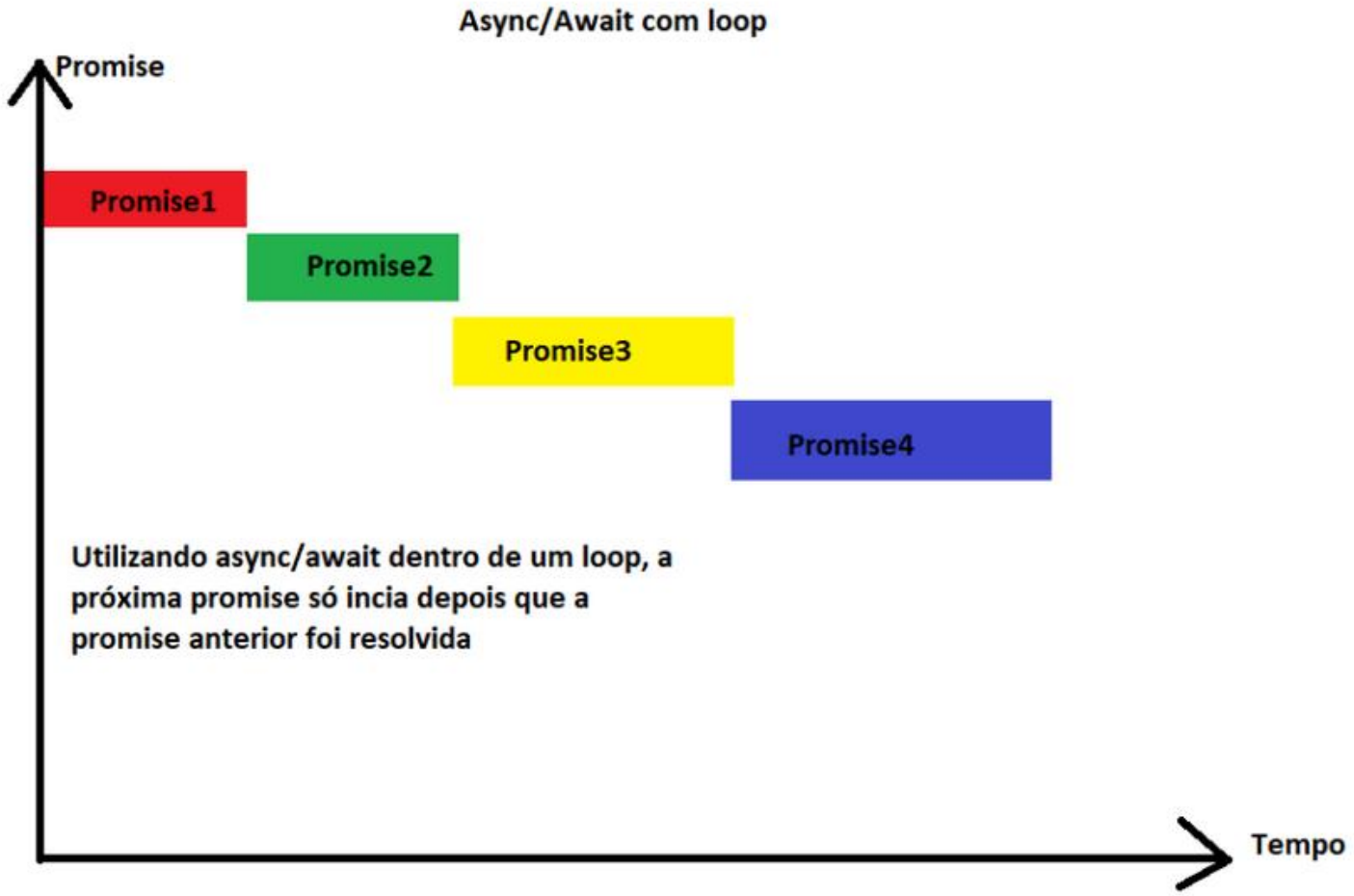
- Promises contam também com diversos métodos, sendo o foco aqui o método then() e o método catch().
- **O método then()** possui dois argumentos, que também são funções callback resolve e reject.
- Enquanto o then() lida com ambos casos (tanto se a Promise for resolvida quanto rejeitada), o catch() lida apenas com casos de rejeição.

- Por sempre retornarem Promises, podemos encadear tanto o then() quanto o catch(), embora geralmente utilizamos o catch() por último na linha de código, pois ele trata dos erros → depois de vários then's, o catch é utilizado para "coletar" todos os erros.

Programação assíncrona (*Review previous class)

Conceitos async/await

- Uma função assíncrona pode conter uma expressão await, que pausa a execução da função assíncrona e espera pela resolução da Promise passada, e depois retoma a execução da função assíncrona e retorna o valor resolvido.



Programação assíncrona (*Review previous class)

Conceitos async/await

- O **método Promise.all()** recebe um iterável de promesses como entrada e retorna uma única Promesse que resolve para uma matriz dos resultados das promesses de entrada.

- Essa promise retornada será cumprida quando todas as promises de entrada forem cumpridas ou se o iterável de entrada não contiver promises.

- Então rejeita imediatamente após qualquer uma das promises de entrada rejeitando ou não promises lançando um erro, e irá rejeitar com esta primeira mensagem/erro de rejeição.

- Em resumo: **o método Promise.all** recebe um array de promises e retorna uma única promise que só irá ser resolvida quando todas as promises do array que foi passado forem resolvidas.

* Outros exemplos e definições (2023).

HTTP - Hypertext Transfer Protocol

- O protocolo HTTP 0,9 foi desenvolvido pela primeira vez no final dos anos 80, mas com uma baixa capacidade
- A combinação dessa invenção com HTML e URLs é agora considerada como a base da iniciativa global de informação na rede mundial de computadores
- Essa inovação foi levada ao mundo da web por Tim Berners-Lee no CERN de Genebra para a troca de informações entre a comunidade física.
- Devido as deficiências do protocolo HTTP 0,9, Berners-Lee pensou em melhorias ao inventar a primeira versão real do HTTP/1.0 em 1991
- Esse novo trabalho foi proposto como RFC 1945 ao órgão regulador da Internet Engineering Task Force (IETF) em 1996.
- História e definições em W3C 1991 e Definição da Arquitetura HTTP em W3C

* Lista de códigos de estado HTTP

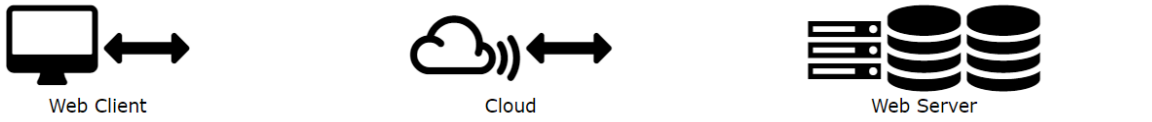
HTTP Conceitos

- HTTP é a abreviação do **protocolo** de transferência de hipertexto. Esse é o principal método pelo qual os dados de páginas da Web são transferidos na rede mundial de computadores. As páginas da Web são armazenadas em servidores, que depois são disponibilizadas no computador cliente à medida que o utilizador acede.

- Protocolo <= É um conjunto de regras de comunicação

- A rede resultante desses protocolos é a rede mundial de computadores (world wide web), como a conhecemos hoje. Ou seja, sem HTTP, a world wide web (WWW) como conhecemos não existiria.

Figura extraída de W3S, (2023).



- Em uma conexão HTTP há um grande problema: os dados transferidos através de uma conexão desse tipo não são criptografados. Todas as informações transmitidas através desta rede via HTTP não são privadas, portanto, qualquer dado de cartão de crédito e informações confidenciais não devem ser enviados se estiver em uma página HTTP. Segurança => HTTPs

HTTP - Endereçamento

- ❑ Todo device tem um endereço IP
- ❑ Toda informação da web é identificada por uma URI.

✓ URL: Uniform Resource Locator

→ URL: localização http://www.isel.pt/

* De acordo com descrito cap 13 do Livro

✓ URI: Universal Resource Identifier

→ URI: conceito <protocolo>://<localizacao>/<caminho>/<recurso>

✓ URN: Uniform Resource Name

→URN: nome <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:example"

HTTP – 1.1_Teoria

- HTTP GET: Requisição de informações especificadas pela URI
- HTTP POST: Requisita que o servidor aceite as informações enviadas para a URI
- HTTP PUT: Requisita que o servidor armazene as informações enviadas para a URI
- HTTP DELETE: Requisita que o servidor apague o recurso identificado pela URI

HTTP – 1.1_Pratica

- As linguagens server-side aumentaram o nível de abstração da comunicação, restando dois métodos HTTP realmente usados
- HTTP GET: Aparece na URI do browser

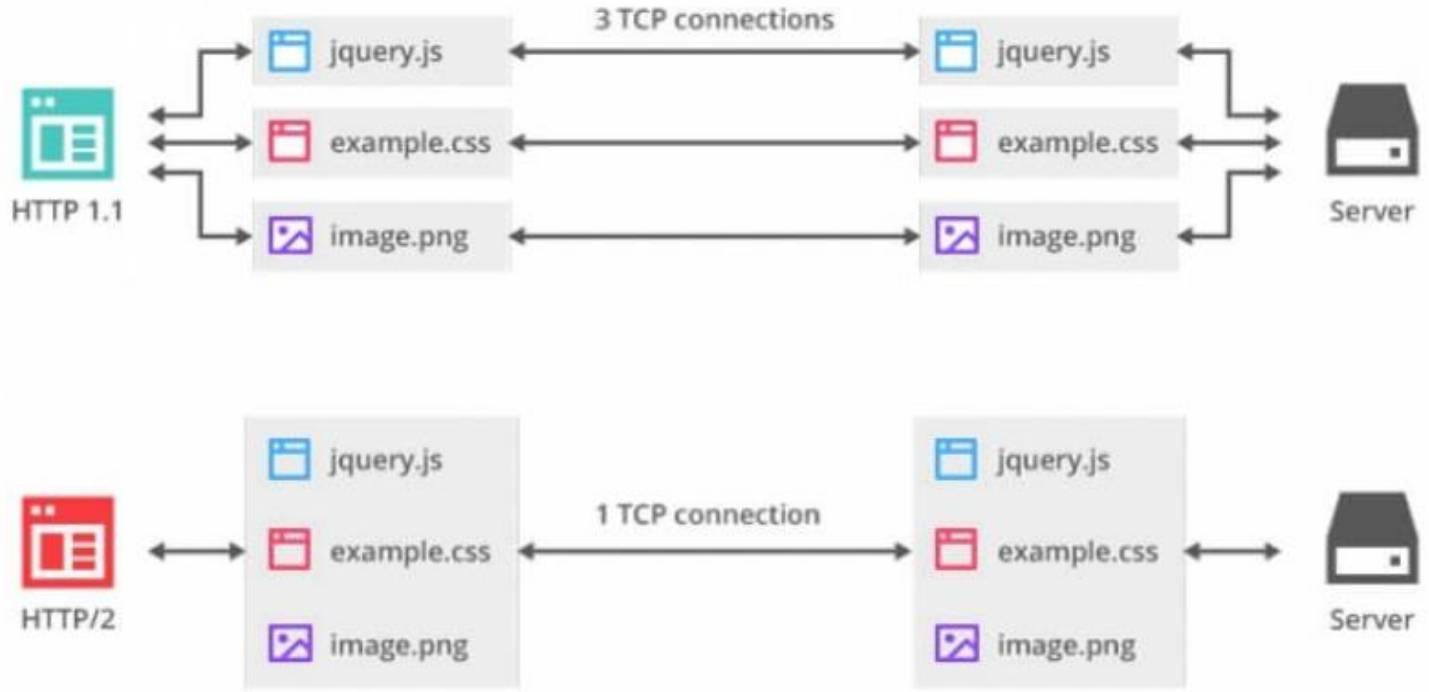
  → Ex: Busca do Google
- HTTP POST: Não aparece na URI do browser

  → Ex: A visualização do seu extrato no banco

HTTP – 1.1_Pratica

- SEF (Search Engine Friendly): URL amigável
  - → EX: https://www.isel.pt/candidatos/porque_o_ISEL

- Funcionalidades
  - → SEO (Search Engine Otimization): Otimização para mecanismos de busca
  - → Número de links, redes sociais, SEF, etc…



Protocolo HTTP/2

- O HTTP/2 foi desenvolvido para resolver alguns problemas que existiam na versão 1.1, principalmente com relação à performance e segurança.

Multiplexação de mensagens

- No HTTP/1, uma conexão é utilizada para baixar um único ficheiro, tornando o processo de download de conteúdo, sequencial.
- No HTTP/2 implementou a multiplexação de mensagens. Com a multiplexação, o navegador não precisa mais esperar receber um recurso para poder requisitar os recursos seguintes.
- Com o HTTP/2, o navegador pode solicitar e ter a resposta de vários recursos de forma paralela através de apenas uma conexão.
- Houve redução no consumo de processamento e memória, uma redução no custo operacional das redes e uma maior capacidade de uso. O resultado geral foi redução de latência na rede e redução de custo em hardware e software.

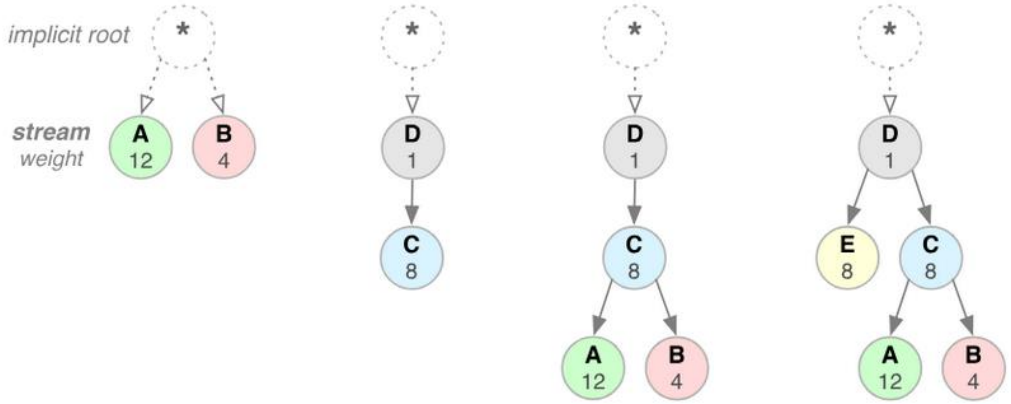
Protocolo HTTP/2

Camada de enquadramento binário

Este é o núcleo de todas as melhorias de desempenho de HTTP/2, que determina como as mensagens HTTP são encapsuladas e transferidas entre o cliente e o servidor.

Priorização de requisições

Possibilita ao navegador requisitar todos os elementos quando descobertos, comunicando ao servidor sua intenção de prioriza algum deles. Isto é realizado nas definições e pesos das streams



Server Push

Permite que o servidor enviar múltiplas respostas ao cliente, a partir de uma única requisição, sem que o cliente tenha solicitado explicitamente.

Segurança e criptografia de dados

O protocolo HTTP/2 foi implementado para trabalhar com ou sem criptografia. No entanto, os principais navegadores declararam que suportariam HTTP/2 somente com criptografia, sendo necessário então a utilização de um certificado SSL.

Protocolo HTTP/2

- **Códigos de status de respostas HTTP**

De acordo com MDN 2023, os códigos de status de resposta HTTP indicam se uma solicitação HTTP específica foi concluída com êxito. As respostas são agrupadas em cinco classes:

I. Respostas Informativas (100 – 199)
II. Respostas bem-sucedidas (200 – 299)
III. Mensagens de redireccionamento (300 – 399)
IV. Respostas de erro do cliente (400 – 499)
V. Respostas de erro do servidor (500 – 599)

* Conforme definição RFC 9110 (Acessado em 2023).

- **Métodos de requisição HTTP**

O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso. Embora esses métodos possam ser descritos como substantivos, eles também são comumente referenciados como HTTP Verbs (Verbos HTTP). (* Conforme descrito em MDN (2023).

Verbo HTTP	Ação	Geral (p.ex. /notas)	Específico (p.ex /notas/{id})
POST	Criar	201 Created	404 Not Found 409 Conflict
GET	Ler	200 OK	200 OK 404 Not Found
PUT	Atualizar	405 Method Not Allowed	200 OK 204 No Content 404 Not Found,
PATCH	Modificar parcialmente	405 Method Not Allowed	200 OK 204 No Content. 404 Not Found
DELETE	Excluir	405 Method Not Allowed	200 OK 404 Not Found

Tabela 1. Ação em CRUD

- **Referência rápida para APIs**

A Tabela 1 a seguir apresenta cada ação em CRUD (acrónimo para Create, Read, Update, and Delete) que pode ser implementada por um servidor e os códigos de retorno associados a elas.

Exercises

- 2º Trabalho disponibilizado em:

https://github.com/isel-leic-ipw/2324i-IPW-LEIC31D/wiki/IPW_IP-2324-1-A1

Cap 18 – livro ENG: https://eloquentjavascript.net/18_http.html

* Concluir exercícios da aulas anteriores.

References

- https://eloquentjavascript.net/18_http.html
- https://www.tutorialspoint.com/http/index.htm
- https://developer.mozilla.org/pt-BR/docs/Web/HTTP
- https://www.w3schools.com/whatis/whatis_http.asp
- Também foram realizadas adaptações e modificações com base no material disponibilizado por Professor Luís Falcão, acesso online em: https://github.com/isel-leic-ipw/
- Aulas gravadas Professor Falcão:
- Part 1: https://videoconf-colibri.zoom.us/rec/share/Eg1oe-3PrY0AlgZYr73zFklnQ1JKYjYUW6yEPMmMKTaR2XUjMJFN8EC1Wecz6MJE.a2RysTo8b7MGtHqP
- Part 2: https://videoconf-colibri.zoom.us/rec/share/dvJUsndNkc1krQS7d-mDPwBwEkrKJvsJUBVRhOpnvjeyFLLf1cX8iK0rHIJWk-HN.V6vNOr6wxMdvbGFz
- Part 3: https://videoconf-colibri.zoom.us/rec/share/UcN_r6j6VWkC6MzCfXtT1jJgIkbBK0lAZUO51I0PL11dSKahV7J7YnjInGlJ6MLt._3RozFUJ_JwsdmrX
- https://videoconf-colibri.zoom.us/rec/share/XUBZUISQJmtNGwbDoN09ExwritlM7qimhueHokLpZABwPrtSJwNDIPi_4inXUdDm.T0aW2Tj2VbbCH27i

Valderi Leithardt, Dr.

Professor IPW

valderi.leithardt@isel.pt