Svelte.js

Grupo 3

Martín Ocqueteau

Nicolás Fraga

Agustín Ríos

1

Consulta a APIs de Chistes 💬

3 Tipos

- Dad Jokes 👨‍🦰
- Geek Jokes🤓
- F-Off🤬

Demostración

I don't trust stairs. They're always up to something.

Who the fuck are you anyway, Mr. Lebowski, why are you stirring up so much trouble, and, who pays you?

- Other Lebowski

No statement can catch the ChuckNorrisException.

La Base de todo: Content

```
<script lang="ts">
  import type { Content } from "../interfaces/joke";
  export let joke: Content;
</script>
```

```ts
/* ./src/interfaces/joke.ts */
export interface Content {
    id?: number;
    type?: string;
    value: string;
}
```

```ts
/* ./src/helpers/joke.ts */
export const newContent =
(value: string, type?: string, id?: number): Content => {
    return {
        id,
        type,
        value
    };
}
```

2

Componentes⚙️

JokeCard

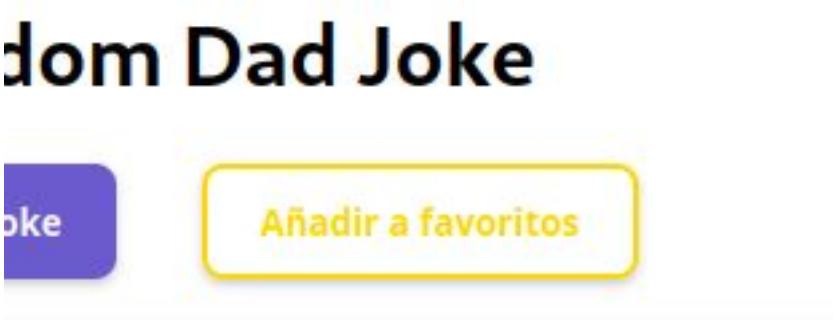
Encargado de mostrar el texto pedido a las APIs en una variable.

Variable actualizada con los métodos de botones. Se actualiza el Prop en base a la promesa de consultar a la API.

```
/* ./src/components/JokeCard.svelte */

<script lang="ts">
  import type { Content } from "../interfaces/joke";
  export let joke: Content;
</script>

<div class="joke-card">
  <p>
    <strong>{joke?.value || "Request a joke!"}</strong>
  </p>
</div>

<style>...</style>

<!-- <JokeCard {joke} /> -->
```

AddFavouriteButton

Encargado de guardar los “jokes” de tipo Content en la lista de favoritos.



```svelte
/* ./src/components/AddFavouriteButton.svelte */

<script lang="ts">
  export let handleClick: Function;
</script>

<button on:click={() => handleClick()}>
     Añadir a favoritos
</button>

<style>...</style>

<!-- <AddFavouriteButton handleClick={saveFunc} /> -->
```

NavBar

Componente que facilita la navegación en la página.

Utiliza componente Link de librería svelte-routing

```
/* ./src/components/NavBar.svelte */
<script lang="ts">
 import { Link } from 'svelte-routing';
</script>
<nav>
  <Link to="/">Home</Link>
  <Link to="/geek-jks">Geek</Link>
  {...}
</nav>
```

Vue Components
Vs.
Svelte Components

Caso del componente “JokeCard”

```
/* ./src/components/JokeCard.svelte */

<script lang="ts">
  import type { Content } from "../interfaces/joke";
  export let joke: Content;
</script>

<div class="joke-card">
  <p>
    <strong>{joke?.value || "Request a joke!"}</strong>
  </p>
</div>

<style>...</style>

<!-- <JokeCard {joke} /> -->
```

```
/* ./src/components/JokeCard.vue */
<script setup lang="ts">
  defineProps({
    joke: String,
  });
</script>

<template>
  <div class="joke-card">
    <p><strong>{{ joke || "Request a joke!" }}</strong></p>
  </div>
</template>

<style scoped>...</style>

<!-- <JokeCard :joke="joke" /> -->
```

```
/* ./src/views/FavouritesView.svelte */
{#each $jokeArray as joke (joke.id)}
   <div class="joke-section">
     <button on:click={() => deleteJoke(joke.id)}>
          Delete
     </button>
     <li>{joke.type}: <JokeCard {joke} /></li>
   </div>
 {/each}

/* ./src/views/FavouritesView.vue */
<div class="joke-vault">
  <li v-for="[idx, joke] in indexedJokes" :key="idx">
     <button @click="jokeStore.delete(joke)">Delete</button>
     <JokeCard :joke="joke" />
   </li>
</div>
```

3

Routing

svelte-routing

Librería con la que manejamos el despliegue de las vistas en nuestra SPA.

Provee los componentes `Router`, `Route` y `Link`

```svelte
/* ./src/App.svelte */
<script lang="ts">
  import { Router, Route } from "svelte-routing";
  import * as view from "./views";
  import NavBar from "./components/NavBar.svelte";
</script>

<Router>
  <NavBar />
  <main>
    <Route path="/" component={view.TheWelcome} />
    <Route path="/" component={view.FOAAS} />  /* No se ve */
    <Route path="/dad-jks" component={view.DadJoke} />
    <Route path="/geek-jks" component={view.GeekJoke} />
    {...}
  </main>
</Router>
```

4

Frontend ✨

Cómo es una vista de Svelte?
Parecida a Composition API

```svelte
<script lang="ts">
  const hello: string = "hello";
</script>

<h1>{ hello }</h1>

<style>
  h1 {
    color: red;
  }
</style>
```

hello

```svelte
<script setup lang="ts">
  import {ref} from 'vue';
  const hello = ref('hello');
</script>
<template>
  <h1>{ hello }</h1>
</template>
<style scoped>
  h1 {
    color: red;
  }
</style>
```

Pidiendo Joke a la API

```
/* ./src/services */
export async function getRandomDadJoke() {
  try {
    const { data } = await
axios.get("https://icanhazdadjoke.com/", {
      headers: { accept: "application/json" },
    });
    return data;
  } catch (error) {
    console.error(error);
  }
}
```

Pidiendo Joke a la API

```
/* ./src/views/DadJokesView.svelte */
const loadJoke = (): null => {
   error = "";
   aviso = "";
   loading = true;
   joke = { value: "Loading..." };
   getRandomDadJoke()
     .then((data) => {joke = newContent(data?.joke, "Dad Joke")})
     .catch((err) => { error = err.message })
     .finally(() => { loading = false });
   return null;
 };
```

Guardando Joke en favoritos

```
/* ./src/views/DadJokeView.svelte */
const saveJoke = (): null => {
   if (joke?.value && !loading) {
     postJoke(joke);
     aviso = "Joke added to favourites";
   } else {
     error = "*No joke to add to favourites";
   }
   return null;
 };
```

Mostrando el “Joke”

```
/* ./src/views/DadJokeView.svelte */
<h1>Random Dad Joke</h1>
<div class="actions">
  <button on:click={loadJoke}>Load Joke</button>
  <AddFavouriteButton handleClick={saveJoke} />
</div>
<JokeCard {joke} />
```

5

Manejo de estado

Svelte Store

Es simplemente un objeto con un método de suscripción que permite que las partes interesadas sean notificadas cada vez que cambia el valor de la tienda.

Para ello Svelte nos ofrece los siguientes métodos:

- **Writable:** Modificar y leer la tienda.
- **Readable:** Leer el store.

```
/* ./src/stores/index.ts */

import { writable } from "svelte/store";
import type { Content } from "../interfaces/joke";

const jokeArray = writable<Array<Content>>([]);
const maxJokeId = writable(0);
```

Svelte Store: Suscriptions

- **subscribe:** Para suscribirse a un **store.** Cada vez que cambia el valor contenido por el store se ejecuta una subrutina definida por nosotros.
- **unsubscribe:** Para des-suscribirse de un **store.**

```
const unsubscribe = myStore.subscribe((value) => {
  /* Do something */
});
{...}
unsubscribe()
```

```
/* ./src/views/FavouritesView.svelte */

<script lang="ts">
  {...}
  let favouriteJokes: Array<Content>;
  let jokesLength: number;

  jokeArray.subscribe((jokes) => (jokesLength = jokes.length));

  const sIfPlural = jokesLength === 1 ? "joke" : "jokes";
</script>
{...}
{#each $jokeArray as joke (joke.id)}
   <div class="joke-section">
     <button on:click={() => deleteJoke(joke.id)}> Delete </button>
     <li>{joke.type}: <JokeCard {joke} /></li>
   </div>
 {/each}
```

Svelte Store: Set/Update

Svelte nos ofrece diferentes formas de modificar el store:

- **update:** Modificamos la tienda usando una función.
- **set:** Modificamos la tienda usando un valor directo.

```
/* ./src/stores/index.ts */

export const postJoke = (joke: Content) => {
    if (joke.id === undefined || joke.id === null || joke.id
=== 0) {
        joke.id = getNewId();
        jokeArray.update((jokes) => {
            return [...jokes, joke];
        });
    }
};

export const deleteJoke = (id: number) => {
    jokeArray.update((jokes) => {
        return jokes.filter(joke => joke.id !== id);
    });
}
```

```
/* ./src/stores/index.ts */

export const clearJokes = () => {
    jokeArray.set([]);
};

const getNewId = () => {
    maxJokeId.update((id) => {
        return id + 1;
    });
    return latestId;
};
```

6

Lo que aprendimos 🧐