Engineer

componentes web

🖼 funciones para definir elementos HTML nuevos y reutilizables

Qué es

Los componentes web, también llamados Web Components, son un conjunto de funciones del navegador que permite definir elementos HTML propios. Una página usa después ese elemento como cualquier otra etiqueta. Los estándares nombran cada parte por separado: los elementos personalizados y los elementos `template` y `slot` en HTML Standard, y los árboles de sombra en DOM Standard. El nombre colectivo titula una nota de un grupo de trabajo del W3C de 24 de julio de 2014, *Introduction to Web Components*, que el W3C dejó de mantener y marcó como documento que no debe citarse como referencia.

Un elemento personalizado se define mediante una clase JavaScript que extiende `HTMLElement` y se registra con un nombre mediante `customElements.define()`. La definición indica al analizador HTML cómo construirlo y cómo debe reaccionar a los cambios. Desde ese momento puede crearse a partir del marcado, con `document.createElement()` o con `new`. Su nombre empieza por una letra ASCII minúscula, no contiene letras ASCII mayúsculas y debe contener un guion. El guion separa los nombres personalizados del propio lenguaje: HTML Standard establece que HTML, SVG y MathML no incorporarán ningún elemento cuyo nombre contenga un guion.

Hay dos tipos. Un elemento personalizado autónomo, como el ejemplo del estándar `<flag-icon country="nl">`, parte de un elemento sin significado incorporado. Un elemento integrado personalizado extiende uno existente: una clase que extiende `HTMLButtonElement`, registrada con `{ extends: "button" }` y usada como `<button is="plastic-button">`, conserva todo lo que hace que un botón sea un botón, incluidos el comportamiento del foco, el envío de formularios y el atributo `disabled`.

Un árbol de sombra es un árbol de nodos separado, unido a un elemento llamado anfitrión; DOM Standard llama árbol ligero al árbol normal del anfitrión. `attachShadow()` crea una raíz de sombra en modo abierto o cerrado, y solo en el abierto la devuelve la propiedad `shadowRoot` del anfitrión. Un elemento `slot` del árbol de sombra marca dónde aparecen los hijos del anfitrión. Muestra los nodos que se le asignan o su propio contenido si no hay ninguno. Un slot con nombre recibe los hijos cuyo atributo `slot` lleva ese nombre. El elemento `template` contiene un fragmento HTML que no se representa y que un script puede clonar. Con un atributo `shadowrootmode`, un `template` representa una raíz de sombra declarativa, de modo que también puede escribirse un árbol de sombra directamente en el marcado.

Utilice el concepto en una decisión

Aplique «¿Qué son los componentes web?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Interfaces que cargan rápido, funcionan en el teléfono y se comportan igual en los navegadores que utilizan sus clientes.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Dirección del desarrollo integral de mapas GIS — cartografía integral, de la base de datos al navegador

Dirección del desarrollo integral de mapas GIS Póngase en contacto →

Por qué importa

Antes, una etiqueta inventada servía de poco: HTML Standard recuerda aquellos elementos como inválidos y de escasa utilidad práctica. Un elemento personalizado es un miembro pleno del documento. El analizador lo construye, los métodos del DOM lo crean y su clase decide qué ocurre cuando cambian sus atributos o entra o sale de una página.

La definición no tiene que llegar primero. Un elemento ya creado por el analizador se convierte en personalizado cuando se registra una definición para su nombre; HTML Standard lo llama actualización, o upgrading. Hasta entonces se trata como un elemento no definido, parecido a un `span`, y se muestran tal cual el texto y los enlaces que contiene. El estándar señala expresamente que estas actualizaciones permiten mejorar de forma progresiva el contenido de un elemento personalizado.

Un elemento personalizado también puede participar en formularios y en el árbol de accesibilidad. Una clase que establece `static formAssociated = true` y llama a `attachInternals()` obtiene un objeto `ElementInternals`. En él, `setFormValue()`

asigna al elemento un valor que su formulario envía como el de cualquier otro control. El mismo objeto establece un rol ARIA y estados ARIA predeterminados, aplicados salvo que el autor de la página los sustituya mediante `role` o los atributos `aria-*` correspondientes. También puede contener estados personalizados que las hojas de estilos seleccionan con la pseudoclase `:state()`.

Para un equipo, el árbol de sombra permite que el componente se baste a sí mismo: los selectores ordinarios de la página no cruzan a su árbol de sombra interno, y el componente decide qué partes expone al estilo exterior. Los valores heredados y las propiedades personalizadas sí pueden entrar, y el contenido del árbol ligero colocado en slots sigue sujeto a los selectores de la página. Cuando varias bibliotecas comparten una página, HTML Standard ofrece registros acotados, creados con `new CustomElementRegistry()`, para que sus elementos coexistan sin que las bibliotecas tengan que coordinar sus nombres.

Cómo se aplica en la práctica

Una tarjeta de producto muestra cómo trabajan juntas las piezas. La página conserva el contenido como marcado ordinario, con un atributo `slot` en el elemento destinado a un slot con nombre:

```
<product-card>
  <span slot="name">Oak stool</span>
  <a href="/shop/oak-stool/">See the details</a>
</product-card>
<script type="module" src="product-card.mjs"></script>
```

El módulo define el elemento. La plantilla se construye una vez y cada tarjeta recibe su propia copia en una raíz de sombra abierta:

```
const template = document.createElement("template");
template.innerHTML = `
  <h2 part="title"><slot name="name">Unnamed product</slot></h2>
  <slot></slot>
`;

class ProductCard extends HTMLElement {
  constructor() {
    super();
    const root = this.attachShadow({ mode: "open" });
    root.append(template.content.cloneNode(true));
  }
```

```
}

if (!customElements.get("product-card")) {
  customElements.define("product-card", ProductCard);
}
```

La página aplica estilo al título mediante la única parte que expone el componente:

```
product-card::part(title) {
  font-size: 1.25rem;
}
```

Antes de ejecutarse `product-card.mjs`, la tarjeta es un elemento no definido y el lector ve el nombre del producto y el enlace como contenido en línea normal. Cuando se ejecuta el módulo, `define()` registra la clase y se actualiza la tarjeta que ya estaba en la página. Su constructor añade una raíz de sombra y la llena con un clon del contenido de la plantilla. El `span` se asigna al slot llamado `name` y aparece dentro del encabezado. El enlace no tiene atributo `slot`, por lo que ocupa el slot predeterminado sin nombre. Una tarjeta sin nombre muestra en su lugar el contenido del propio slot, «Unnamed product» («producto sin nombre»). La comprobación `customElements.get()` evita que una segunda copia del módulo, llegada en otro paquete, haga fallar `define()`.

Los componentes compartidos solo ayudan si se usan, y el portafolio del sitio recoge un caso al respecto. El sistema de diseño «Liquid Glass» de iOS 26 parte de una interfaz en la que cada pantalla nueva reinventa botones, insignias y tarjetas, con pequeñas diferencias, hasta que cada cambio exige retocar cinco versiones hechas a medida. La respuesta descrita es un conjunto canónico de componentes a partir del que se construyen todas las pantallas. Entre esos componentes están Card, Label, Button, GlassIconButton y DirectoryGrid, con reglas de análisis que rechazan una insignia o un chip hechos a mano y remiten al componente canónico. La página no dice si son elementos personalizados, pero la disciplina se aplica igual: una definición por componente, impuesta mediante herramientas y no por la memoria del revisor. El registro de elementos personalizados impone la primera mitad al rechazar una segunda definición para un mismo nombre.

Dónde surgen los problemas

Un nombre con guion hace las veces de rol. «Un elemento personalizado autónomo no tiene ningún significado especial: representa a sus hijos», dice HTML Standard. Su ejemplo `taco-button` muestra el coste: para actuar como botón necesita `tabindex` , el rol ARIA `button` , un nombre accesible, `aria-disabled` , manejadores de teclas que conviertan Intro y Espacio en clics y estilos para el estado desactivado. Aun así, un elemento personalizado no puede declarar el comportamiento de foco predeterminado, por lo que `tabindex="0"` puede no coincidir con los botones de la plataforma. Cuando el elemento parte de uno existente, el estándar indica que extender uno integrado reduce el trabajo de su autor y de sus usuarios. Para uno autónomo, establezca el rol y los estados predeterminados mediante `ElementInternals` , en lugar de dejarlos a cada página que lo use.

El constructor hace todo el trabajo. HTML Standard impone reglas a los constructores de elementos personalizados: primero va `super()` sin argumentos; el constructor no debe consultar los atributos ni los hijos del elemento, porque uno creado en vez de actualizado no tiene ninguno; y el elemento no debe adquirir atributos ni hijos propios. Incumplir esas reglas puede producir un elemento que el analizador o los métodos DOM no logren crear. Reserve el constructor para el estado inicial, los valores predeterminados, los receptores de eventos y la raíz de sombra, y traslade la obtención de datos y la representación a `connectedCallback` .

Se espera que `connectedCallback` **se ejecute una sola vez.** Puede ejecutarse varias. De forma predeterminada, mover un elemento ejecuta `disconnectedCallback` y después `connectedCallback` , de modo que el elemento reinicia su estado como si se hubiera retirado e insertado de nuevo. Proteja con un indicador la preparación que solo debe ocurrir una vez y deshaga en `disconnectedCallback` lo iniciado por `connectedCallback` . Si debe conservar su estado al moverse con `moveBefore()` , defina `connectedMoveCallback` : su mera presencia sustituye a la pareja predeterminada, incluso con el cuerpo vacío.

Un nombre se define dos veces. `customElements.define()` lanza `NotSupportedError` si ya están registrados el nombre o la clase, y `SyntaxError` si el nombre no es válido, por ejemplo porque no tiene guion o contiene una mayúscula. Compruebe antes con `customElements.get()` y dé un registro acotado propio a una biblioteca que deba coexistir con otras.

El modo cerrado se trata como protección. El modo cerrado oculta la raíz a la propiedad `shadowRoot` del anfitrión y oculta los nodos internos de la ruta compuesta de un evento tal como la ven los receptores externos a ese árbol de sombra cerrado. CSS Shadow Parts afirma claramente que los árboles de sombra no se concibieron como una barrera de seguridad, sino como una comodidad para los autores de páginas. Mantenga los secretos fuera de los componentes y elija el modo cerrado por su efecto sobre el código exterior, no como cerradura.

Se llama a `attachShadow()` donde no puede funcionar. DOM Standard solo permite una raíz de sombra en elementos con un nombre válido de elemento personalizado y en una lista breve de elementos integrados: entre ellos `div`, `section`, `p` y `span`, pero no `button`. En un botón integrado personalizado, la llamada lanza `NotSupportedError`. También cambia su comportamiento si el elemento ya tiene una raíz: si es declarativa y del mismo modo, la llamada elimina todo su contenido y la devuelve vacía; en los demás casos lanza una excepción. Compruebe primero si existe una raíz de sombra para conservar, en vez de borrar, el marcado que el servidor envió en una raíz declarativa.

Se espera que los estilos de la página lleguen al interior. CSS Shadow Parts llama al límite barrera de estilo y señala que también dificulta personalizar el componente cuando la página lo necesita. Exponga con `part` los elementos a los que la página pueda dar estilo y selecciónelos desde fuera con `::part()`. Un componente anidado en otro necesita `exportparts` en su anfitrión interior para transmitir sus partes a la página. Para colores y espaciados que deban seguirla, lea las propiedades personalizadas definidas en el anfitrión, que alcanzan el árbol de sombra por herencia.

El contenido solo existe después del script. Un componente que escribe todo su texto con JavaScript no muestra nada a los lectores hasta cargar la definición, y nada en absoluto si el script falla. Conserve el contenido en el árbol ligero y asígnelo a slots, como hace la tarjeta de producto, para que la actualización mejore lo que ya existe. Si el contenido debe esperar, el ejemplo del propio HTML Standard lo oculta hasta que `customElements.whenDefined()` se resuelve para cada elemento que coincide con `:not(:defined)`.

Lecturas adicionales

- HTML Standard: Custom elements — reglas de nombres, requisitos de constructores, actualizaciones, elementos asociados a formularios y `ElementInternals`, con ejemplos completos.

- DOM Standard: Shadow tree — raíces de sombra y sus anfitriones, slots y nodos asignados, y los pasos de `attachShadow()`.
- HTML Standard: Scripting — el elemento `template` con su atributo `shadowrootmode` y el elemento `slot`.
- CSS Shadow Parts — el borrador de trabajo del W3C de 16 de diciembre de 2025 que define `part`, `exportparts` y el pseudoelemento `::part()`.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué son los componentes web?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/web-components/