Introdução às Tecnologias *Web*

Guião da Aula Teórico-Prática sobre Formulários

Antes da aula

No módulo de HTML Forms do W3Schools, ler as secções sobre HTML Forms, HTML Form Elements, HTML Input Types, e HTML Input Attributes. No módulo de HTML APIs, ler a secção sobre HTML Web Storage.

Durante a aula

Comece por descarregar o ficheiro de material de apoio à aula teórico-prática, o qual contém `index.html`, `apresentacao.css` (na pasta `estilos`), `comportamento.js` (em `scripts`), e `resultado.png` (em `resultados`). A imagem PNG mostra o aspeto que a página HTML deve ter após a resolução dos exercícios desta aula.

O objetivo desta aula é a aquisição de competências sobre formulários *web*, incluindo a construção de um formulário em HTML, a gestão de dados usando código JavaScript, e o armazenamento de dados no *local storage* do *browser*. A representação visual de um formulário não é contemplada nos exercícios seguintes, mas são fornecidos exemplos de regras CSS.

Exercício 1: Analise a estrutura e conteúdo de `index.html` num editor de código, em particular a secção com o formulário de encomenda de piza, e uma segunda secção com o histórico de encomendas. Se abrir a página num *browser*, pode notar que faltam vários elementos face à imagem em `resultado.png`, que irão sendo preenchidos ao longo dos próximos exercícios.

Exercício 2: Analise atentamente o código em `comportamento.js`, o qual está organizado nas seguintes partes: a) constantes, para facilitar alterações futuras; b) variáveis globais, em especial o histórico de encomendas; c) construtores de objetos, para representar pizas, clientes, e encomendas; d) inicialização da aplicação, quando o *browser* tiver carregado o documento HTML; e) reação a eventos do utilizador, como sejam o premir de botões; f) obtenção de dados do formulário, com funções especializadas nos dados de pizas e clientes; e g) gestão do histórico de encomendas, incluindo a gravação no *local storage* do *browser*, o carregamento para memória, e a visualização em forma de tabela.

Exercício 3: Preencha o formulário no *browser* e carregue no botão de fazer encomenda de piza. Verifique que a tabela de histórico de encomendas foi atualizada. Na consola do *browser*, inspecione o valor da variável global `encomendas`, devendo encontrar objetos que guardam os dados do cliente e da piza. Por fim, confirme que o histórico de encomendas está a ser gravado no *local storage* do *browser* (no Chrome está em Application › Storage).

Exercício 4: Acrescente mais opções de ingredientes ao formulário HTML, nomeadamente fiambre, queijo, e tomate. É necessário alterar algum código JavaScript para que estes novos ingredientes possam fazer parte de encomendas de piza?

Exercício 5: Em `index.html`, acrescente um botão para apagar o histórico de encomendas, entre o título da secção e a tabela de histórico, conforme mostrado em `resultado.png`. Em `comportamento.js`, guarde o nome desse botão numa constante chamada `BOTAO_APAGAR_ENCOMENDAS` e atualize o código da função `defineEventHandlersParaElementosHTML()` para que, ao ser premido o botão, seja invocada `trataApagarHistoricoEncomendas()`. Esta função já está documentada e parcialmente codificada, faltando o mais importante, que é apagar o histórico em memória e gravar essa alteração no *local storage* do *browser*, podendo para este último efeito ser chamada a função `gravaHistoricoEncomendas()`.

O botão de apagar o histórico de encomendas vai ser útil para os dois exercícios seguintes, os quais vão requerer alterações nas estruturas de dados que representam pizas e clientes. Assim, para evitar que o *local storage* do *browser* guarde representações diferentes, e potencialmente incompatíveis, dos dados de encomendas, recomenda-se que o histórico seja apagado antes de serem feitos testes a novo código.

Exercício 6: Acrescente os campos morada e data de nascimento aos dados do cliente. Use o tipo adequado de elemento `input` para facilitar a introdução de dados pelo utilizador e tenha em conta que ambos os campos são de preenchimento obrigatório e que a morada tem no máximo 80 letras e dígitos. Coloque os nomes desses campos nas constantes `MORADA_CLIENTE` e `NASCIMENTO_CLIENTE` e atualize o construtor de objetos `Cliente` e a função `obtemDadosCliente()` para passarem a considerar os novos dados. Por fim, em `mostraHistoricoEncomendas()` adicione as colunas `Morada` e `Nascimento` à tabela de histórico de encomendas.

A partir deste momento, se fizer uma nova encomenda, o *local storage* do *browser* (e o objeto em memória que representa o histórico de encomendas) vai passar a guardar o nome, morada, e data de nascimento do cliente, em vez de apenas o nome. Note que não é necessário alterar as funções `gravaHistoricoEncomendas()` e `carregaHistoricoEncomendas()`, pois estas tiram partido de `JSON.stringify()` e `JSON.parse()`, que servem para converter qualquer objeto em memória para texto em formato JSON *(JavaScript Object Notation)* e vice-versa.

Exercício 7: Acrescente o campo quantidade de pizas ao formulário, de preenchimento obrigatório pelo utilizador, tendo o cuidado de serem apenas admitidos números entre 1 e 9, com valor por omissão 1. Guarde o nome desse campo na constante `QUANTIDADE_PIZAS` e codifique a função `obtemQuantidadePizas()`. Depois, atualize o construtor de objetos `Encomenda` para passar a receber a quantidade de pizas através de um terceiro parâmetro, para além dos objetos que representam a piza e o cliente. Procure no código onde é que esse construtor está a ser utilizado e atualize a respetiva instrução de criação de uma nova encomenda. Falta apenas mostrar a quantidade de pizas na tabela de histórico de encomendas, numa coluna chamada `Quantidade`, e experimentar fazer novas encomendas.

Se estiver a ter problemas com a visualização do histórico de encomendas, em particular durante o arranque da aplicação, a causa pode estar na existência de encomendas com diferentes representações no *local storage* do *browser* (por exemplo, com e sem a quantidade de pizas), devendo o histórico ser apagado.