



# Tarea 6

# Svelte

Julio Andrade  
Gerardo Crot  
M<sup>a</sup> Josefa Espinoza



# Tabla de contenidos

**01**

**Demo**

**02**

**Rollup y Store**

**03**

**Interacción con  
componentes**

**04**

**Conclusiones y  
aprendizajes**



**01**

**Demo**



**02**

# **Rollup y Store**

# Rollup

Es el encargado de tomar los componentes de svelte y hacer el bundle para que el navegador los lea

Es importante definirlo porque asimila a un build de la aplicación



# Rollup

Svelte proporciona un Rollup inicial al crearlo.

Tuvimos que definir cómo extras el acceder a las variables de entorno y la configuración para usar archivos .json

```
rollup.config.js

// For importing json files
json({
  include: 'node_modules/**',
  exclude: [ 'node_modules/foo/**', 'node_modules/bar/**' ],
  preferConst: true,
  indent: ' ',
  compact: true,
  namedExports: true
}),
//
replace({
  // 2 level deep object should be stringify
  __myapp: JSON.stringify({
    env: {
      isProd: production,
      WEATHER_API_KEY: process.env.WEATHER_API_KEY,
    }
  })
}),
}),
```

# Store

Usamos los writable de Svelte para definirnos la store y sus mutaciones.

Definimos los métodos asíncronos dentro de las actions para mayor control de estas

```
store.js

import { writable, derived } from 'svelte/store';
import { getWeatherFromCity } from '../scripts/weather';

/** Store for your data.
This assumes the data you're pulling back will be an array.
If it's going to be an object, default this to an empty object.
**/

export const apiData = writable([]);
export const count = writable(0);
export const search = writable("");
export const weather = writable({});
export const searchVisibility = writable(true);
export const weatherSet = writable(false);
```

# Store

```
store.js

import { writable, derived } from 'svelte/store';
import { getWeatherFromCity } from '../scripts/weather';

/** Store for your data.
This assumes the data you're pulling back will be an array.
If it's going to be an object, default this to an empty object.
*/
export const apiData = writable([]);
export const count = writable(0);
export const search = writable("");
export const weather = writable({});
export const searchVisibility = writable(true);
export const weatherSet = writable(false);

// Mutations
export const changeSearch = (value) => {
  search.set(value);
};

export const changeWeather = (value) => {
  weather.set(value);
  console.log(value);
  weatherSet.set(true);
};

export const hideSearchBar = () => {
  searchVisibility.set(false);
};

export const showSearchBar = () => {
  searchVisibility.set(true);
};

export const unsetWeather = () => {
  weatherSet.set(false);
  weather.set({});
};

export const restoreSearch = () => {
  search.set("");
};
```

```
store.js

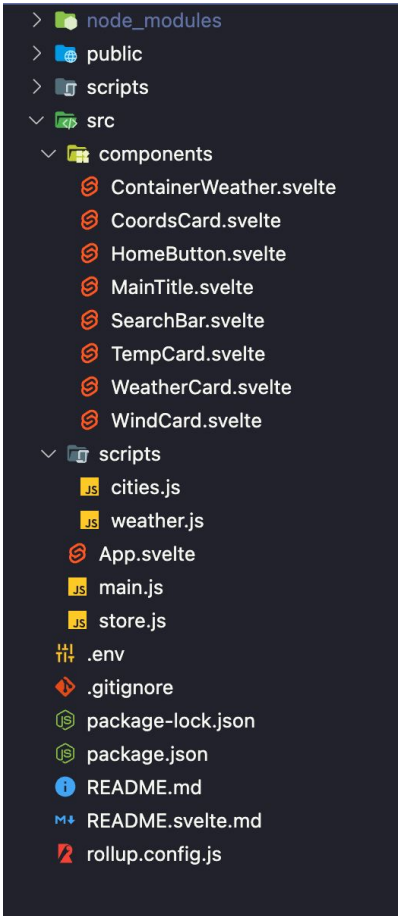
// Actions
export const makeSearch = async (value) => {
  const weatherData = await getWeatherFromCity(value);
  if (!weatherData) return null;
  changeSearch(value);
  hideSearchBar();
  changeWeather(weatherData);
};

export const goHome = () => {
  unsetWeather();
  restoreSearch();
  showSearchBar();
};
```



# Orden en el repositorio

Adicionalmente al store.js la aplicación contaba con distintos componentes y scripts para facilitar la interacción



```
> node_modules
> public
> scripts
└─ src
  └─ components
    ├── ContainerWeather.svelte
    ├── CoordsCard.svelte
    ├── HomeButton.svelte
    ├── MainTitle.svelte
    ├── SearchBar.svelte
    ├── TempCard.svelte
    ├── WeatherCard.svelte
    └── WindCard.svelte
  └─ scripts
    ├── cities.js
    ├── weather.js
    ├── App.svelte
    ├── main.js
    └── store.js
  ├── .env
  ├── .gitignore
  ├── package-lock.json
  ├── package.json
  ├── README.md
  ├── README.svelte.md
  └── rollup.config.js
```

The image shows a file explorer view of a project directory. The root directory contains several folders and files. The 'src' folder is expanded, showing two sub-folders: 'components' and 'scripts'. The 'components' folder contains eight Svelte component files: ContainerWeather.svelte, CoordsCard.svelte, HomeButton.svelte, MainTitle.svelte, SearchBar.svelte, TempCard.svelte, WeatherCard.svelte, and WindCard.svelte. The 'scripts' folder contains five files: cities.js, weather.js, App.svelte, main.js, and store.js. Below the 'src' folder, there are several configuration and documentation files: .env, .gitignore, package-lock.json, package.json, README.md, README.svelte.md, and rollup.config.js.



# 03

## Interacción con componentes

# Barra de búsqueda

Usamos un if para condicionar si se muestra el componente dependiendo del estado de la aplicación mediante un bind:value a un subscribe a la store.

Además de ejecutar una acción al apretar el botón submit.

```
src/components/SearchBar.svelte

<script>
import {
  search,
  makeSearch,
  searchVisibility
} from '../store.js';

let searchValue;
search.subscribe(value => {
  searchValue = value;
});

let searchVisibilityValue;
searchVisibility.subscribe(value => {
  searchVisibilityValue = value;
});
</script>

{#if searchVisibilityValue}
<div id="bar" class="search-wrapper input-group mb-3">
  <input
    class="input-search"
    bind:value={searchValue}
    type="text"
    placeholder="Busca un lugar"
  />
  <button
    class="btn btn-outline-secondary"
    type="submit"
    on:click={() => makeSearch(searchValue)}
  >
    Buscar
  </button>
</div>
{/if}
```

# Tarjetas

Ahora se condiciona su renderización mediante un if únicamente

Se utilizan métodos y subscribes para obtener la información

El poder definir todo previamente hizo el código mucho más legible

```
src/components/WeatherCard.svelte

<script>
import {
  weather,
  weatherSet,
} from '../store.js';

let weatherWeatherValue;
weather.subscribe((value) => {
  weatherWeatherValue = value.weather;
});

let weatherSetValue;
weatherSet.subscribe((value) => {
  weatherSetValue = value;
});

const getUrl = () => {
  const icon = weatherWeatherValue[0].icon;
  return `http://openweathermap.org/img/wn/${icon}@2x.png`;
};

const getDescription = () => {
  return weatherWeatherValue[0].description;
};
</script>

{#if weatherSetValue}
<div class="weather-card">
  <div class="weather-icon">
    <img alt="weather" src={ getUrl() } />
  </div>
  <div class="weather-info">
    <p>{ getDescription() }</p>
  </div>
</div>
{/if}
```

# Funciones útiles

Estas funciones son externalizadas al funcionamiento del store y proveen la información del clima

```
src/scripts/weather.js

export const getWeatherFromCity = async (city) => {
  const coords = await getCoords(city);
  if (coords.message) return coords.message;
  const url = `https://api.openweathermap.org/data/2.5/weather?lat=${coords.lat}&lon=${coords.lng}&appid=${apiKey}`;
  try {
    const response = await axios.get(url);
    return response.data;
  } catch (error) {
    console.log(error);
  }
};
```

```
src/scripts/cities.js

import cities from "cities.json";

export const getCoords = (city) => {
  const myCity = cities.filter(
    (c) => c.name.toLowerCase() === city.toLowerCase()
  );
  if (!myCity[0]) return null;
  return { lat: myCity[0].lat, lng: myCity[0].lng };
};
```

# Uso de estilos

El poder definir los estilos para cada componente, en estilo css, permite tener un mejor control sobre estos

```
src/components/WeatherCard.svelte

{#if weatherSetValue}
<div class="weather-card">
  <div class="weather-icon">
    <img alt="weather" src={ getUrl() } />
  </div>
  <div class="weather-info">
    <p>{ getDescription() }</p>
  </div>
</div>
{/if}

<style>
.weather-card {
  display: flex;
  flex-direction: row;
  align-items: center;
  justify-content: center;
  padding: 1rem;
  border-radius: 0.5rem;
  background-color: hsla(209, 82%, 52%, 0.63);
  font-size: auto;
  color: white;
  margin: 5px;
  height: 150px;
}
</style>
```



# **04**

## **Conclusiones y aprendizajes**

# Mucho más simple los bindings

Vincular valores de la store con  
elementos del HTML nunca fue más fácil.

Es cosa de hacer un bind y listo  
(parecido a Vue con el vue-model)

```
src/components/SearchBar.svelte

<script>
import {
  search,
  makeSearch,
  searchVisibility
} from '../store.js';

let searchValue;
search.subscribe(value => {
  searchValue = value;
});

let searchVisibilityValue;
searchVisibility.subscribe(value => {
  searchVisibilityValue = value;
});
</script>

{#if searchVisibilityValue}
<div id="bar" class="search-wrapper input-group mb-3">
  <input
    class="input-search"
    bind:value={searchValue}
    type="text"
    placeholder="Busca un lugar"
  />
  <button
    class="btn btn-outline-secondary"
    type="submit"
    on:click={() => makeSearch(searchValue)}
  >
    Buscar
  </button>
</div>
{/if}
```



# Facilidad de interacción

Nuestra opinión es que la interacción sobre todo entre componentes se facilita al tener centralizado el manejo de estados, sus cambios, acciones ,etc. En comparación a manejar dichos elementos en cada componente.

Uploaded using RayThis Extension

```
let weatherSetValue;  
weatherSet.subscribe((value) => {  
  weatherSetValue = value;  
});
```

Uploaded using RayThis Extension

```
<button  
  title="Volver a la barra de búsqueda"  
  on:click={() => goHome()}  
  class="home-button"  
>
```

# Rollup puede parecer intimidante

Nos encontramos con problemas con el archivo de configuración de rollup, donde definir por ejemplo cómo se manejan los json se configuran de forma distinta a como uno está acostumbrado a llegar e instalar una librería

```
rollup.config.js

export default {
  input: 'src/main.js',
  output: {
    sourcemap: true,
    format: 'iife',
    name: 'app',
    file: 'public/build/bundle.js'
  },
  plugins: [
    svelte({
      compilerOptions: {
        // enable run-time checks when not in production
        dev: !production
      }
    })
  ],
  // we'll extract any component CSS out into
  // a separate file - better for performance
  css: { output: 'bundle.css' },

  // If you have external dependencies installed from
  // npm, you'll most likely need these plugins. In
  // some cases you'll need additional configuration -
  // consult the documentation for details:
  // https://github.com/rollup/plugins/tree/master/packages/commonjs
  resolve: {
    browser: true,
    dedupe: ['svelte']
  },
  // For importing json files
  json: {
    include: 'node_modules/**',
    exclude: [ 'node_modules/foo/**', 'node_modules/bar/**' ],
    preferConst: true,
    indent: ' ',
    compact: true,
    namedExports: true
  },
  //
  replace: {
    // 2 level deep object should be stringify
    __myapp: JSON.stringify({
      env: {
        isProd: production,
        WEATHER_API_KEY: process.env.WEATHER_API_KEY,
      }
    })
  },
  commonjs(),

  // In dev mode, call 'npm run start' once
  // the bundle has been generated
  !production && serve(),

  // Watch the 'public' directory and refresh the
  // browser on changes when not in production
  !production && livereload('public'),

  // If we're building for production (npm run build
  // instead of npm run dev), minify
  production && terser()
},
watch: {
  clearScreen: false
}
};
```



**Dudas o comentarios**



# Tarea 6

# Svelte

Julio Andrade  
Gerardo Crot  
M<sup>a</sup> Josefa Espinoza

