



IIC3585-1
DCC

Tarea 4

Web Components

GRUPO 5: SFEIR - MUÑOZ - VALDÉS

Objetivo

<suscripcion>

The screenshot displays a pricing table with four columns representing different subscription tiers. Each tier includes a name, a description of the number of monthly visits, a price per month, a 'Get started' button, and a list of features. The 'Freelance' plan is highlighted as the 'Best' option.

Basic	Freelance Best	Business	Enterprise
500 monthly visits	10 000 monthly visits	100 000 monthly visits	1 000 000+ monthly visits
\$0 / mo \$0 incl. 21% Vat	\$16 / mo \$18 incl. 21% Vat	\$49 / mo \$56 incl. 21% Vat	\$129 / mo \$146 incl. 21% Vat
Get started	Get started	Get started	Get started
Features included: <ul style="list-style-type: none">✓ Daily visits✓ Monthly visits reporting✓ Google Analytics Integration✓ Facebook Integration✓ Basic Dashboard New✓ Email Support	Basic plan Plus: <ul style="list-style-type: none">✓ Twitter integration✓ Instagram integration✓ Advanced Dashboard✓ Acquisitions Dashboard New✓ Client Billing New✓ Client Dashboard✓ Reporting tool✓ Multiple accounts✓ Support	Freelance plan Plus: <ul style="list-style-type: none">✓ Competitor Dashboard✓ Behavior Dashboard✓ SEO Dashboard✓ Full API controll New✓ Advanced Reporting New✓ User permissions✓ Advanced Support	Business plan Plus: <ul style="list-style-type: none">✓ Custom Domain✓ White Label solution✓ Rich Snippets✓ Team Insights Dashboard✓ 24/7 Support New



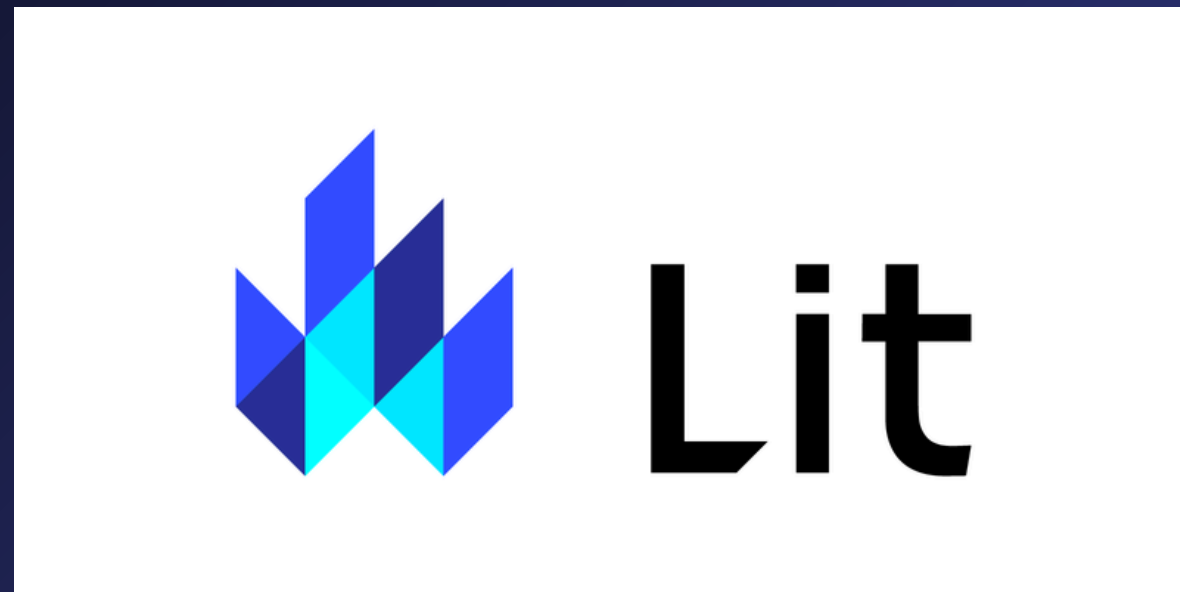
<acordion>

The screenshot shows an accordion interface with three items. The first item is expanded, displaying its description. The second and third items are collapsed, showing only their titles. The third item is highlighted with a blue header.

- ▼ The first item
Here's the description of item one
- ▶ The second item
- ▶ The third, longer item
- ▶ The first item
- ▼ The second item
Here's the description of item two
- ▶ The third, longer item
- ▶ The first item
- ▶ The second item
- ▼ The third, longer item
[Web Components](#) is a suite of different technologies allowing you to create reusable custom elements (with their functionality encapsulated away from the rest of your code) and utilize them in your web apps.

Mini-Aplicación Suscripciones

Lit



+

Standard Elements



<subscription>

Standard Elements - Template HTML

```
const templateSubscription = document.createElement('template');  
templateSubscription.innerHTML = `...`;
```

- Define la **estructura y estilo** base del componente, que se inyectará en su **Shadow DOM**.
- Su contenido no se renderiza automáticamente: es **clonado dinámicamente** usando `cloneNode(true)`.
- **style**: encapsula todas las reglas CSS, incluyendo **responsividad** y **condicionales vía atributos**.
- **Estructura HTML .card**: contiene subelementos dinámicos (.plan-name, .price, .features, etc.), que son **referenciados** en el script.

<suscription>

Standard Elements - Template HTML

```
<div class="card">
  <div class="ribbon">Best</div>
  <div class="plan-name"></div>
  <div class="visits"></div>
  <div class="pricing">
    <div class="price"></div>
    <div class="tax-included"></div>
  </div>
  <div class="description"></div>
  <ul class="features"></ul>
  <button class="cta-button">Get Started</button>
</div>
```

<subscription>

Standard Elements - SubscriptionCard class

```
class SubscriptionCard extends HTMLElement {  
  constructor() {  
    ...  
  }  
}
```

- Define un nuevo **custom element** que extiende la clase base `HTMLElement`, lo cual es requerido por la API de Web Components.
- **constructor()**: se ejecuta cuando se **instancia** el componente (por ejemplo, cuando escribimos `<subscription-card>` en el HTML).

<subscription>

Standard Elements - Shadow DOM

```
this._shadow = this.attachShadow({ mode: 'open' });  
this._shadow.appendChild(templateSubscription.content.cloneNode(true));
```

- Se encapsula el contenido dentro de un **shadow** tree, impidiendo colisiones de estilos o comportamiento con el DOM externo.
- mode: 'open' permite acceso a this.shadowRoot.

<subscription>

Standard Elements - Referencias internas

Almacenan referencias a nodos internos del componente para poder actualizarlos luego de forma dinámica:

```
this._planNameEl = this._shadow.querySelector('.plan-name');  
this._visitsEl = this._shadow.querySelector('.visits');  
this._priceEl = this._shadow.querySelector('.price');  
this._taxIncludedEl = this._shadow.querySelector('.tax-included');  
this._descriptionEl = this._shadow.querySelector('.description');  
this._featuresEl = this._shadow.querySelector('.features');  
this._btn = this._shadow.querySelector('.cta-button');
```

- Esto evita buscar repetidamente los elementos cada vez que cambian los atributos.
- El acceso se hace **dentro del shadow DOM**, por eso usamos `this._shadow.querySelector(...)`.

<subscription>

Standard Elements - Custom Element

```
customElements.define('subscription-card', SubscriptionCard);
```

- Registra el componente como una **etiqueta HTML válida** (<subscription-card>).
- Se basa en la API Custom Elements del estándar de Web Components.

<subscription>

Standard Elements - Ciclo de vida y reactividad

observedAttributes

```
static get observedAttributes() {  
  return ['plan', 'visits', 'price', 'tax-included', 'description', 'features', 'highlighted'];  
}
```

- Habilita la ejecución de `attributeChangedCallback()` cuando se modifica alguno de estos atributos.
- Esto genera un **ciclo reactivo declarativo**, sin necesidad de librerías externas.

<subscription>

Standard Elements - Ciclo de vida y reactividad

connectedCallback() / disconnectedCallback()

```
connectedCallback() {  
  this._updateContent();  
  this._btn.addEventListener('click', this._onClick);  
}  
  
disconnectedCallback() {  
  this._btn.removeEventListener('click', this._onClick);  
}
```

- ***connectedCallback***: inicializa contenido y listeners cuando el componente entra al DOM.
- ***disconnectedCallback***: limpia eventos cuando se remueve del DOM (buen manejo de memoria).

<subscription>

Standard Elements - Ciclo de vida y reactividad

attributeChangedCallback

```
attributeChangedCallback(name, oldValue, newValue) {  
  if (oldValue !== newValue) {  
    this._updateContent();  
  }  
}
```

Permite una actualización reactiva, regenerando el contenido cada vez que cambian los atributos relevantes.

<subscription>

Standard Elements - Funciones utilitarias privadas

```
_formatVisits(), _formatPrice(), _calculatePriceWithTax()
```

- **_formatVisits**: interpreta valores numéricos para visitas mensuales (e.g., 0 → “Visitas ilimitadas”).
- **_formatPrice**: transforma el número a una cadena con símbolo \$.
- **_calculatePriceWithTax**: aplica un 19% de IVA al valor base y retorna el precio final como string.

<suscription>

Standard Elements - Método _updateContent()

```
_updateContent() {  
    // Actualiza el DOM shadow en base a atributos HTML  
}
```

- Actualiza el textContent de cada sección.
- Aplica o remueve la clase .visible para mostrar/ocultar elementos opcionales.
- Parsea el atributo features como JSON, y genera dinámicamente los con íconos y textos.
- Gestiona el atributo booleano tax-included.

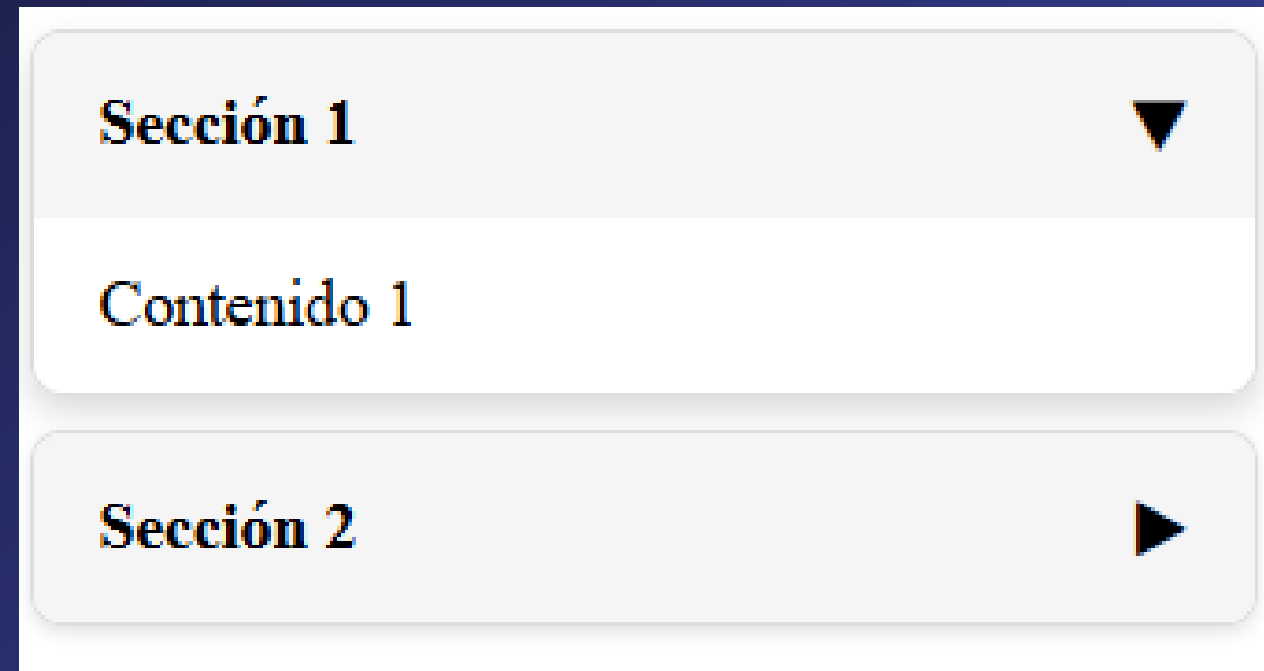
<subscription>

Standard Elements - Evento personalizado

```
this.dispatchEvent(new CustomEvent('subscribe', { detail, bubbles: true, composed: true }));
```

- Lanza un evento subscribe al hacer click en el botón.
- **bubbles: true:** el evento se propaga hacia arriba en el DOM.
- **composed: true:** el evento puede salir del Shadow DOM y ser capturado desde el DOM principal.
- **detail:** incluye datos clave del plan (plan, price, visits, taxIncluded), útiles para listeners externos.

Implementación <acordion>



Tecnologías y estándares aplicados

- Custom Elements API (`customElements.define`)
- Shadow DOM (`attachShadow({ mode: 'open' })`)
- HTML Templates (`template.innerHTML`)
- Eventos personalizados (`CustomEvent`)

Estructura del Componente

```
<acordion-list>  
  <acordion-item header="Sección 1">Contenido 1</acordion-item>  
  <acordion-item header="Sección 2">Contenido 2</acordion-item>  
</acordion-list>
```

- <acordion-item> es un custom element autónomo
- <acordion-list> actúa como coordinador del estado (escucha eventos y aplica lógica de apertura/cierre)

Ciclo de Vida

Constructor

- Se crea un Shadow DOM aislado
- Se inyecta el HTML + estilos vía `template.innerHTML`

```
class AcordionItem extends HTMLElement {
  constructor() {
    super();
    const template = document.createElement('template');
    template.innerHTML = `
      <style>
        .item {
          border: 1px solid #ddd;
          margin: 8px 0;
          border-radius: 8px;
          overflow: hidden;
          box-shadow: 0 2px 4px 0 rgba(0,0,0,0.1);
          transition: box-shadow 0.3s ease;
          width: 100%;
          height: auto;
          max-width: 300px;
        }
      </style>
    `;
  }
}
```

Ciclo de Vida

Connected Callback

- Se lee el atributo header.
- Se asigna comportamiento al evento click.

```
connectedCallback() {  
  const header = this.shadowRoot.querySelector('.header');  
  const title = this.getAttribute('header') || 'Título';  
  this.shadowRoot.querySelector('.title').textContent = title;  
  
  header.addEventListener('click', () => {  
    this.dispatchEvent(new CustomEvent('toggle', {  
      bubbles: true,  
      composed: true  
    }));  
  });  
}
```


Comunicación entre componentes

```
this.dispatchEvent(new CustomEvent('toggle', {  
  bubbles: true,  
  composed: true  
}));
```

- Cada accordion-item emite un evento toggle
- El evento viaja hasta accordion-list, que actúa como controlador central
- **bubbles: true:** el evento se propaga hacia arriba en el DOM.
- **composed: true:** el evento puede salir del Shadow DOM y ser capturado desde el DOM principal.

Comunicación entre componentes

Lógica en acordeon-list cierra todos los ítems y abre solo el clickeado:

```
class Acorcion extends HTMLElement {  
  connectedCallback() {  
    this.addEventListener('toggle', (e) => {  
      const items = this.querySelectorAll('acordion-item');  
      items.forEach(item => item.close());  
      const item = e.target.closest('acordion-item');  
      if (item) item.open();  
    });  
  }  
}
```

Lit v/s Standard Elements

Característica	LitElement	HTMLElement clásico
Renderizado reactivo	Automático con <code>render()</code>	Manual (<code>innerHTML</code> , <code>querySelector</code>)
Observación de props	Declarativa (<code>properties</code>)	Manual con <code>attributeChangedCallback</code>
Estilos	Declarativo y scoped (<code>static styles</code>)	En template o adjunto al Shadow DOM
Eventos	<code>@click="\${...}"</code>	<code>addEventListener()</code> manual
Plantilla HTML	<code>html\`` (template literals)</code>	<code><template></code> + <code>cloneNode()</code>