


Explicación de Códigos HTML y JavaScript


PROGRAMACIÓN INTEGRATIVA DE COMPONENTES
WEB

Código 11

 [11template]


- ▶ HTML:
 - ▶ - Se define una estructura básica HTML con un título y un template vacío.
 - ▶ - El componente personalizado ``elemento-template`` se inserta en el body.
 - ▶ - Se carga el script ``my-element.js``.
- ▶ JavaScript:
 - ▶ - Define un componente ``myElement``.
 - ▶ - El método ``getTemplate()`` crea un template con contenido HTML.
 - ▶ - El método ``getStyles()`` devuelve estilos específicos para el componente.
 - ▶ - El método ``render()`` añade el contenido del template al componente.
 - ▶ - El método ``connectedCallback()`` se llama cuando el componente es añadido al DOM, llamando al método ``render()``.
 - ▶ - Se define el componente ``elemento-template``.

Código 12

 [12sd]

- ▶ HTML:
 - ▶ - Se define una estructura HTML similar a la del código 1, pero se carga el componente ``my-element``.
- ▶ JavaScript:
 - ▶ - Similar al código 1, pero este componente usa shadow DOM (``this.attachShadow({ mode: "open" })``).
 - ▶ - El shadow DOM encapsula el estilo y el marcado del componente, evitando colisiones de estilos.

Código 15

 [15slot]

- ▶ HTML:
 - ▶ - Se define un componente ``my-element`` con slots para personalizar el contenido.
- ▶ JavaScript:
 - ▶ - Define slots dentro del template para permitir la inserción de contenido personalizado.
 - ▶ - Los métodos son similares a los del código 2, pero con slots.


Código 16



[16slotted]


- ▶ HTML:
 - ▶ - Similar al código 3, pero se añaden clases y estilos a los slots.
- ▶ JavaScript:
 - ▶ - Utiliza `::slotted` para aplicar estilos a los elementos insertados en los slots.
 - ▶ - Los métodos son similares a los del código 3.

Código 5

 [17attribute]


- ▶ HTML:
 - ▶ - Se define un componente `my-element` con atributos para personalizar el contenido.
- ▶ JavaScript:
 - ▶ - Obtiene los atributos del componente en el constructor.
 - ▶ - Los métodos son similares a los códigos anteriores, pero los atributos se utilizan para definir el contenido del componente.

Código 18

 [18attributeChangedCallback]

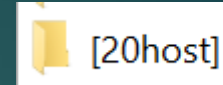
- ▶ HTML:
 - ▶ - Similar al código 5.
- ▶ JavaScript:
 - ▶ - Implementa ``observedAttributes`` y ``attributeChangedCallback`` para manejar cambios en los atributos del componente.
 - ▶ - Similar al código 5, pero permite la actualización dinámica del contenido basado en cambios de atributos.

Código 19

 [19disconnectedCallback]


- ▶ HTML:
 - ▶ - Se define un componente ``my-custome-element`` y se remueve del DOM.
- ▶ JavaScript:
 - ▶ - Define ``connectedCallback`` y ``disconnectedCallback`` para manejar la adición y eliminación del componente del DOM.
 - ▶ - ``connectedCallback`` se llama cuando el componente es añadido al DOM.
 - ▶ - ``disconnectedCallback`` se llama cuando el componente es removido del DOM.

Código 20



- ▶ HTML:
 - ▶ - Se define el componente ``my-element`` con varios estilos y slots.
- ▶ JavaScript:
 - ▶ - Utiliza ``:host`` para aplicar estilos al componente mismo.
 - ▶ - ``:host-context`` para aplicar estilos basados en el contexto del componente.
 - ▶ - Los métodos son similares a los códigos anteriores, con un enfoque en estilos del host.

Código 21

 [21CustomProperties]

- ▶ HTML:
 - ▶ - Define el componente ``my-element`` con propiedades CSS personalizadas.
- ▶ JavaScript:
 - ▶ - Utiliza propiedades CSS personalizadas (``--primary-color``, etc.) para estilos.
 - ▶ - Los métodos son similares a los códigos anteriores, pero con enfoque en propiedades CSS personalizadas.

Código 25

- ▶ HTML:
 - ▶ - Define un componente ``product-card`` con atributos para una tarjeta de producto.
- ▶ JavaScript:
 - ▶ - Similar al código 6, pero con más atributos y un diseño de tarjeta de producto.
 - ▶ - Define ``observedAttributes`` y ``attributeChangedCallback`` para manejar cambios en los atributos.