SUBFLASH REBORN

TALLER 02

WEB COMPONENTS

Libérate de la fontanería digital con Google LitElements

Por **Fernando García**

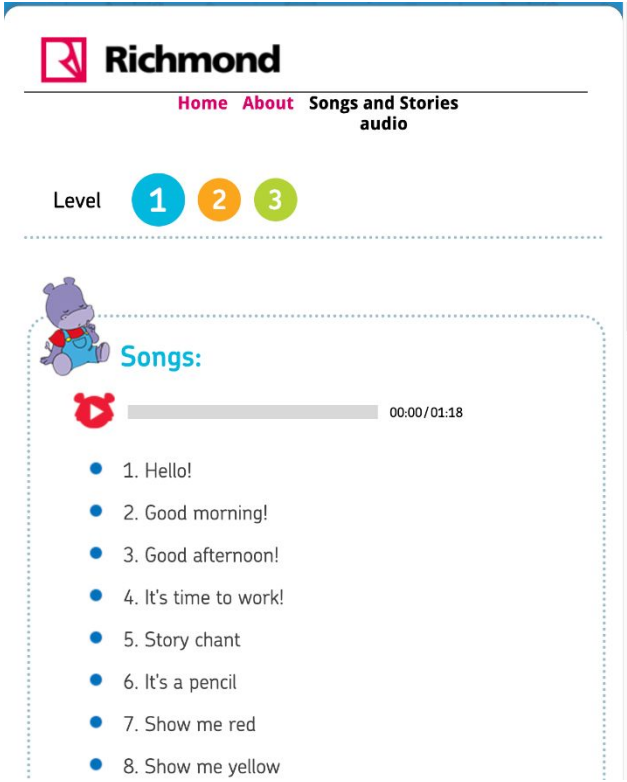
JACA 2024

www.subflash.com

Web components - ¿Porqué? ¿Qué solucionan?

Las páginas web están llenas de porciones de código html repetitivo al que hay que dar estilo y modificar dinámicamente.

Web components - ¿Porqué? ¿Qué solucionan?



Web components - ¿Porqué? ¿Qué solucionan?

La primera dificultad es tener que crear todo ese código html directamente con javascript para poder reutilizarlo.

Esta razón está en la raíz de creación de frameworks enteros como React, Angular, Vue, Ember, Backbone…

Web components - ¿Porqué? ¿Qué solucionan?

Y la segunda es crear hojas de estilo css siguiendo convenciones de nombres suficientemente consistentes para no provocar conflictos con el resto de la página.

Esto ha dado lugar también a otros frameworks y como Bootstrap o Tailwind y a convenciones de nombres como bem.

Web components - ¿Porqué? ¿Qué solucionan?

Problema:
Cada framework de javascript ha optado por su filosofía particular. Hay que aprenderlos uno a uno y son poco o nada compatibles entre ellos.

Y con css pasa exactamente los mismo, cada librería de componentes ha inventado sus propias clases, o formas de cómo dar estilo a los componentes y hay que aprenderlos uno a uno.

Web components - ¿Porqué? ¿Qué solucionan?

Como efecto secundario y esto es un offtopic de opinión personal, hoy tenemos desarrolladores y diseñadores que son super expertos en uno o dos de esos frameworks, hasta el punto de que algunos ni siquiera son conscientes de que por debajo todo sigue siendo javascript + css totalmente estándar.

Web components - ¿Porqué? ¿Qué solucionan?

El standard web components se basa en tres tecnologías:

- **Custom Elements**, que permite definir elementos personalizados
- **Shadow Dom**, que permite aislar la css del resto del documento
- **HTML templates**, que permite crear plantillas que reutilizables que no se renderizan directamente en la página

Web components - Custom Elements

- Registro de nombres para las etiquetas que inventemos
- Deben contener obligatoriamente un guión en su nombre
- Una vez registrado, el componente web se comporta como cualquier otra etiqueta HTML nativa

Web components - Shadow dom

- Ninguna hoja de estilo externa afecta al interior del web component y viceversa
- Tampoco colisionan los nombres de clases o identificadores (id) con los definidos fuera del nodo de shadow dom
- Con una excepción, las variables css son la puerta para poder dar estilo a los web components desde el exterior
- Existe el shadow dom open y el shadow dom closed

Web components - HTML templates

Definen la etiqueta <template> y <slot>

- La etiqueta **`template`** permite tener en la página elementos HTML que no se pintan, solo sirven como plantillas para crear otros elementos.

- La etiqueta **`slot`** permite componer unos elementos dentro de otros, facilitando la creación de elementos contenedores.

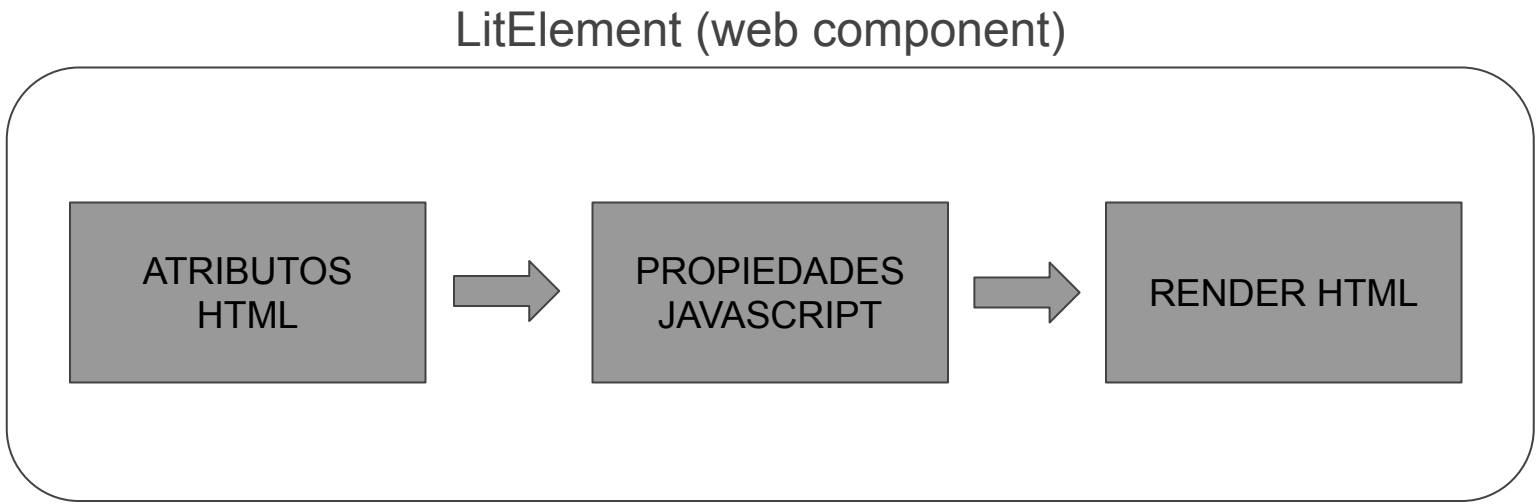
Web components - LitElement

Estas tres tecnologías son un estándar, funcionan en cualquier navegador medianamente moderno y son lo único que necesitamos, junto con algo de conocimiento de javascript para crear componentes web.

Pero lo cierto es que utilizarlas “a pelo” es muy laborioso.

Web components - LitElement

Y para facilitar la creación de los web components, google ha creado la librería llamada LitElement.



LitElement - propiedades

```
export class SfMensaje extends LitElement {

  static get properties() {
    return {
      texto: { type: String },
      variante: { type: String },
      time: { type: String }
    }
  }
```

```
<sf-mensaje
    texto="Hola Don Pepito"
    variante="recibido"
    time="0:52:13"
></sf-mensaje>
```

web components - estilo encapsulado

```
export class SfMensaje extends LitElement {

  static get styles() {
    return css`
      .contenedor {
        padding: 8px;
        border-radius: 4px;
      }

      .enviado {
        background-color: aquamarine;
        text-align: end;
      }

      .time {
        font-size: 12px;
      }
    `;
  }
```

```
render() {
  return html`
      <div class="contenedor ${ this.variante }">
          <span>${ this.texto }</span><br />
          <span class="time">${ this.time }</span>
      </div>
  `;
}
```

PLAIN HTML

```html
<section class="contenedor-mensajes">
    <div class="contenedor recibido">
        <span>Hola Don Pepito</span>
        <br>
        <span class="time">0:49:24</span>
    </div>
    <div class="contenedor enviado">
        <span>Hola Don José</span>
        <br>
        <span class="time">0:49:26</span>
    </div>
    <div class="contenedor enviado">
        <span>Pasó usted ya por casa</span>
        <br>
        <span class="time">0:49:28</span>
    </div>
    <div class="contenedor recibido">
        <span>Por su casa yo pasé</span>
        <br>
        <span class="time">0:49:30</span>
    </div>
</section>
```

WEB COMPONENTS

```html
<section class="contenedor-mensajes">
    <sf-mensaje
        texto="Hola Don Pepito"
        variante="recibido"
        time="0:52:13"
    ></sf-mensaje>
    <sf-mensaje
        texto="Hola Don José"
        variante="enviado"
        time="0:52:20"
    ></sf-mensaje>
    <sf-mensaje
        texto="Pasó usted ya por casa"
        variante="recibido"
        time="0:52:27"
    ></sf-mensaje>
    <sf-mensaje
        texto="Por su casa yo pasé"
        variante="enviado"
        time="0:52:38"
    ></sf-mensaje>
</section>
```

web components - … y mucho más

- solo se repinta lo que cambia
- ciclo de vida para entrometerse antes y despues
- eventos personalizados
- propiedades complejas (arrays, objetos, …)
- relación entre atributos html y propiedades javascript
- compatibilidad 100% entre navegadores
- no quieres shadow dom, utiliza light dom
- slots con nombre
- directivas en la librería de plantillas lit-html
- …