Interface Pessoa-Máquina

Licenciatura em Engenharia Informática

Ficha Prática #06

Rafael Braga
d13414@di.uminho.pt

Daniel Murta
d6203@di.uminho.pt

José Creissac Campos
jose.campos@di.uminho.pt

(v. 2024)

Conteúdo

1 Objetivos

1. Por em prática os conceitos básicos da *framework* Vue.js.

2 Vue.js

Vue.js é uma *framework* progressiva para a construção de interfaces pessoa-máquina. Foi desenhada para poder ser adotada de forma incremental, desde uma utilização simples para enriquecer partes de um projeto existente, até ao desenvolvimento de aplicações completas.

2.1 Principais características

Algumas das principais características de Vue.js incluem:

Reatividade: Vue.js utiliza um sistema de reatividade que permite atualizar automaticamente o DOM quando o estado da aplicação muda.

Diretivas: Oferece diretivas personalizadas como `v-if`, `v-for` e `v-model` para estender o HTML com funcionalidades interativas.

Componentes: Permite construir aplicações complexas através de componentes reutilizáveis e encapsulados.

Transições: Suporta transições e animações para elementos do DOM, facilitando a adição de efeitos visuais interativos.

Nesta ficha iremos praticar a utilização do sistema de reatividade e das diretivas.

2.2 DOM virtual

Para otimizar a atualização do DOM e melhorar o desempenho, Vue.js utiliza um DOM virtual. Em vez de manipular diretamente o DOM real do *browser*, o Vue.js trabalha com uma cópia em memória (o DOM virtual).

Com o auxílio do DOM virtual a *framework* determina os elementos que precisam de ser atualizados no DOM real e apenas essas alterações são efetuadas, evitando redesenhar a página inteira. Esta abordagem é significativamente mais eficiente que a manipulação direta do DOM, pois reduz a quantidade de operações de atualização necessárias, possibilitando um melhor desempenho e uma melhor experiência de utilização para o utilizador final.

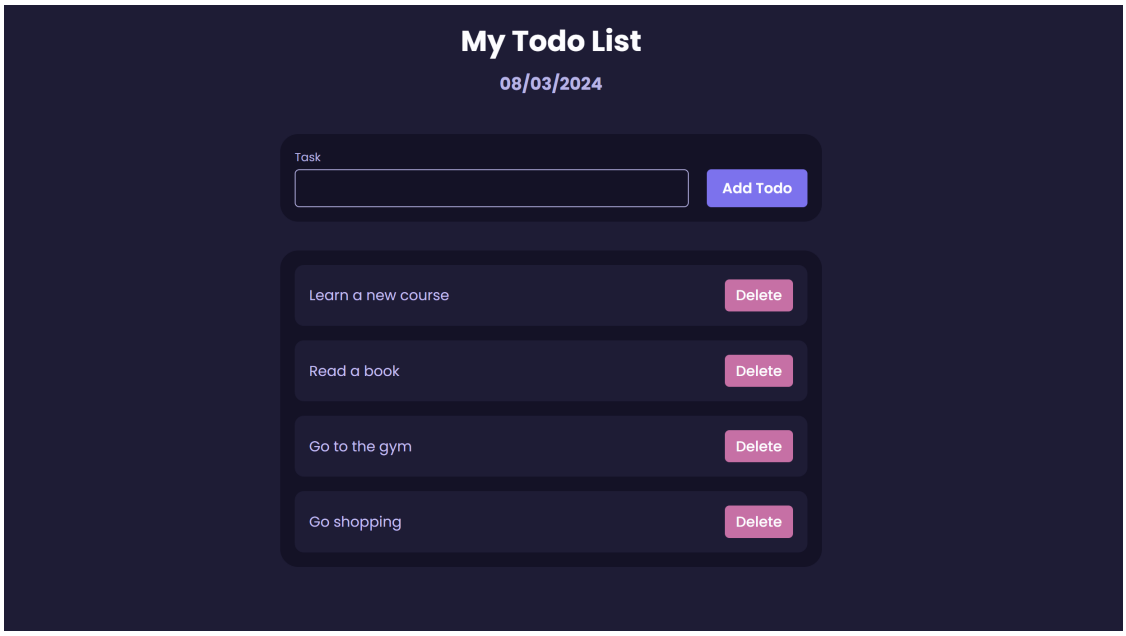

Figura 1: Website de lista de todos

A utilização do DOM virtual, no entanto, apresenta também alguns problemas, uma vez que é necessário que a *framework* identifique corretamente os elementos do DOM que deve comparar. Isso verifica-se, por exemplo, na manipulação de listas. Uma forma de evitar problemas a este nível é identificar claramente cada elemento da lista por meio do atributo `key`, específico de Vue.js.

3 Exercícios

Resolva os seguintes exercícios.

3.1 Lista de *Todos*

Relembre o Exercício 3.2 da Ficha Prática #05. Nele pedia-se o desenvolvimento do *website* apresentado na Figura 1, uma lista de tarefas (*todos*) a serem realizadas num dia. Pretende-se, agora, que desenvolva uma nova versão desta aplicação, desta vez com recurso a Vue.js. Para tal, considere a implementação base fornecida nesta ficha que contém os seguintes elementos:

- O ficheiro "index.html" que contém toda a estrutura do *website*. Neste ficheiro deverão ser adicionadas um conjunto de diretivas Vue.js que deverão permitir adicionar comportamento dinâmico a este ficheiro HTML.

- O ficheiro "styles.css" que contém o conjunto de regras CSS que permite chegar ao aspeto do *website* apresentado.
- O ficheiro "todo.js" onde se deverá acrescentar todo o comportamento do *website*. O comportamento deverá ser definido à custa da *framework* Vue.js.

Relembre que o *website* deverá permitir acrescentar tarefas (únicas) à lista, bem como ir removendo tarefas à medida que o utilizador as termine. Para tal implemente as seguintes etapas:

1. Crie uma instância do objeto `Vue` onde se poderá definir o comportamento do *website*. Para tal use o método `Vue.createApp`. Faça com que esta instância controle todos os elementos filhos do elemento `<div>` do ficheiro HTML com o id "app".

2. Adicione a data atual ao elemento `<h3>` com o id "list-date". Para tal deverá adicionar uma propriedade `today` à instância Vue, cujo valor inicial deverá corresponder à representação textual da data atual. Utilize para isso o método `toLocaleDateString()` da classe `Date`. Utilize uma diretiva Vue para fazer o *output* desta propriedade no elemento `<h3>`.

3. Crie uma propriedade com o nome `enteredTodo` cujo valor inicial deverá ser uma *string* vazia. Esta propriedade deverá estar vinculada (*bind*) ao atributo `value` do elemento `<input>` com o id "task-input". Adicione ainda, recorrendo à framework Vue.js, um *event handler* ao evento `input` do elemento anterior. Este *event handler* deverá atualizar o valor da propriedade `enteredTodo`. Para tal, crie um método `setEnteredTodo` na instância Vue e ligue o evento a este método.

4. Adicione à instância Vue uma propriedade `todos` cujo valor inicial é um array vazio. Crie um método na instância Vue com o nome `submitTodo`. O método deverá adicionar a tarefa escrita pelo utilizador (cujo valor está na propriedade *enteredTodo*, como definido na alínea anterior) ao array de *todos*. Defina o método como *event handler* do evento `submit` da `<form>` com o id "todo-form". Tenha em atenção que este evento de submição faz, por omissão, o *refresh* de toda a página. Como fazer para prevenir este comportamento?

   Melhore agora o método para que a tarefa só seja adicionada se contiver texto (não deve consistir apenas em espaços[1]) e se não existir já no array de *todos*[2]. Caso já exista no array de *todos*, deverá ser mostrada uma mensagem de erro

[1] Pode consultar os métodos disponíveis para manipular strings em String – Javascript | MDN. [2] Pode consultar os métodos disponíveis para manipular arrays em Array – Javascript | MDN.

ao utilizador (utilize a função `alert`). Verifique o funcionamento deste método imprimindo o array na consola (através da função `console.log`). Pode ver a consola através das *developer tools* do seu *browser*. No final, o método deverá limpar o texto introduzido pelo utilizador.

5. Renderize toda a lista de *todos* no elemento `<ul>` cujo id é "todo-list" com recurso à diretiva `v-for`. Para cada elemento no array de *todos* deverá criar um elemento `<li>` com uma classe "todo-list-item". Esse elemento deverá conter um elemento `<p>` onde será feito o output da tarefa e um `<button>` que deverá conter a label "Delete".

6. Crie o método *deleteTodo* na instância Vue que deverá receber como argumento o índice da tarefa a ser removida do array de "todos", e ligue este método ao evento `click` do elemento `<button>` criado na etapa anterior. Para invocar este método com o índice da tarefa correto, modifique a diretiva `v-for` feita na alínea anterior para o seguinte formato:

   ```
   (element, index) in array
   ```

   em que `array` é o nome do array a ser renderizado, `element` vai iterar sobre cada um dos valores do array, e *index* é o índice do elemento no array em cada iteração.

7. Crie um elemento `<h3>`, com as classes "subtitle" e "no-todos" e com o texto "No todos for today!", diretamente abaixo do elemento `<ul>` com o id "todo-list". O objetivo é mostrar este elemento ao utilizador quando o array de "todos" for vazio. Caso contrário, deverá ser mostrada a lista de "todos". Utilize para isto duas alternativas: uma através da diretiva `v-show` e outra através das diretivas `v-if` e `v-else`. Consulte as *developer tools* do seu *browser* para entender as diferenças entre estas duas alternativas. Quais lhe parecem ser os casos de uso apropriados para cada uma delas?

8. Considere que uma análise heurística da interface concluiu que se deveria obedecer à heurística de Nielsen que aconselha a prevenir os erro. Pretende-se que o botão "Add Todo" não esteja disponível se não existir texto na tarefa a adicionar. Acrescente à instância Vue uma propriedade calculada (`computed`) booleana que será verdadeira se a propriedade `enteredTodo` não contiver texto. Vincule agora o atributo `disabled` do botão ao valor da propriedade que criou. Finalmente, adicione ao CSS regras para que quando o botão estiver *disabled* a sua cor de fundo seja escura, a sua margem seja desenhada a tracejado, e o cursor do rato fique com o sinal de proibido (`cursor: not-allowed;`).

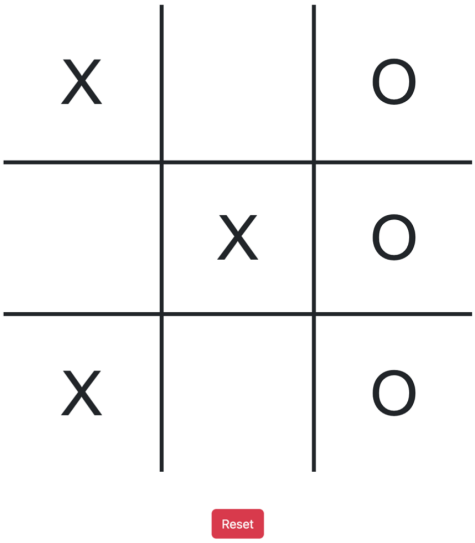

Figura 2: Jogo do Galo

3.2 Jogo do Galo

Relembre o Exercício 3.2 da Ficha Prática #05. Nele pedia-se o desenvolvimento de uma página web para jogar ao jogo do Galo (ver Figura 2). Pretende-se agora que o jogo seja desenvolvido recorrendo à *framework* Vue.js.

Implemente o jogo, utilizando os conhecimentos adquiridos sobre Vue.js. Para utilizar a *framework*, adicione o seguinte *script* no elemento `<head>` do ficheiro HTML:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```