JavaScript: Pila de Ejecución y Bucle de Eventos

1. Pila de Ejecución en JavaScript

La pila de ejecución en JavaScript (call stack) funciona como una pila de platos: el último en entrar es el primero en salir (FILO). Cuando una función se llama, se coloca en la cima de la pila. Cuando termina, se elimina y el control vuelve a la función anterior.

Ejemplo básico con dos funciones:

```
function prueba1() {
  console.log("Estoy dentro de prueba1");
  return "Fin prueba1";
}

function prueba2() {
  console.log("Inicio de prueba2");
  const resultado = prueba1();
  console.log("Volvemos a prueba2 con:", resultado);
  return "Fin prueba2";
}

const temp = prueba2();
console.log("De vuelta en el contexto global con:", temp);
```

Paso a paso:

1. Se ejecuta prueba2 desde el contexto global.
2. Dentro de prueba2 se llama a prueba1, que entra en la pila.
3. Cuando prueba1 termina, se elimina de la pila.
4. Regresamos a prueba2, que también termina.
5. Volvemos al contexto global.

Ejemplo con función recursiva:

```
function crearNuevoContexto(n) {
  console.log("Entrando en contexto con n =", n);
  if (n === 0) {
    console.log("¡n es 0, detenemos aquí!");
```

```
    return;
  }
  crearNuevoContexto(n - 1);
  console.log("Saliendo del contexto con n =", n);
}

crearNuevoContexto(3);
```

2. Bucle de Eventos y Cola de Mensajes

JavaScript también maneja código asíncrono mediante la cola de mensajes (FIFO: First In, First Out). Cuando ocurre un evento o se resuelve un temporizador, la función asociada se coloca en la cola. El event loop revisa si la pila está vacía y entonces ejecuta lo que haya en la cola.

Ejemplo con event listener:

```
<button id="miBoton">Haz clic</button>

<script>
  function clickHandler() {
    console.log("Clic en el botón");
  }

  document.getElementById("miBoton").addEventListener("click", clickHandler);

  console.log("Primer log");
  console.log("Segundo log");
</script>
```

El flujo:

1. Se ejecutan los console.log de forma inmediata.
2. Si el usuario hace clic, clickHandler va a la cola.
3. Cuando la pila está vacía, se ejecuta clickHandler.

Ejemplo con setTimeout:

```
console.log("Primer log");
```

JavaScript: Pila de Ejecución y Bucle de Eventos

```
setTimeout(() => {
  console.log("Segundo log");
}, 0);

console.log("Tercer log");
```

Aunque el temporizador es 0, el segundo log se retrasa hasta que la pila se vacíe.

Problema común: bloqueo de la pila impide ejecutar eventos:

```
function tareaLarga() {
  let ahora = Date.now();
  while (Date.now() - ahora < 5000) {}
  console.log("Fin de tarea larga");
}

function clickHandler() {
  console.log("¡Clic recibido!");
}

document.getElementById("miBoton").addEventListener("click", clickHandler);

tareaLarga(); // Bloquea la pila
```

El clickHandler no puede ejecutarse hasta que tareaLarga termine. Esto bloquea la interacción del usuario y da una mala experiencia.

Resumen visual:

- Pila de ejecución: ejecuta código línea a línea.
- Cola de mensajes: almacena funciones asíncronas.
- Event loop: mueve tareas de la cola a la pila cuando esta está vacía.