Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores e Licenciatura em Engenharia Informática, Redes e Telecomunicações

Introdução à Programação na Web / Programação na Internet

Inverno de 2023/2024

* Teste Global de Época Normal (10 de Janeiro de 2024) - Duração 1h30

NÚMERO _____ **NOME** _____

GRUPO 1 [4 valores]

Nas seguintes questões, selecione a única opção verdadeira. Uma resposta correta conta a totalidade da cotação da pergunta; uma resposta incorreta desconta 25% da cotação da pergunta ao total do grupo; uma questão não respondida conta 0 valores.

1. O que é um *closure* na linguagem JavaScript?
   A. Uma forma de proteger as variáveis de um objeto
   B. Uma característica da linguagem que liberta o espaço de variáveis não usadas, de modo a otimizar a memória utilizada
   C. Um mecanismo da linguagem para criação de métodos privados num objeto
   D. Uma forma de encapsular o ambiente léxico de uma função, permitindo o acesso às suas variáveis, mesmo depois da função retornar

2. Ao analisar a URL *http://www.example.com/products?id=123&category=electronics*, qual das seguintes afirmações é verdadeira?
   A. O path neste URL é *id=123&category=electronics*
   B. A query string contém dois parâmetros, *id* e *category*
   C. *www.example.com* identifica um recurso
   D. Nenhuma das opções anteriores está correta

3. Ao lidar com um pedido para um recurso que o cliente não tem permissões para aceder, apesar de apresentar credenciais válidas, o *status code* que deve ser retornado é:
   A. 400
   B. 403
   C. 404
   D. 500

4. Um pedido HTTP idempotente:
   A. Pode alterar o estado da aplicação
   B. Pode ser repetido várias vezes
   C. pode ter os métodos: DELETE, GET ou PUT
   D. Todas as anteriores

5. Os cookies HTTP servem para:
   A. Guardar informação sensível entre o cliente e o servidor
   B. Autenticar o cliente perante o servidor
   C. O servidor guardar dados no cliente e obtê-los futuramente
   D. Determinar a localização do cliente

6. Qual o propósito do npm no processo de desenvolvimento em Node.js:
   A. Simplificar a instalação do ficheiro package.json
   B. Disponibilizar um sistema de controlo de versões para o código JavaScript
   C. Simplificar a instalação, partilha e gestão de módulos de *third-party*
   D. Gerar a documentação automática de APIs

7. Qual é o principal propósito do header `Content-Type` do protocolo HTTP?
   A. Especificar o tipo de conteúdo enviado nos pedidos HTTP.
   B. Especificar o tipo de conteúdo enviado nas respostas HTTP.
   C. Especificar o tipo de conteúdo do corpo da mensagem HTTP.
   D. Indicar ao browser como apresentar a página web.

NÚMERO _____ **NOME** _____

GRUPO 2 [4 valores]

Faça a correspondência entre o que está no lado esquerdo da tabela (identificado com números) e o que lhe está relacionado no lado direito da tabela (identificado por letras). Se por exemplo considerar que a opção 1 do lado esquerdo está relacionada com a opção c do lado direito, responda 1-c.

1. O `this` numa função em javaScript é:

1	undefined ou objeto global	**a**	A função foi chamada como método
2	Objeto usado na chamada à função	**b**	A função foi chamada como função global
3	Novo objeto	**c**	A função foi chamada através das funções `call()` ou `apply()` de `Function`.
4	Objeto arbitrário	**d**	A função foi chamada como função construtora

2. As séries dos Status Codes das respostas HTTP indicam:

1	2xx	**a**	O cliente enviou um pedido com erro
2	3xx	**b**	A resposta enviada corresponde ao pedido realizado
3	4xx	**c**	A resposta indica que o pedido não foi satisfeito por falha no servidor
4	5xx	**d**	O cliente tem que fazer algo adicional para obter o recurso que pretende

3. No Handlebars:

1	As entradas são	**a**	propriedades do objeto
2	A saída é	**b**	um template e um objeto
3	O template tem	**c**	texto
4	As expressões acedem a	**d**	texto e expressões Handlebars

4. Nos Arrays javaScript:

1	O método `map()` retorna	**a**	uma função predicado
2	O método `filter()` retorna	**b**	um array com o mesmo número de elementos do array sobre o qual foi chamado
3	O método `forEach()`	**c**	um array, no máximo com o mesmo número de elementos do array sobre o qual foi chamado, mas pode ter menos.
4	O método `filter()` recebe	**d**	retorna `undefined`

5. Num documento HTML:

1	O documento é constituído por	**a**	nome e podem ter um valor
2	Os elementos têm	**b**	marca de início e opcionalmente marca de fim
3	Na marca de início podem existir	**c**	elementos e texto
4	Os atributos têm	**d**	atributos

6. No contexto da fetch API:

1	A função fetch recebe	**a**	todos os dados da resposta.
2	O objeto response tem	**b**	uma Promise que é resolvida com um objeto Response
3	O método Json do objeto response retorna	**c**	todos os dados do pedido
4	A função fetch retorna	**d**	uma Promise que é resolvida com um objeto, caso seja possível

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores e Licenciatura em Engenharia Informática, Redes e Telecomunicações

Introdução à Programação na Web / Programação na Internet

Inverno de 2023/2024

** Teste Global de Época Normal (10 de Janeiro de 2024) - Duração 1h30

NÚMERO _____ **NOME** _____

GRUPO 1 [4 valores]

Nas seguintes questões, selecione a única opção verdadeira. Uma resposta correta conta a totalidade da cotação da pergunta; uma resposta incorreta desconta 25% da cotação da pergunta ao total do grupo; uma questão não respondida conta 0 valores.

1. Ao lidar com um pedido para um recurso que o cliente não tem permissões para aceder, apesar de apresentar credenciais válidas, o *status code* que deve ser retornado é:
   A. 500
   B. 404
   C. 403
   D. 400

2. Um pedido HTTP idempotente:
   A. Pode ser repetido várias vezes
   B. Pode alterar o estado da aplicação
   C. Pode ter os métodos: DELETE, GET ou PUT
   D. Todas as anteriores

3. Os cookies HTTP servem para:
   A. Autenticar o cliente perante o servidor
   B. Guardar informação sensível entre o cliente e o servidor
   C. O servidor guardar dados no cliente e obtê-los futuramente
   D. Determinar a localização do cliente

4. Qual o propósito do npm no processo de desenvolvimento em Node.js:
   A. Disponibilizar um sistema de controlo de versões para o código JavaScript
   B. Simplificar a instalação do ficheiro package.json
   C. Simplificar a instalação, partilha e gestão de módulos de *third-party*
   D. Gerar a documentação automática de APIs

5. Qual é o principal propósito do header `Content-Type` do protocolo HTTP?
   A. Indicar ao browser como apresentar a página web.
   B. Especificar o tipo de conteúdo enviado nas respostas HTTP.
   C. Especificar o tipo de conteúdo do corpo da mensagem HTTP.
   D. Especificar o tipo de conteúdo enviado nos pedidos HTTP.

6. Ao analisar a URL *http://www.example.com/products?id=123&category=electronics*, qual das seguintes afirmações é verdadeira?
   A. www.example.com identifica um recurso
   B. A query string contém dois parâmetros, id e category
   C. O path neste URL é id=123&category=electronics
   D. Nenhuma das opções anteriores está correta

7. O que é um *closure* na linguagem JavaScript?
   A. Uma forma de proteger as variáveis de um objeto
   B. Uma característica da linguagem que liberta o espaço de variáveis não usadas, de modo a otimizar a memória utilizada
   C. Uma forma de encapsular o ambiente léxico de uma função, permitindo o acesso às suas variáveis, mesmo depois da função retornar
   D. Um mecanismo da linguagem para criação de métodos privados num objeto

NÚMERO _____ **NOME** _____

GRUPO 2 [4 valores]

Faça a correspondência entre o que está no lado esquerdo da tabela (identificado com números) e o que lhe está relacionado no lado direito da tabela (identificado por letras). Se por exemplo considerar que a opção 1 do lado esquerdo está relacionada com a opção c do lado direito, responde 1-c.

1. As séries dos Status Codes das respostas HTTP indicam:

1	2xx	**a**	O cliente enviou um pedido com erro
2	3xx	**b**	A resposta enviada corresponde ao pedido realizado
3	4xx	**c**	A resposta indica que o pedido não foi satisfeito por falha no servidor
4	5xx	**d**	O cliente tem que fazer algo adicional para obter o recurso que pretende

2. No Handlebars:

1	As entradas são	**a**	propriedades do objeto
2	A saída é	**b**	um template e um objeto
3	O template tem	**c**	texto
4	As expressões acedem a	**d**	texto e expressões Handlebars

3. Nos Arrays javaScript:

1	O método `map()` retorna	**a**	retorna `undefined`
2	O método `filter()` retorna	**b**	um array com o mesmo número de elementos do array sobre o qual foi chamado
3	O método `forEach()`	**c**	um array, no máximo com o mesmo número de elementos do array sobre o qual foi chamado, mas pode ter menos.
4	O método `filter()` recebe	**d**	uma função predicado

4. Num documento HTML:

1	O documento é constituído por	**a**	nome e podem ter um valor
2	Os elementos têm	**b**	marca de início e opcionalmente marca de fim
3	Na marca de início podem existir	**c**	elementos e texto
4	Os atributos têm	**d**	atributos

5. No contexto da fetch API:

1	A função fetch recebe	**a**	todos os dados da resposta.
2	O objeto response tem	**b**	uma Promise que é resolvida com um objeto Response
3	O método Json do objeto response retorna	**c**	todos os dados do pedido
4	A função fetch retorna	**d**	uma Promise que é resolvida com um objeto, caso seja possível

6. O `this` numa função em javaScript é:

1	undefined ou objeto global	**a**	A função foi chamada como método
2	Objeto usado na chamada à função	**b**	A função foi chamada como função global
3	Novo objeto	**c**	A função foi chamada através das funções `call()` ou `apply()` de `Function`.
4	Objeto arbitrário	**d**	A função foi chamada como função construtora

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores e Licenciatura em Engenharia Informática, Redes e Telecomunicações

Introdução à Programação na Web / Programação na Internet

Inverno de 2023/2024

*** Teste Global de Época Normal (10 de Janeiro de 2024) - Duração 1h30

NÚMERO _____ **NOME** _____

GRUPO 1 [4 valores]

Nas seguintes questões, selecione a única opção verdadeira. Uma resposta correta conta a totalidade da cotação da pergunta; uma resposta incorreta desconta 25% da cotação da pergunta ao total do grupo; uma questão não respondida conta 0 valores.

1. O que é um *closure* na linguagem JavaScript?
   A. Um mecanismo da linguagem para criação de métodos privados num objeto
   B. Uma forma de proteger as variáveis de um objeto
   C. Uma forma de encapsular o ambiente léxico de uma função, permitindo o acesso às suas variáveis, mesmo depois da função retornar
   D. Uma característica da linguagem que liberta o espaço de variáveis não usadas, de modo a otimizar a memória utilizada

2. Ao analisar a URL *http://www.example.com/products?id=123&category=electronics*, qual das seguintes afirmações é verdadeira?
   A. O path neste URL é *id=123&category=electronics*
   B. *www.example.com* identifica um recurso
   C. A query string contém dois parâmetros, *id* e *category*
   D. Nenhuma das opções anteriores está correta

3. Ao lidar com um pedido para um recurso que o cliente não tem permissões para aceder, apesar de apresentar credenciais válidas, o *status code* que deve ser retornado é:
   A. 403
   B. 400
   C. 500
   D. 404

4. Um pedido HTTP idempotente:
   A. Pode ser repetido várias vezes
   B. pode ter os métodos: DELETE, GET ou PUT
   C. Pode alterar o estado da aplicação
   D. Todas as anteriores

5. Os cookies HTTP servem para:
   A. Autenticar o cliente perante o servidor
   B. Guardar informação sensível entre o cliente e o servidor
   C. Determinar a localização do cliente
   D. O servidor guardar dados no cliente e obtê-los futuramente

6. Qual o propósito do npm no processo de desenvolvimento em Node.js:
   A. Gerar a documentação automática de APIs
   B. Simplificar a instalação do ficheiro package.json
   C. Disponibilizar um sistema de controlo de versões para o código JavaScript
   D. Simplificar a instalação, partilha e gestão de módulos de *third-party*

7. Qual é o principal propósito do header `Content-Type` do protocolo HTTP?
   A. Especificar o tipo de conteúdo do corpo da mensagem HTTP.
   B. Especificar o tipo de conteúdo enviado nos pedidos HTTP.
   C. Indicar ao browser como apresentar a página web.
   D. Especificar o tipo de conteúdo enviado nas respostas HTTP.

NÚMERO _____ **NOME** _____

GRUPO 2 [4 valores]

Faça a correspondência entre o que está no lado esquerdo da tabela (identificado com números) e o que lhe está relacionado no lado direito da tabela (identificado por letras). Se por exemplo considerar que a opção 1 do lado esquerdo está relacionada com a opção c do lado direito, responde 1-c.

1. Num documento HTML:

1	O documento é constituído por	**a**	marca de início e opcionalmente marca de fim
2	Os elementos têm	**b**	nome e podem ter um valor
3	Na marca de início podem existir	**c**	atributos
4	Os atributos têm	**d**	elementos e texto

2. As séries dos Status Codes das respostas HTTP indicam:

1	2xx	**a**	A resposta enviada corresponde ao pedido realizado
2	3xx	**b**	O cliente enviou um pedido com erro
3	4xx	**c**	O cliente tem que fazer algo adicional para obter o recurso que pretende
4	5xx	**d**	A resposta indica que o pedido não foi satisfeito por falha no servidor

3. O `this` numa função em javaScript é:

1	undefined ou objeto global	**a**	A função foi chamada como função global
2	Objeto usado na chamada à função	**b**	A função foi chamada como método
3	Novo objeto	**c**	A função foi chamada como função construtora
4	Objeto arbitrário	**d**	A função foi chamada através das funções `call()` ou `apply()` de `Function`.

4. Nos Arrays javaScript:

1	O método `map()` retorna	**a**	um array com o mesmo número de elementos do array sobre o qual foi chamado
2	O método `filter()` retorna	**b**	uma função predicado
3	O método `forEach()`	**c**	retorna `undefined`
4	O método `filter()` recebe	**d**	um array, no máximo com o mesmo número de elementos do array sobre o qual foi chamado, mas pode ter menos.

5. No contexto da fetch API:

1	A função fetch recebe	**a**	uma Promise que é resolvida com um objeto Response
2	O objeto response tem	**b**	todos os dados da resposta.
3	O método Json do objeto response retorna	**c**	uma Promise que é resolvida com um objeto, caso seja possível
4	A função fetch retorna	**d**	todos os dados do pedido

6. No Handlebars:

1	As entradas são	**a**	um template e um objeto
2	A saída é	**b**	propriedades do objeto
3	O template tem	**c**	texto e expressões Handlebars
4	As expressões acedem a	**d**	texto

GRUPO 3 [7 valores]

1. [1.5] Considere o seguinte código em JavaScript (à esquerda) e o output da sua execução (à direita).

```
setTimeout(function () {
   console.log("Callback1 after 2 seconds...")
   setTimeout(function () {
          console.log("Callback2 after 4 seconds...")
          setTimeout(function () {
                 console.log("Callback 3 after 6 seconds...")
          }, 2000)
   }, 2000)
}, 2000)
```

```
Callback1 after 2 seconds...
Callback2 after 4 seconds...
Callback3 after 6 seconds...
```

Crie uma versão alternativa utilizando Promises, que apresente o mesmo resultado e nos mesmos tempos que a implementação anterior. Nesta resolução não use async/await.
SUGESTÃO: Crie uma função Auxiliar que recebe a mensagem a mostrar e o tempo a esperar

2. [3] Pretende-se criar um módulo que encapsula a utilização da função fetch, apresentando na consola o URI a que está a ser feito cada pedido e, quando recebida a resposta, o URI a que esta corresponde e se o conteúdo da resposta for JSON, mostra um objeto resultado da conversão. A listagem seguinte mostra um exemplo da sua utilização:

```
import fetch from './logged-fetch.mjs'
await fetch("https://api.chucknorris.io/jokes/random")
await fetch("https://www.isel.pt", { method: "GET", headers: { 'User-Agent': "SLB"} })
```

```
Making request to uri https://api.chucknorris.io/jokes/random
Response received for uri https://api.chucknorris.io/jokes/random
Json response received converted to object
{
  . . . // Omitido por brevidade
  value: 'Chuck Norris can't defy gravity. Gravity would never oppose him.'
}
------------------
Making request to uri https://www.isel.pt
Response received for uri https://www.isel.pt
Non Json response received
```

Implemente o módulo logged-fetch.mjs, de modo a apresentar este comportamento. Note que devem ser suportados todos os tipos de pedidos e não apenas GET, como os apresentados no exemplo.

3. [2.5] Considere o código em Node.js apresentado, onde as funções `mw`, `mw1`, `mw2` e `mw3` são *middlewares* express, onde nenhum deles pára o processamento do pedido. A tabela seguinte apresenta em cada linha um pedido HTTP (Método e URI path) e nas colunas seguintes com o valor S ou vazia, consoante as funções correspondentes sejam executadas ou não em cada um dos pedidos.

	Method	URI path	mw1	mw2	mw3	mw
1	GET	/tasks/urgent	S	S		S
2	GET	/tasks			S	S
3	GET	/?tasks				
4	GET	/tasks/1/edit				S

Complete o código com o registo das rotas de modo a que a execução da tabela se verifique.

```
const app = express()
const p1 = "/tasks/top"
const p2 = "/tasks"
const p3 = "/tasks/:id"
_____________ // Register the routes
```

GRUPO 4 [5 valores]

1. [3,5] Pretende-se implementar um middleware express que regista informação sobre os pedidos HTTP realizados a uma aplicação Express em Node.js, nomeadamente: o tipo de pedido, o URI e o valor de alguns Headers selecionados. Nas listagens seguintes apresenta-se a utilização e uma parte da implementação desta solução. Com esta implementação, o pedido GET http://localhost:1904/handler1, mostra a seguinte mensagem na consola do servidor:

```
GET - /handler1

Content-Type:
User-Agent: vscode-restclient
```

http-logger.mjs

```
1  function consoleLogger(message) { console.log(message) }
2
3  export __________ function(headers = [], logFunction = consoleLogger){  // 1
4     return function (req, rsp, next){
5         let str = `${__________} - ${__________}\n\n`  // 2
6         if(headers.length) {
7             str += headers.map(h => `${h}: ${req.____(h) || ""}`).join('\n') // 3
8         }
9         logFunction(__________)  // 4
10        __________  // 5
11    }
12 }
```

server.mjs

```
1  import express from "express";
2  import loggerConstructor from "./http-logger.mjs"
3  const app = express()
4
5  // Log headers Content-Type e User-Agent
6  const logger = __________(["Content-Type", "User-Agent"])  // 1
7
8  app.use(__________)    // 2
9  app.get("/handler1",handler1)
10 app.get("/handler2",handler2)
11
12 const PORT = 1904
13 app._______(PORT, ()=> console.log(`Server listening at http://localhost:${PORT} `)) // 3
14
15 function handler1(req,rsp){rsp.json({message : "handler1"})}
16 function handler2(req,rsp){rsp.json({message : "handler2"})}
```

   a. [2.5] Complete a implementação do módulo http-logger.mjs.
   b. [1.5] Complete a implementação do módulo server.mjs.
   c. [1] O módulo `http-logger` recebe como argumento uma função que processa a mensagem de log e que, por omissão, a escreve na consola. No entanto, caso se pretenda que `logFunction` possa usar node:fs/promises para escrever a mensagem num ficheiro, a implementação dada não é adequada. Justifique e indique o que teria de ser alterado no módulo `http-logger` para que tal seja possível.

Os docentes,
Luís Falcão e Valderi Leithardt