Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação na Internet

Teste de Época Especial, Inverno de 2018/2019

GRUPO 1

1. [2,5] Dadas um método HTTP e ação/característica do pedido ou resposta que se segue, crie uma frase usando apenas uma das seguintes palavras: SEMPRE, PODE, NUNCA.
   Exºs: GET - criação de um recurso: O método GET NUNCA deve ser usado para criar um recurso.
   a. [0,5] POST - Atualização de um recurso
   b. [0,5] PUT - Criação de um recurso
   c. [0,5] GET - *body* da resposta sem conteúdo
   d. [0,5] PUT - *body* do pedido sem conteúdo
   e. [0,5] DELETE - *body* da resposta com conteúdo

2. [3,5] A função `nodify(fn)` recebe como argumento fn uma função que recebe N argumentos e retorna uma `Promise`. Retorna uma nova função que recebe os mesmos N argumentos mais um, em último lugar, que é uma função de *callback*, com o idioma convencionado em NodeJS. A listagem seguinte apresenta um exemplo de utilização desta função.

```
function f(success, a,b,c) {
 return new Promise((resolve, reject) => {
   setTimeout(() => {
     success ? resolve(a+b+c):reject("err")
     console.log(`Success: ${success} `)
   }, 2000)
 })
}
```

```
function showCbFunctionResult(f, ...args) {
 f.apply(this, args.concat(cb))
 function cb(err, res) {
   err ? console.log(`Error is ${err}`)
       : console.log(`Result is ${res}`)
 }
}

let fp = nodify(f)
showCbFunctionResult(fp, true, 1,2,3)
showCbFunctionResult(fp, false, 1,2,3)
```

   a. [2,5] Implemente a função `nodify` de modo ter o comportamento indicado.
   NOTA: A função `nodify` deve funcionar com qualquer função que respeite os requisitos enunciados e não apenas com a função `f` do exemplo.

   b. [1] Indique o output na consola da execução do código na listagem anterior.

3. [2] Para a seguinte definição do módulo `wacky` assinale com V as utilizações que executam sem erros e com F as utilizações que dão erro na execução. Para as respostas V, indique o resultado na consola dessa execução.

```
const wacky = nr => () => console.log(nr/3)
wacky.dup =  nr => console.log(nr*2)
module.exports = wacky
```

   a. require('./wacky')(9).dup(3)
   b. require('./wacky').dup(3)
   c. require('./wacky')(9)
   d. require('./wacky')(9)(3)

4. [2] Considere a seguintes regras de estilo. Classifique as afirmações abaixo de verdadeiras ou falsas, justificando.

```
.table { border-collapse: collapse; }
th, td { border-bottom: 1px solid #ddd; }
tr.even { background-color: #f2f2f2;}
tr td.first, tr th.first { width: 60%; }
```

   a. A 1ª regra aplica-se a todas as tabelas.
   b. A 2ª regra aplica-se a elementos th e td.
   c. A 3ª regra aplica-se a elementos tr em posição par.
   d. A 4ª regra aplica-se a todos elementos `tr`, `td` e `th`.

GRUPO 2

1. [10] No âmbito da aplicação desenvolvida no trabalho prático, pretende-se implementar a funcionalidade de *auto-complete* para pesquisa do nome de equipas. Esta funcionalidade apresenta uma lista de sugestões de nomes de equipas ao utilizador, à medida que ele escreve texto num elemento de formulário.

   a. [2] Especifique o *endpoint* a disponibilizar para suportar esta funcionalidade. Note que não se exige que o implemente; o que se pretende é que defina tudo o que for necessário para um utilizador da API poder usar esse endpoint. Considere os requisitos que estão explícitos e implícitos na alínea que se segue.

   b. [3] Implemente a função **`getTeamNameSuggestions(teamNameStart, teamCount)`** que, no *browser*, realiza um pedido ao *endpoint* da alínea anterior para obter a lista das equipas cujo nome começa por **`teamNameStart`**. A lista tem dimensão máxima especificada em **`teamCount`**. A função retorna uma Promise que em caso de sucesso produz um *array* de objetos com o nome e id da equipa. Caso ocorra algum erro na tentativa de obtenção da lista, a promessa conclui-se com a indicação do código de erro ocorrido. A implementação certifica-se que, mesmo que o *endpoint* suporte mais codificações, é obtida uma resposta com conteúdo JSON.

   c. [3] Considere o seguinte excerto da página HTML onde este funcionalidade está disponível. Acrescente o que considerar necessário para que, à medida que o utilizador escreve texto, o conteúdo do `div` **`suggestions`** seja actualizado com as sugestões obtidas usando a função da alínea anterior. Garanta que este `div` só está visível se **`teamName`** tiver pelo menos dois caracteres e se as sugestões foram obtidas com sucesso. Na implementação considere que já estão definidas as regras CSS necessárias, **`.hiddenAutoComplete { … }`** e **`.visibleAutoComplete { ... }`**.

```
<input type="text" name="teamName" id="teamName" />
<div class="hiddenAutoComplete" id="suggestions"></div>
```

   d. [2] Dadas as características da FootballData API, que problemas/limitações antevê na implementação desta funcionalidade? Descreva possíveis soluções para atenuar/suprimir essas limitações.

Duração: 2 horas
ISEL, 20 de Fevereiro de 2019