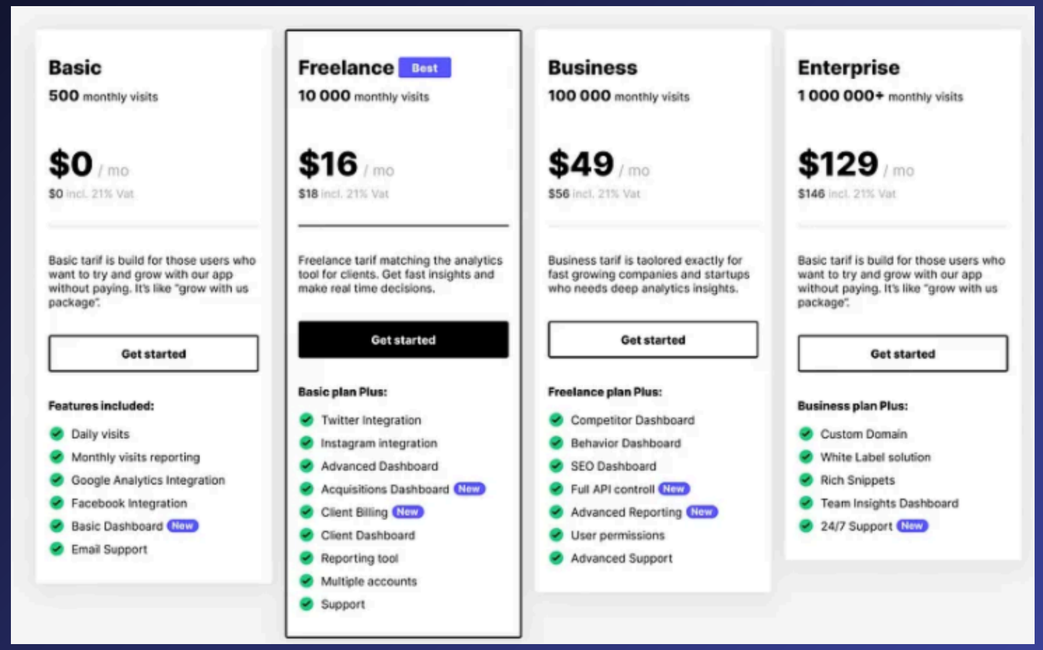
IIC3585-1
DCC

Tarea 4
Web Components

GRUPO 5: SFEIR - MUÑOZ - VALDÉS

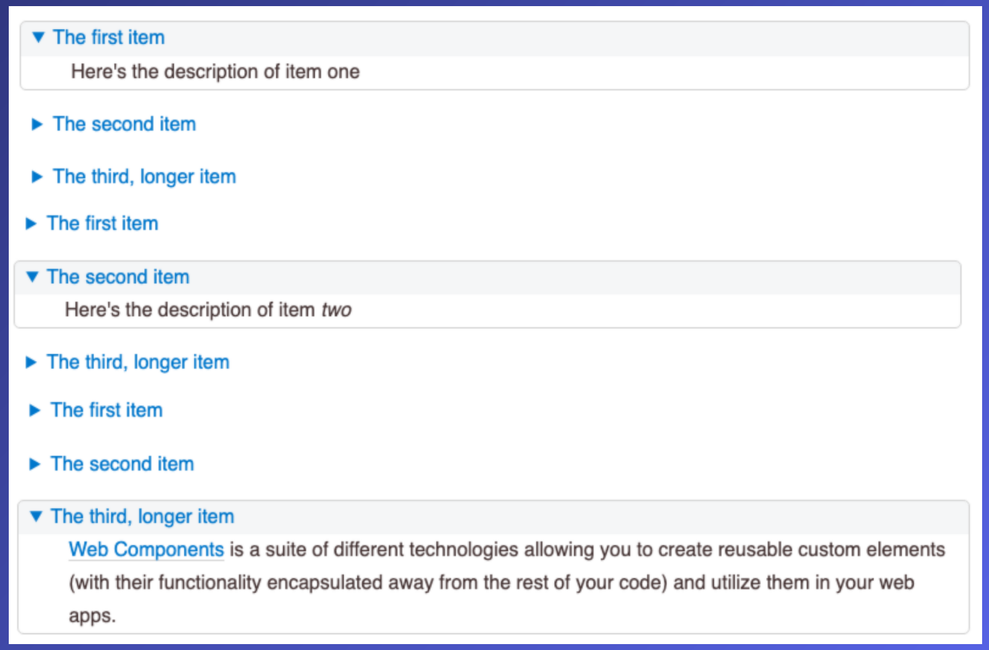
Objetivo

<suscripcion>



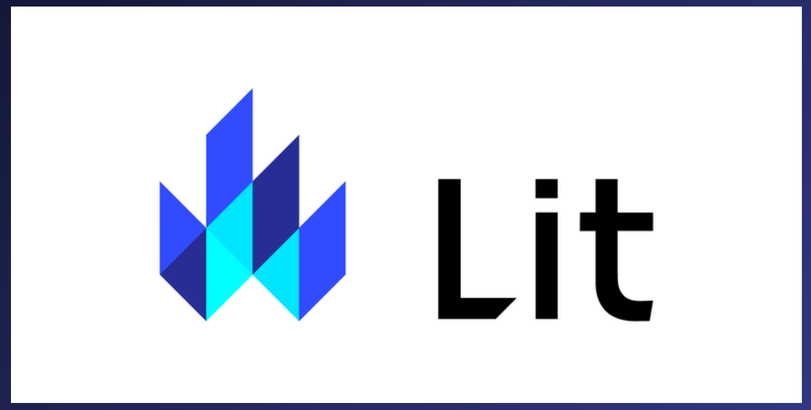
+

<acordion>



Mini-Aplicación Suscripciones

Lit



+

Standard Elements

<suscription>

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>
```

<suscription>

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).

<suscription>

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.

<suscription>

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(...).

<suscription>

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.

<suscription>

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.

<suscription>

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).

<suscription>

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.

<suscription>

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 <li> con íconos y textos.
- Gestiona el atributo booleano tax-included.

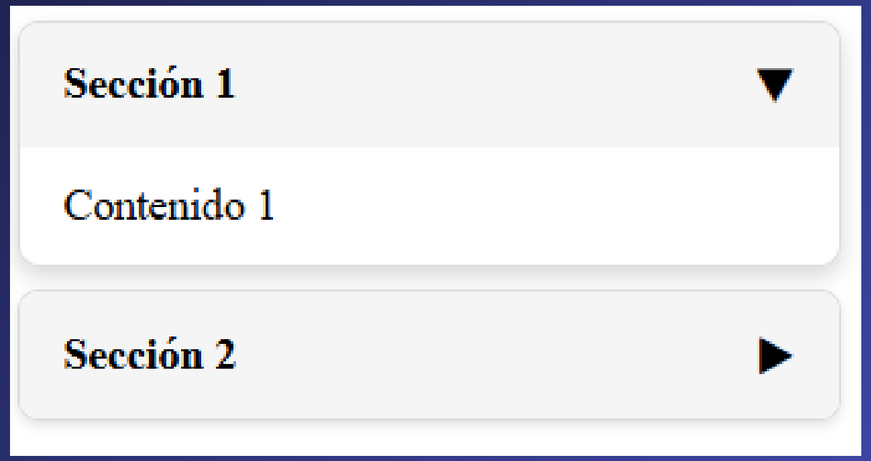
<suscription>

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 rgba(0,0,0,0.1);
          transition: box-shadow 0.3s ease;
          width: 100%;
          height: auto;
          max-width: 300px;

        }
```

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 acordion-item emite un evento toggle
- El evento viaja hasta acordion-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 acordion-list cierra todos los ítems y abre solo el clickeado:

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