DHTML DI LIVELLO SUPERIORE: ESPANDERE HTML CON I WEB COMPONENTS

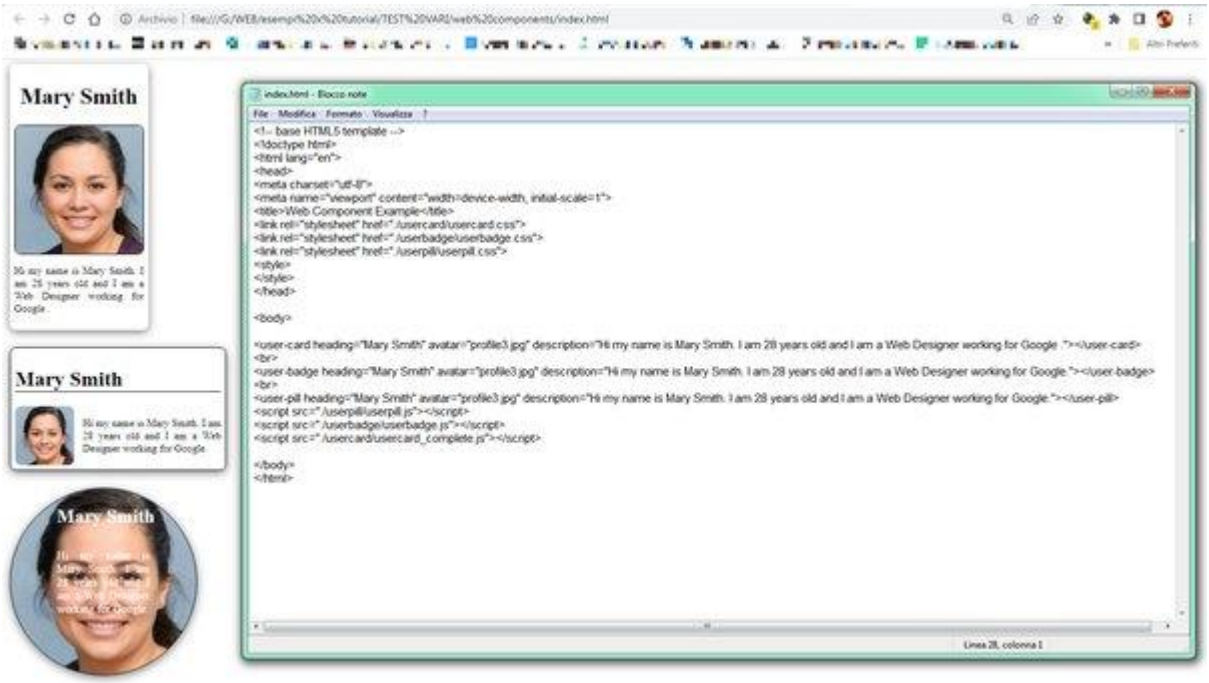
I componenti Web sono una suite di diverse tecnologie che consente di creare elementi personalizzati riutilizzabili nelle pagine Web.

Come sviluppatori, sappiamo tutti che riutilizzare il codice il più possibile è una buona idea. Questo tradizionalmente non è stato così facile per le strutture di mark-up personalizzate: pensiamo a porzioni complesse di HTML (e allo stile e allo script associati) che a volte abbiamo dovuto scrivere per eseguire il rendering dei controlli dell'interfaccia utente personalizzati e a come usarli più volte può trasformare la nostra pagina web in un pasticcio, se non prestiamo attenzione.

La tecnica dei **Web Components** mira a risolvere tali problemi: è costituito da tre tecnologie principali, che possono essere utilizzate insieme per creare elementi personalizzati versatili con funzionalità incapsulate che possono essere riutilizzati dove vuoi senza timore di collisioni di codice.

- **Custom Tags**: un insieme di API JavaScript che consentono di definire elementi personalizzati e il loro comportamento, che possono quindi essere utilizzati come desiderato nella interfaccia utente.
- **Shadow DOM**: un insieme di API JavaScript per collegare un albero DOM "ombra" incapsulato a un elemento — che viene visualizzato separatamente dal DOM del documento principale — e controllare le funzionalità associate. In questo modo, puoi mantenere private le caratteristiche di un elemento, in modo che possano essere script e stilizzati senza il timore di collisioni con altre parti del documento.
- **Template HTML**: gli elementi <template>e <slot> consentono di scrivere modelli di mark-up che non vengono visualizzati nella pagina sottoposta a rendering. Questi possono quindi essere riutilizzati più volte come base della struttura di un elemento personalizzato.

La pagina qui sotto é stata creata con 3 Web Components creati tramite la prima tecnologia, quella dei Custom Tags, senza utilizzo di Template o di Shadow DOM.



Le regole essenziali per la creazione di Web Components di questo tipo sono solo tre:

1. utilizzare tags custom che contengano un trattino '-';
2. dichiarare una classe che definisce il web component da assegnare al custom tag
3. definire il contenuto del custom tag

Come esempio si porta il codice del Web Component *user-badge*, con tag **<user-badge>**:

```
1. class userbadge extends HTMLElement {
2. constructor() {
3. super();
4. this.innerHTML = `<div class=userbadge><h1> ${this.getAttribute('heading')} </h1><div class=box><img src= ${this.getAttribute('avatar')} /><p> ${this.getAttribute('description')} </p></div></div>`;
5. }
6. }
7. window.customElements.define('user-badge', userbadge);
```

Come potete vedere, la riga 1 definisce la classe *userbadge,* che viene assegnata al custom tag *user-badge* nella riga 7. La riga 4 si occupa di comporre il contenuto HTML inserendo la chiamata ai parametri passati dal codice HTML:

```
<user-badge heading="Mary Smith" avatar="profile3.jpg" description="Hi my name is Mary Smith. I am 28 years old and I am a Web Designer working for Google."></user-badge>
```

Il documento HTML dovrà ovviamente contenere il richiamo al file JavaScript che contiene la definizione del tag,

```
<script src="userbadge.js"></script>
```

e può contenere formattazione CSS interna o esterna tramite file.

```
<link rel="stylesheet" href="userbadge.css">
```

La sola limitazione di questa tecnologia Custom Tags é che non si può aggiungere facilmente contenuto tra i tag di apertura e di chiusura dell' elemento; ciò é superabile con una escamotage che prevede il salvataggio del contenuto interno in una variabile tramite JavaScript, con una espressione del tipo:

```
var hcustomContent = userbadge[i].innerHTML;
```

il cui contenuto andrà inserito nel concatenamento della definizione del contenuto del tag (riga 4 nel codice completo) con una procedura del tipo:

```
var hcustomContent = this.innerHTML;
this.innerHTML = `<div class=userbadge><h1> ${this.getAttribute('heading')}
</h1><div class=box><img src= ${this.getAttribute('avatar')} /><p>` +
hcustomContent + `</p></div></div>`;
```

Qui però le cose diventano abbastanza complesse quando si inseriscono sia testo che ulteriori tags HTML tra i tags di apertura e chiusura.

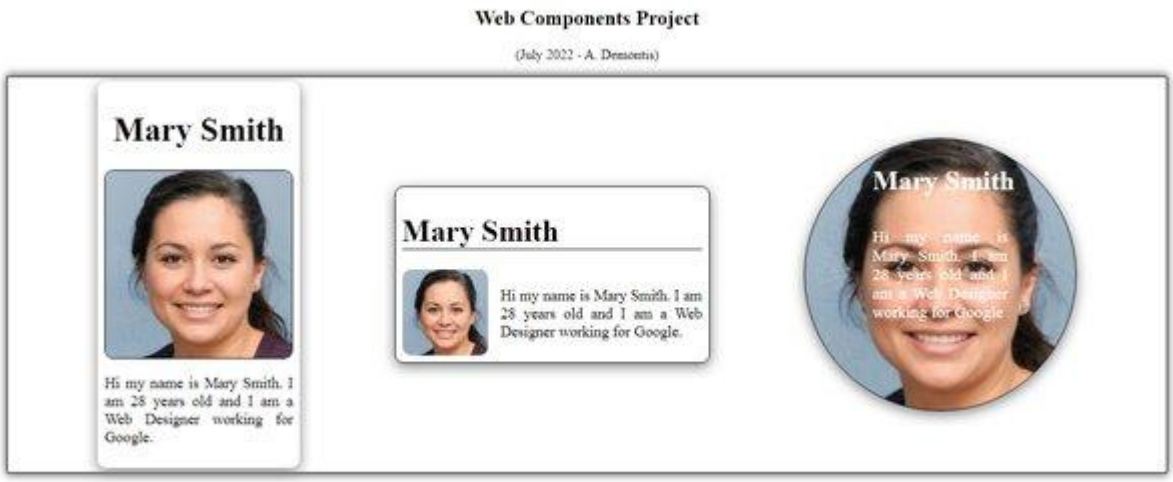
Come esempio si riporta ancora il codice del Web Component *user-badge*, con tag **<user-badge>** nella sua versione finale modificata per poter contenere testo ed elementi HTML tra i tags di apertura e chiusura:

```
1. class userbadge extends HTMLElement {
2. constructor() {
3. super();
4.
5. var hcustomContent = this.innerHTML;
6. this.innerHTML = `<div class=userbadge><h1> ${this.getAttribute('heading')}
   </h1><div class=box><img src= ${this.getAttribute('avatar')} /><p>` +
   hcustomContent + `</p></div></div>`;
7. }
8. }
9. window.customElements.define('user-badge', userbadge);
```

Al quale ovviamente farà capo un file HTML leggermente modificato per sfruttare la nuova struttura:

```
<user-badge heading="Mary Smith" avatar="profile3.jpg">
Hi my name is Mary Smith. I am 28 years old and I am a Web Designer
working for <b>Google</b>.
</user-badge>
```

Si veda per maggiori dettagli la pagina del Progetto Web Components.



A. Demontis - 21 luglio 2022