UI5
Web Component

Sfruttare la potenza dei Web Components nell'ecosistema UI5

Salvatore La Spata (salvatore.la.spata@gmail.com)

Agenda

1. Cosa sono i Web Component?
2. Perchè SAP ha deciso di introdurre il supporto?
3. Caratteristiche Principali
4. Vantaggi - Compatibilità
5. Catalogo dei componenti
6. Live Demo

Cosa sono i **Web Components** e **perchè SAP** ha pensato **(Bene)** di integrare nel suo **ecosistema.**

Web Components - Web APIs

E’ una tecnologia che racchiude un insieme di specifiche standard per la creazione di elementi HTML **personalizzati e riutilizzabili** in pagine ed applicazioni Web.

Dal 2012 in Working Draft del **W3C**

Allo stato attuale i Web Component non si basano più su specifiche a se stanti, ma fanno riferimento agli standard di **HTML** e **DOM**, di cui ormai fanno parte integrante.

https://developer.mozilla.org/en-US/docs/Web/API/Web_components

Caratteristiche

Custom element

insieme di **API JavaScript** per la creazione di elementi del **DOM** personalizzati associati ad uno specifico tag **HTML**.

Shadow DOM

insieme di **API JavaScript** che consentono di gestire un **DOM** specifico per un componente, indipendente dal DOM della pagina Web, realizzando il meccanismo di incapsulamento a cui accennavamo prima.

HTML template

integrazione alle specifiche dell'**HTML** che consente di definire porzioni di markup che non viene interpretato al caricamento della pagina Web, ma che viene istanziato a runtime.

Vantaggi

A prova di futuro:

essendo **standard web**, sono **compatibili** con qualsiasi versione di qualsiasi framework di sviluppo web**.**

Incapsulati:

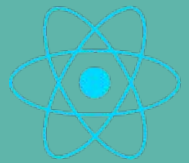
l'HTML/CSS nel DOM è protetto dall'interferenza della pagina web e viceversa, il che li rende **stabili in qualsiasi ambiente** e adatti non solo alle applicazioni, ma anche alle librerie e ai micro-frontend.

Eleganti:

elementi HTML personalizzati, nascondono la complessità dell'implementazione dietro un singolo tag HTML, rendendoli **facilmente utilizzabili con le API DOM standard**.

Compatibilità

Frameworks:




Browsers:




Catalogo dei componenti.

Grazie al playground è possibile provare e modificare in real time i sample
(grazie allo strumento Storybook)

N.B. Non sono presenti tutti i componenti del framework **SAPUI5.**

Tutto bello… ma come si usano?

Sono distribuiti come moduli ES6 tramite il package manager NPM:

- `@ui5/webcomponents` - libreria di componenti base (buttons, inputs, pickers, etc.);
- `@ui5/webcomponents-fiori` - libreria di componenti avanzati, utilities, etc.
- `@ui5/webcomponents-icons` - collezione di icone (general-purpose and business-oriented icons).

Per sviluppare un progetto utilizziamo un build tools chiamato Vite JS* oppure è possibile usare Yeoman**

E' possibile prototipare e sviluppare anche su Stack Blitz (un ide online pre-configurato)

* `npm create vite@latest`

** `npm install -g yo generator-easy-ui5` → `yo easy-ui5`

Architettura di UI5 Web Component

properties / attributes
 	
SAPUI5→**Properties**
	`<ui5-checkbox id="cb" value-state="Error"></ui5-checkbox>`
	
	`const myCb = document.getElementById("cb");`
	`myCb.valueState = "Error";`
	`myCb.setAttribute("value-state", "Error");`

slots
 	
SAPUI5→**Aggregation**
	`<ui5-popover>`
	`  <div slot="header">This will be used as a header</div>`
	`  <div>Some popover content</div>`
	`  This text will also go to the default slot.`
	`  <div slot="footer">`
	`  <ui5-button>Do some action</ui5-button>`
	`  </div>`
	`</ui5-popover>`

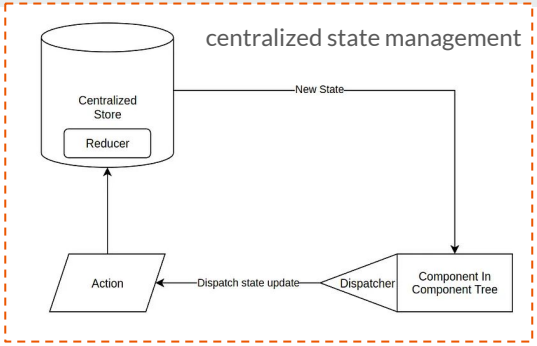
Architettura di UI5 Web Component

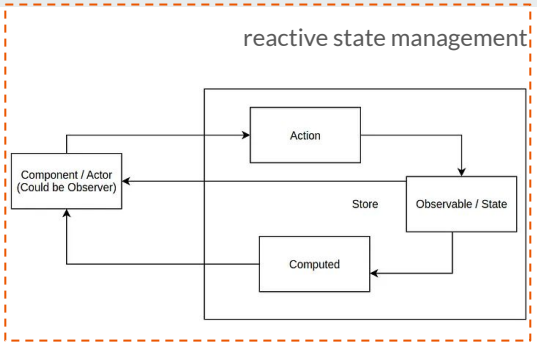
events
 	
SAPUI5→Events
	`const myMessage = document.getElementsByTagName("ui5-message-strip")[0];`
	
	`myMessage.addEventListener("close", () => {`
	`  console.log("The user dismissed the message");`
	`});`

public methods
 	
SAPUI5→Methods
	`const myDialog = document.getElementsByTagName("ui5-dialog")[0];`
	
	`myDialog.show();`

Il **JSONModel** usato in SAPUI5 nelle librerie/framework moderni viene gestito tramite dallo **state manager**

State management





`Centralized`
React Context
Redux

Offre un contenitore di stati centralizzato con un flusso di dati unidirezionale, enfatizzando l'immutabilità e le funzioni pure.

`Reactive`
Mobx

Fornisce aggiornamenti automatici dei componenti in base ai cambiamenti di stato osservati, con particolare attenzione al modello di programmazione dichiarativo e alla gestione di valori e azioni calcolate.

`Atomic`
Recoil Js
Jotai

Utilizza singoli atomi per rappresentare lo stato, promuovendo la collocazione dello stato con i componenti e un approccio leggero e diretto alla gestione dello stato.

Vediamolo all’opera

UI5 Web component vs Sapui5 freestyle

```
import "@ui5/webcomponents/dist/Button.js";
const App = () => {
  return (
    <>
      <ui5-button design="Emphasized">
            Hello UI5 Web Components
      </ui5-button>
    </>
  )
}

export default App
```

```
<mvc:View
  controllerName="com.myorg.myui5app.controller.App"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns:mvc="sap.ui.core.mvc"
  displayBlock="true"
  xmlns="sap.m">
  <App id="app">
    <Button text="Hello World"
      type="Emphasized" />
  </App>
</mvc:View>
```

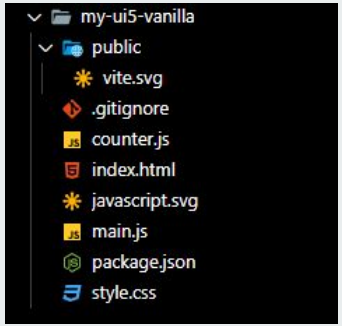
```
sap.ui.define(["sap/ui/core/mvc/Controller"], function (BaseController) {
    "use strict";
    return BaseController.extend("com.myorg.myui5app.controller.App", {
        onInit() {},
    });
});
```

Vanilla Javascript

npm create vite@latest

- Project Name: my-ui5-vanilla
- Framework: Vanilla
- Variant: JavaScript

```
√ Project name: ... my-ui5-vanilla
√ Select a framework: » Vanilla
√ Select a variant: » JavaScript
```



Installare quindi la dependenza:

```
npm install @ui5/webcomponents
```

Importare la dependenza e quindi utilizzare i componenti:

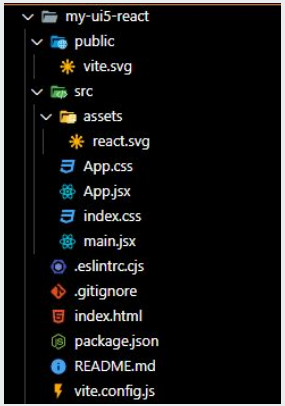
```
import "@ui5/webcomponents/dist/Button.js";

document.querySelector('#app').innerHTML = `
  <ui5-button design="Emphasized" >Hello UI5 Web Components</ui5-button>
`;

// path: my-ui5-vanilla/main.js
```

React JS

npm create vite@latest

- Project Name: my-ui5-vanilla
- Framework: React
- Variant: JavaScript

```
√ Project name: ... my-ui5-react
√ Select a framework: » React
√ Select a variant: » JavaScript
```



Installare quindi la dipendenza:

```
npm install @ui5/webcomponents
```

Importare la dipendenza e quindi utilizzare i componenti:

```jsx
import "@ui5/webcomponents/dist/Button.js";
const App = () => {
  return (
    <>
      <ui5-button design="Emphasized">Hello UI5 Web Components</ui5-button>
    </>
  )
}

export default App

// Path: my-ui5-react/src/App.jsx
```

Tutto questo grazie all'**open source.**

https://github.com/SAP/ui5-webcomponents

Risorse utili

[Fiori for Web Design Guidelines]

UI Elements - SAP Web Components NEW Overview.

[Official Documentation (Playground)]

https://sap.github.io/ui5-webcomponents/

[GitHub Repository Documentation]

SAP/ui5-webcomponents

[Tutorial Sap Developers]

https://developers.sap.com/tutorials/ui5-webcomponents-react-introduction.html

Conclusioni

Come in tutti gli strumenti non è la soluzione definitiva ma è sicuramente una valida alternativa alla complessità e dello scaffolding del framework **SAPUI5.**

Live Coding