React4teachers

2022 edition

1. React4teachers

From Zero 2 Hero

Conselleria d'educacio de les Illes Balears

By Alberto Soto

UD2 - Vanilla2React

2022

2. Contenido del curso

- 1.- Introducción
- 2.- De VanillaJs a React
- 3.- React framework: Motivos de exito
- 4.- React framework: Class components vs Function components
- 5.- React framework: State management - Control de el estado de la aplication
- 6.- React framework: Routing
- 7.- Storybook
- 8.- Monorepo

- 9.- TailwindCSS y StyledComponents
- 10.-Despliegue de aplicaciones

3. De VanillaJs a React y mas alla

Actualmente tenemos 3 framework principales para acelerar el desarrollo en entorno cliente, aunque en todo momento aparecen nuevos y mas eficientes.

- AngularJs, by Google
- React, by Facebook
- Vuejs
- Svelte

4. Angularjs

```
<div ng-app="">

<p>Name: <input type="text" ng-model="name"></p>
<p>You wrote: {{ name }}</p>

</div>
```

Angular es un framework implementado por Google que solía seguir una arquitectura MVC y que ha evolucionado a un framework MVVC. Al igual que Backbone, está enfocado a SPAs y refuerza la separación entre lógica de aplicación, lógica de visualización y plantillas.

Es interesante notar que Angular permite al programador extender el HTML creando nuevas etiquetas que pueden ser usadas en plantillas y que encapsulan partes de la página

5. Vuejs

Vue comenzó como un framework MVC similar a Angular pero enfocado a ser más simple y requerir menos boilerplate. La última versión ya cuenta con un enfoque basado en componentes y un virtual-DOM de forma similar a como lo hace React.

6. Vuejs

FormComponent.vue

```
<template>
  <div class="form pt-6">
    <div class="summary text-red" v-if="$v.form.$error">
      Form has errors
    </div>
    <form @submit.prevent="submit">
      <div class="flex justify-center my-6">
        <div
           class="px-4"
           :class="{ 'hasError': $v.form.name.$error }">
          <label class="mr-2 font-bold text-grey">Name</label>
          <input type="text" class="input" v-model="form.name">
        </div>
        <div
          class="px-4"
          :class="{ 'hasError': $v.form.email.$error }">
          <label class="mr-2 font-bold text-grey">Email</label>
          <input type="email" class="input" v-model="form.email">
        </div>
      </div>
      <div class="text-center">
        <button type="submit" class="button">
          Submit
        </button>
      </div>
    </form>
  </div>
</template>

<script>
import { required, email, minLength } from "vuelidate/lib/validators";

export default {
  name: "FormComponent",

  data() {
    return {
      form: {
        name: "",
        email: ""
      }
```

```
    };
  },

  validations: {
    form: {
      name: { required, min: minLength(10) },
      email: { required, email }
    }
  },

  methods: {
    submit() {
      this.$v.form.$touch();
      if(this.$v.form.$error) return
      // to form submit after this
      alert('Form submitted')
    }
  }
};
</script>
```

7. Svelte

Origin[1]

App.svelte

```
<script>
 let src = '/tutorial/image.gif';
 let name = 'Rick Astley';
</script>

<!-- {src} is short for src={src} -->
<img {src} alt="{name} dancing">
```

Mientras que los frameworks tradicionales como React y Vue realizan la mayor parte de su trabajo en el navegador, Svelte traslada ese trabajo a un paso de compilación que tiene lugar cuando se construye la aplicación.

[1] https://svelte.dev/examples/dynamic-attributes

En lugar de utilizar técnicas como la difusión virtual del DOM, Svelte escribe código que actualiza quirúrgicamente el DOM cuando el estado de tu aplicación cambia.

8. ReactJS

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

React fue el primer framework en utilizar un DOM virtual. Fue creado por Facebook y la arquitectura que deben seguir las aplicaciones que lo utilizan es mínima.

9. Introduciendo React

React permite realizar las siguientes tareas:

- Utilizarlo para apps de escritorio hibridas, a traves de electron
- Generar mobile Apps, a traves de Expo o React Native*
- Programar PWA o SPA

10. Mi primera pagina con react

Los primeros pasos con react son extremadamente sencillos:

https://reactjs.org/docs/create-a-new-react-app.html

```
npx create-react-app ch2-react-intro
```

```
cd my-app
npm start
```

Veamoslo en directo

11. Funcionamiento basico

- Tenemos un evento de inicio con vanilla
- Tenemos class components que heredan de React-component
- Soporta web vitals
- Ejecutamos el servidor con yarn start

Pregunta: que bundler crees que utiliza?

Respuesta[2]

12. App.js

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
```

[2] https://github.com/facebook/create-react-app#create-react-app--

```
  );
}

export default App;
```

13. React JSX

Desde React generaron una extension de fichero para dar soporte extendedido a los template literals, denominado JSX o TSX si utilizais Typescript.

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```

14. React JSX

Condicionantes:

- No tienes porque utilizar la sintaxis de template literals, ya que se activa por defecto
- Los nodos tienen que tener, como minimo un parent, en caso contrario utiliza un nodo vacio
- No utilizaremos algunos atributos html como class, que es palabra reservada, en su caso usaremos className
- Podemos utilizar capacidades js a partir de {}

15. Learn by practice

Ejercicio 1

No hay mejor manera de experimentar que encontrar errores mientras desarrollamos. Vamos a crear una pagina con componentes para ello:

- Haz las modificaciones necesarias para habilitar Tailwind[3]

[3] https://tailwindcss.com/docs/guides/create-react-app

- src/app.js El parent element y unico sera app.js
- src/layout/header.js Generaremos un header component
- src/main.js un main component que renderizara dependiendo del estado de la pagina

Distribuye en dichos elementos el siguiente codigo final, haciendd las modificaciones necesarias para los resultados de la leccion 1.

16. Ejercicio 1

```
<body class="text-gray-700 bg-white">
<nav>
    <div class="container mx-auto px-6 py-2 flex justify-between items-
center">
        <a class="font-bold text-2xl lg:text-4xl alternative-font"
 href="#">
            React 4 teachers
        </a>
        <div class="block lg:hidden">
            <button class="flex items-center px-3 py-2 border rounded
 text-gray-500 border-gray-600 hover:text-gray-800 hover:border-teal-500
 appearance-none focus:outline-none">
                <svg class="fill-current h-3 w-3" viewBox="0 0 20 20"
 xmlns="http://www.w3.org/2000/svg">
                    <title>React 4 teachers</title>
                    <path d="M0 3h20v2H0V3zm0 6h20v2H0V9zm0
 6h20v2H0v-2z"/>
                </svg>
            </button>
        </div>
        <div class="hidden lg:block">
            <ul class="inline-flex">
                <li><a class="px-4 font-bold" href="/">Home</a></li>
                <li><a class="px-4 hover:text-gray-800"
 href="#">Ejercicio 1</a></li>
                <li><a class="px-4 hover:text-gray-800"
 href="#">Ejercicio 2</a></li>
                <li><a class="px-4 hover:text-gray-800"
 href="#">Ejercicio 3</a></li>
            </ul>
        </div>
    </div>
</nav>
```

```
<div id="app" class="py-20" style="background: linear-gradient(90deg,
 #667eea 0%, #764ba2 100%)">
    <div class="container mx-auto px-6">
        <h2 class="text-4xl font-bold mb-2 text-white">
            React 4 teachers
        </h2>
        <h3 class="text-2xl mb-8 text-gray-200">
            Capitulo 1: Explorando vanillajs!
        </h3>
        <button class="bg-white font-bold rounded-full py-4 px-8 shadow-
lg uppercase tracking-wider">
            Quiero aprender react, no vanillajs!
        </button>
    </div>
</div>
</body>
```

17. Ejercicio 1

Consideraciones: - Los estilos en linea se deben evitar en toda regla: Convierte a un elemento comun - Cuando generamos componentes sus ciclos de vida son independientes. Eso quiere decir que puedes tener ficheros css por componente - Cuando usamos import (ECMA6) en lugar de require (ECMA5) debes tener en cuenta el tipo de export que realizas*

```
import {Hero} from "./components/Hero"
import Header from "./layout/Header"

export default function Header() {...}
export function Hero() {...}
```