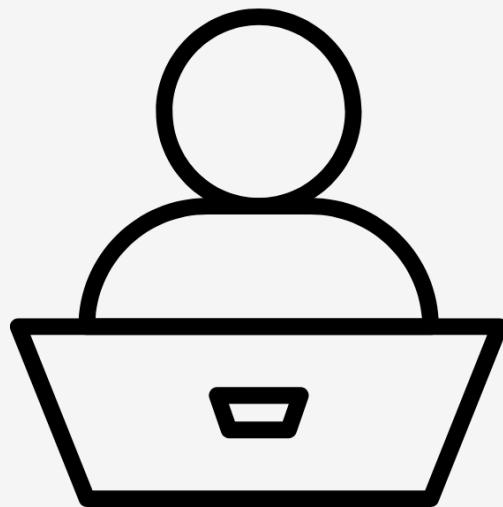


diccionario

HTML

maria jose angulo



Estructura de Documentos

Estructura de Documentos

<!DOCTYPE>: Versión de (X)HTML

<html>: Documento HTML

<head>: Información de la página

<body>: Contenidos de la página

Listas

: Lista ordenada

: Lista desordenada

: Elemento de lista

<dl>: Lista de definiciones

☆<dt>: Término de definición

☆<dd>: Descripción del término

Marcado de Texto

<h[1-6]>: Encabezado (h1 a h6)

****: Énfasis fuerte

<div>: Sección de la página

****: Énfasis

<blockquote>: Cita larga

****: Sección en línea

<a>: Enlace

<p>: Párrafo

<abbr>: Abreviatura

**
**: Salto de línea

<acronym>: Acrónimo

<address>: Dirección

<hr />: Regla horizontal

<pre>: Texto preformatado

<dfn>: Definición

<code>: Código

<cite>: Cita

****: Texto eliminado

<ins>: Texto insertado

<sub>: Subíndice

<sup>: Superíndice

<bdo>: Dirección del texto

Formularios

<form>: Formulario

<fieldset>: Colección de
campos

<label>: Etiqueta de
entrada

<input>: Entrada de
formulario

<select>: Caja de selección

<textarea>: Entrada de
texto grande

<button>: Botón

<optgroup>: Grupo de
opciones

<option>: Opciones de
selección

Eventos de Formularios

Eventos de Formulario

onBlur: Cuando el elemento pierde el foco

onChange: Cuando el valor del elemento cambia

onFocus: Cuando el elemento recibe el foco

onReset: Cuando el formulario se reinicia

onSelect: Cuando se selecciona un texto

★ **onSubmit:** Cuando se envía el formulario

Tablas

<table>: Tabla

<caption>: Título de la tabla

<thead>: Encabezado de la tabla

<tbody>: Cuerpo de la tabla

◦ **<tfoot>:** Pie de la tabla

<colgroup>: Grupo de columnas

◦ **<col />:** Columna

◦ **<tr>:** Fila de la tabla

◦ **<th>:** Celda de encabezado

◦ **<td>:** Celda de tabla

EXPLICACIÓN DEL Diccionario de HTML

HTML (Hypertext Markup Language): Lenguaje de marcado utilizado para estructurar contenido en la web.

Etiqueta: Elemento en HTML que define una parte del contenido. Se compone de una apertura y una posible cierre (ej.: <p> y </p>).

Atributo: Información adicional que se puede agregar a una etiqueta para describirla o modificar su comportamiento (ej.:).

Elemento: Combinación de una etiqueta de apertura, su contenido y una etiqueta de cierre (ej.: <p>Este es un párrafo.</p>).

Etiquetas Comunes

<html>: Define el inicio de un documento HTML.

<head>: Contiene información sobre el documento, como su título y enlaces a recursos.

<title>: Especifica el título del documento que aparece en la pestaña del navegador.

<body>: Contiene el contenido principal de la página web.

<h1> a <h6>: Encabezados de diferentes niveles, donde <h1> es el más importante y <h6> el menos importante.

<p>: Define un párrafo.

<a>: Crea un enlace a otra página web o recurso.

: Inserta una imagen en la página.

: Crea una lista desordenada (con viñetas).

: Crea una lista ordenada (numerada).

: Define un elemento de lista, que puede ser parte de una lista ordenada o desordenada.

<table>: Crea una tabla.

<tr>: Define una fila en una tabla.

<th>: Define una celda de encabezado en una tabla.

<td>: Define una celda de datos en una tabla.

<form>: Crea un formulario interactivo.

<input>: Define un campo de entrada en un formulario.

<button>: Crea un botón interactivo.

<label>: Define una etiqueta para un elemento de formulario.

<select>: Crea un menú desplegable.

<option>: Define un elemento dentro de un menú desplegable.

Conceptos de Estilo

CSS (Cascading Style Sheets): Lenguaje utilizado para estilizar documentos HTML. Permite controlar la presentación visual y el diseño.

Conceptos de Script

JavaScript: Lenguaje de programación que se usa comúnmente en la web para añadir interactividad a las páginas.

Otros Términos

DOCTYPE: Declaración que define el tipo de documento y la versión de HTML que se está utilizando.

Viewport: Área visible de una página web donde se puede ver el contenido; importante para el diseño responsivo.

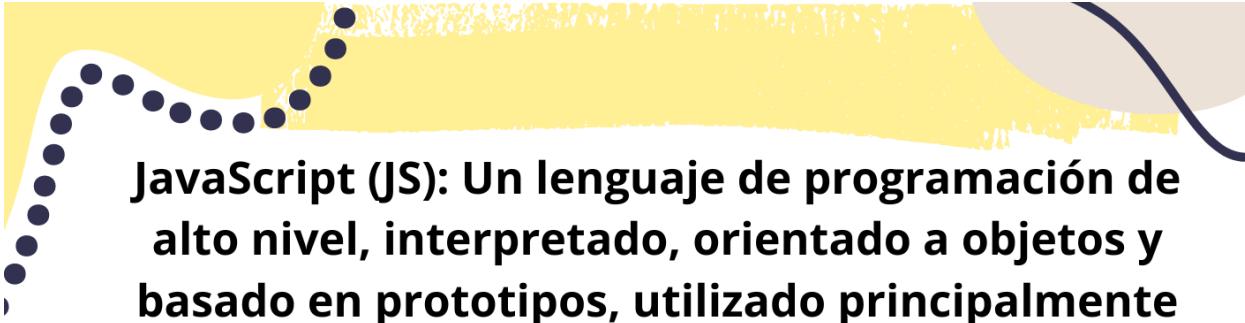
Metaetiquetas: Etiquetas dentro de <head> que proporcionan información sobre el documento (ej.: <meta charset="UTF-8">).

Este diccionario proporciona una base para entender los conceptos y etiquetas más comunes en HTML.

DICCIONARIO

JS

angulo maria jose



JavaScript (JS): Un lenguaje de programación de alto nivel, interpretado, orientado a objetos y basado en prototipos, utilizado principalmente para crear contenido interactivo en la web.

Variable: Un contenedor que almacena datos que pueden cambiar durante la ejecución de un programa. Se puede declarar usando var, let o const.

Función: Un bloque de código que realiza una tarea específica y puede ser llamado varias veces. Se declara usando la palabra clave function, o como una expresión de función.

Objeto: Una colección de propiedades y métodos. Se crea utilizando llaves {}. Por ejemplo: const persona = { nombre: "Juan", edad: 30 };

Array (Arreglo): Una lista ordenada de elementos que puede contener múltiples valores, se define con corchetes []. Por ejemplo: const frutas = ["manzana", "banana", "cereza"];



Evento: Una acción o suceso que ocurre en el navegador, como un clic, un movimiento del ratón o la carga de una página. Se pueden manejar con "event listeners".

DOM (Document Object Model): Una interfaz de programación para documentos HTML y XML que representa la estructura del documento como un árbol de nodos.

API (Application Programming Interface): Conjunto de definiciones y protocolos para construir y interactuar con software y aplicaciones. Las APIs pueden ser usadas para acceder a funcionalidades del navegador o servicios externos.

Tipos de Datos

Primitivos: Tipos de datos que no son objetos y tienen un valor inmutable.

Incluyen:

String: Cadena de caracteres. Ejemplo:
"Hola, mundo!".

Number: Números, pueden ser enteros o flotantes. Ejemplo: 42, 3.14.

Boolean: Verdadero o falso. Ejemplo:
true, false.

Null: Un valor intencionalmente vacío.
Ejemplo: let valor = null;.

Undefined: Un valor que no ha sido asignado. Ejemplo: let variable;.

Symbol: Un tipo de dato único e inmutable introducido en ES6.

Objetos: Una estructura de datos que puede contener múltiples pares de clave-valor.

Estructuras de Control

Condicionales: Permiten ejecutar diferentes bloques de código basados en condiciones.

Ejemplo: if, else, switch.

Bucles (Loops): Permiten repetir un bloque de código varias veces. Ejemplo: for, while, do...while.

Métodos y Funciones Comunes

console.log(): Muestra información en la consola del navegador.

parseInt(): Convierte una cadena en un número entero.

parseFloat(): Convierte una cadena en un número de punto flotante.

Math.random(): Genera un número aleatorio entre 0 y 1.

Array.prototype.forEach(): Ejecuta una función en cada elemento de un arreglo.

Promesas y Asincronía

Promesa: Un objeto que representa la eventual finalización (o fracaso) de una operación asíncrona y su valor resultante.

Async/Await: Una forma de manejar promesas que permite escribir código asíncrono que se asemeja a un código síncrono.

Diccionario de JavaScript

Otros Términos

JSON (JavaScript Object Notation): Un formato de intercambio de datos ligero, fácil de leer y escribir para humanos y fácil de parsear y generar para las máquinas.

Closure: Una función que retiene su contexto léxico, incluso cuando se ejecuta en un contexto diferente.

Scope (Ámbito): Se refiere a la accesibilidad de variables y funciones en diferentes partes del código. Hay ámbitos global y local.

Este diccionario proporciona una visión general de algunos de los términos y conceptos más comunes en JavaScript.

1. ¿Qué es el DOM?

El DOM es una representación estructurada del documento HTML como una jerarquía de nodos. Cada elemento HTML se convierte en un "nodo" que puede ser manipulado a través de JavaScript.

2. Selección de Elementos del DOM

getElementById: Selecciona un elemento por su ID.

```
const elemento = document.getElementById('miID');
```

getElementsByClassName: Selecciona todos los elementos que tienen una clase específica.

```
const elementos = document.getElementsByClassName('miClase');
```

getElementsByTagName: Selecciona todos los elementos con una etiqueta específica.

```
const elementos = document.getElementsByTagName('div');
```

querySelector: Selecciona el primer elemento que coincide con un selector CSS.

```
const elemento = document.querySelector('.miClase');
```

querySelectorAll: Selecciona todos los elementos que coinciden con un selector CSS.

```
const elementos = document.querySelectorAll('p.miClase');
```

3. Manipulación de Elementos

Cambiar el contenido:

```
elemento.innerHTML = 'Nuevo contenido';
```

Cambiar estilos:

```
elemento.style.color = 'red';
```

```
elemento.style.display = 'none'; // Oculta el elemento
```

Agregar o quitar clases:

```
elemento.classList.add('nueva-clase');
elemento.classList.remove('clase-para-quitar');
```

4. Creación y eliminación de elementos

Crear un nuevo elemento:

```
const nuevoElemento = document.createElement('div');
nuevoElemento.innerHTML = 'Soy un nuevo div';
document.body.appendChild(nuevoElemento); // Agrega el nuevo elemento al final del body
```

Eliminar un elemento:

```
const elementoParaEliminar = document.getElementById('miID');
elementoParaEliminar.parentNode.removeChild(elementoParaEliminar);
```

5. Eventos

El manejo de eventos permite interactuar con los usuarios:

Agregar un evento:

```
elemento.addEventListener('click', function() {
    alert('Elemento clickeado');
});
```

6. Consideraciones de rendimiento

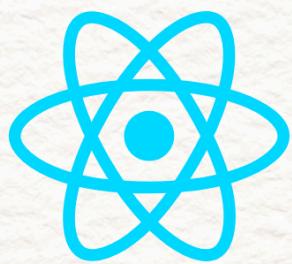
Al manipular el DOM, es importante considerar el rendimiento. Las operaciones de reflow y repaint pueden ser costosas, así que a menudo se recomienda:

Guardar las modificaciones en una variable y luego aplicar todos los cambios de una vez.
Utilizar fragmentos de documentos para agregar múltiples nodos al DOM de forma más eficiente.

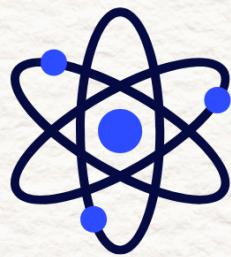
Resumen

El manejo del DOM es esencial para crear aplicaciones web interactivas y dinámicas. Usar JavaScript para seleccionar, manipular y eventos te permitirá mejorar la experiencia del usuario en tu sitio web.

DICCIONARIO



React



*Angulo
Maria Jose*

1. Componentes: Son las unidades básicas de cualquier aplicación de React. Un componente puede ser una clase o una función y puede manejar su propio estado y propiedades.
2. JSX: Es una sintaxis que combina JavaScript con HTML. Permite escribir elementos de React de una manera que se parece a HTML, pero está basado en JavaScript.
3. Estado (State): Es un objeto que determina cómo se renderiza un componente. Cada vez que el estado cambia, el componente se vuelve a renderizar.
4. Props: Son las propiedades que se pasan a un componente desde su parente. Las props permiten la comunicación entre componentes y son de solo lectura.
5. Ciclo de vida (Lifecycle): Son una serie de métodos que son llamados en diferentes etapas de la vida de un componente, como cuando se monta, actualiza y desmonta.
6. Hooks: Funciones que permiten usar el estado y otras características de React en componentes funcionales. Ejemplos son `useState`, `useEffect`, y `useContext`.
7. Contexto (Context): Proporciona una forma de pasar datos a través del árbol de componentes sin tener que pasar props manualmente en cada nivel.
8. Redux: Una biblioteca para la gestión del estado en aplicaciones de JavaScript, que se utiliza comúnmente con React. Permite gestionar el estado global de la aplicación de manera predecible.
9. Router (React Router): Una biblioteca que permite manejar la navegación entre diferentes componentes y rutas en una aplicación de React.
10. Virtual DOM: Una representación en memoria del DOM real. React utiliza el Virtual DOM para optimizar las actualizaciones de la interfaz de usuario, haciendo solo los cambios necesarios.
11. Render Props: Es una técnica de React para compartir código entre componentes mediante una función que retorna un elemento React.
12. Higher-Order Components (HOC): Son funciones que toman un componente y retornan un nuevo componente. Se utilizan para reutilizar lógica de componentes.

Diccionario de HTML con React

1. Elementos HTML

`<div>`: Contenedor genérico utilizado para agrupar otros elementos.

``: Contenedor en línea, usado para contenido pequeño o estilo aplicado a una parte del texto.

`<h1>` a `<h6>`: Encabezados, de mayor a menor importancia.

`<p>`: Define un párrafo de texto.

`` y ``: Listas desordenadas y ordenadas, respectivamente.

``: Elemento de lista.

`<form>`: Contenedor para elementos de formularios.

`<input>`: Permite al usuario ingresar datos. Existen diferentes tipos (texto, contraseña, checkbox, etc.).

`<button>`: Botón que puede ser clickeado por el usuario.

``: Inserta imágenes en el documento. Usa el atributo `src` para especificar la fuente.

`<a>`: Define un enlace a otra página o recurso, usando el atributo `href`.

2. Atributos Comunes

className: En React, se utiliza className en lugar de class para asignar clases CSS a los elementos. Ejemplo: <div className="mi-clase">.

style: Permite aplicar estilos en línea utilizando un objeto JavaScript en React. Ejemplo: <div style={{ color: 'red' }}>Texto rojo</div>.

id: Asigna un identificador único a un elemento, se usa como en HTML.

onClick: Maneja eventos de clic en elementos. Ejemplo: <button onClick={manejarClick}>Clic aquí</button>.

3. Componentes

Componente Funcional: Una función de JavaScript que devuelve un elemento React. Ejemplo:

```
function MiComponente() {
  return <h1>Hola, mundo!</h1>;
}
```

Componente de Clase: Una clase de JavaScript que extiende React.Component. Ejemplo:

```
class MiComponente extends React.Component {
  render() {
    return <h1>Hola, mundo!</h1>;
  }
}
```

4. Props (Propiedades)

Props: Son utilizadas para pasar datos de un componente padre a un componente hijo. Se acceden a través de this.props en componentes de clase o simplemente como un argumento en componentes funcionales. Ejemplo:

```
function Saludo(props) {
  return <h1>Hola, {props.nombre}!</h1>;
}
```

5. Estado (State)

State: Los datos que cambian en un componente. Se inicializa en el constructor para componentes de clase o mediante el hook useState en componentes funcionales. Ejemplo:

```
const [contador, setContador] = useState(0);
```

6. Hooks

useState: Hook que permite agregar estado a componentes funcionales.

useEffect: Hook que permite realizar efectos secundarios en componentes funcionales, como suscripciones o llamadas a APIs.

7. Renderizado Condicional

Utiliza expresiones JavaScript para realizar renderizado condicional, por ejemplo:

```
{isLoggedIn ? <LogoutButton /> : <LoginButton />}
```

8. Listas y Claves

Al renderizar listas de elementos, es importante utilizar una propiedad key para ayudar a React a identificar qué elementos han cambiado. Ejemplo:

```
const items = ["Item 1", "Item 2", "Item 3"];
const lista = items.map((item, index) => <li key={index}>{item}</li>);
```

9. Formularios

Manejo de formularios en React, donde cada campo se puede controlar mediante el estado.

Ejemplo:

```
function MiFormulario() {
  const [nombre, setNombre] = useState("");
  const manejarCambio = (event) => {
    setNombre(event.target.value);
  };

  return (
    <form>
      <input type="text" value={nombre} onChange={manejarCambio} />
    </form>
  );
}
```

10. JSX

JSX: Una extensión de sintaxis para JavaScript que permite escribir elementos de HTML dentro de JavaScript. Ejemplo:

```
const elemento = <h1>Hola, mundo!</h1>;
```

Conclusión

Este diccionario te proporciona un resumen de los conceptos clave de HTML en el contexto de React.

Diccionario de Términos de React

1. Componentes

Componente Funcional: Una función que devuelve un elemento React. Se utilizan generalmente para componentes que no necesitan manejar estado local.

Componente de Clase: Una clase que extiende React.Component, permitiendo manejar estado y métodos de ciclo de vida.

2. JSX

JSX (JavaScript XML): Una sintaxis que permite escribir HTML dentro de JavaScript. Es transformado por Babel a llamadas a React.createElement().

3. Props (Propiedades)

Props: Son objetos que se pasan a los componentes como argumentos. Permiten la comunicación entre componentes.

4. Estado (State)

State: Un objeto que maneja los datos internos de un componente. El estado puede cambiar con el tiempo y puede afectar el renderizado del componente.

5. Hooks

useState: Hook que permite agregar estado a un componente funcional.

useEffect: Hook que se utiliza para manejar efectos secundarios, como suscripciones a eventos o llamadas a APIs.

useContext: Hook que permite acceder a datos del contexto en un componente funcional.

6. Renderizado Condicional

Renderizado Condicional: Práctica de mostrar diferentes elementos o componentes basados en condiciones específicas.

7. Listas y Claves

Renderizado de Listas: La forma de renderizar un conjunto de elementos mediante el método `map()`.

Key: Propiedad única asignada a cada elemento de una lista para ayudar a React a identificar qué elementos han cambiado.

8. Ciclo de Vida

Ciclo de Vida del Componente: Conjunto de métodos en componentes de clase que permiten ejecutar código en fases específicas del ciclo de vida de un componente (montaje, actualización, desmontaje).

9. Contexto

Contexto: Una forma de pasar datos a través del árbol de componentes sin tener que pasar `props` manualmente en cada nivel.

10. Routing

React Router: Biblioteca que permite la navegación y el manejo de rutas en aplicaciones React, facilitando la creación de SPA (Single Page Applications).

11. Prop Types

PropTypes: Una forma de validar las `props` pasadas a un componente, asegurando que los datos de entrada tengan el tipo correcto.

12. Higher-Order Components (HOC)

HOC: Una función que toma un componente y devuelve uno nuevo, permitiendo reutilizar lógica de componentes.

13. Render Prop

Render Prop: Un patrón que permite a un componente compartir su código de renderizado utilizando una función que devuelve un elemento JSX.

14. Framer Motion

Framer Motion: Una biblioteca para animaciones en aplicaciones React que proporciona una API sencilla para animaciones complejas.

15. Async/Await

Async/Await: Sintaxis utilizada para manejar operaciones asíncronas, como llamadas a API, de una manera más legible y sencilla en comparación con las promesas.

Conclusión

Este diccionario proporciona una visión general de los términos clave y conceptos que se utilizan en React.