Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

1 º **Ano**/ 2 º **Semestre**

Unidade Curricular: Desenvolvimento Web - Back-End **Docente:** David Jardim

---

FICHA DE TRABALHO 4

Exercícios:

1. Crie uma pasta para a ficha 4
2. Crie um ficheiro *app.js*
    a. Crie um objeto da forma literal com as seguintes propriedades: name, age, gender e converta o objeto para JSON utilizando a função JSON.`stringify`
    b. `Crie uma string manualmente no formato JSON que represente o objeto criado anteriormente e converta a string para um objeto utilizando a função JSON.parse`
3. Crie um ficheiro *person.js*
    a. Crie um objeto/classe **Person** usando a sintaxe da função construtor
    b. Adicione duas propriedades ao objeto
        firstName
        i. **lastName**
    c. Utilizando herança prototipada adicione um novo método *greet* que efetua uma saudação com o primeiro e o último nome: "Hello John Doe"
    d. Crie duas instâncias do objeto **Person** com nomes diferentes.
    e. Utilizando herança prototipada adicione uma nova propriedade **age**
    f. É possível aceder/alterar essa propriedade?
    g. Altere o método *greet* para utilizar a idade
    h. Imprima para a consola a propriedade **__proto__**, qual o seu conteúdo?
    i. Compare a propriedade **__proto__** das duas instâncias criadas previamente. Qual o resultado?
4. Crie um ficheiro function_arrays.js
    a. Crie um array vazio
    b. Utilizando a função push adicione 3 **funções anónimas** ao array que escrevam uma mensagem na consola: "Hello World 1, 2, 3"
    c. Como podemos invocar todas as funções que foram adicionadas ao array utilizando a expressão ***for***?
    d. Replique a alínea anterior utilizando a expressão ***forEach***
5. Crie um ficheiro *emitter.js* para criarmos a nossa versão de um Event Emitter
    a. Crie um objeto/classe **Emitter** usando a sintaxe da ***class declaration***
    b. Adicione uma propriedade **events** que consiste num objeto vazio
    c. Utilizando herança prototipada adicione um novo método *on* que recebe dois argumentos: *type* e *listener*.

Cofinanciado por:

MADEIRA 14-20 REGIÃO AUTÓNOMA DA MADEIRA PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

d. O objetivo do método *on* é registar *listeners* que são funções que devem ser chamadas quando um determinado evento ocorrer. Os listeners devem ser adicionados à uma propriedade com o nome *events* que será um array de funções. Caso o tipo do evento não exista terá que ser inicializado com um array vazio.
e. Utilizando herança prototipada adicione um novo método *emit* que recebe um argumento: *type.* O objetivo deste método é invocar todos os *listeners* previamente registados. Utilize a expressão ***forEach*** como no exercício anterior.
f. Exporte a classe Emitter como um módulo utilizando a funcionalidade ***module.Exports***
g. No ficheiro *app.js* importe o módulo utilizando a funcionalidade ***require***
h. Crie uma nova instância da classe Emitter utilizando a palavra chave ***new***
i. Registe duas funções anónimas à instância utilizando o método *on* para o mesmo evento. Não se esqueça que o método recebe dois argumentos, *type* – nome do evento e *listener* – função que será executada quando o evento ocorrer
j. Invoque o evento na instância utilizando o método *emit.* O argumento da função é o nome do evento.
k. Registe um novo evento com um nome diferente e invoque-o

6. Crie um ficheiro config.js para armazenar os nomes dos eventos
   a. Crie um novo objeto events que irá armazenar os nomes dos eventos como atributos
   b. Exporte o objeto como um módulo utilizando a funcionalidade ***module.Exports***
   c. No ficheiro *app.js* importe o módulo utilizando a funcionalidade ***require***
   d. Substitua todas as ocorrências das "magic strings" pelas propriedades constantes do ficheiro *config.js*