~~~
# 📜 JavaScript Advanced HOFs, Callbacks, and Closures


## **JavaScript Challenges on HOFs, Callbacks, and Closures**

<details>
  <summary><strong>Create a function that takes another function as an
argument and calls it after 3 seconds (HOF +
Callback).</strong></summary>

  ```js
  function delayedExecution(callback) {
      setTimeout(callback, 3000);
  }

  // Example usage
  delayedExecution(() => console.log("Executed after 3 seconds"));
  ```
</details>

<details>
  <summary><strong>Implement your own version of `.map()` as a
higher-order function.</strong></summary>

  ```js
  function customMap(array, callback) {
      let result = [];
      for (let i = 0; i < array.length; i++) {
          result.push(callback(array[i], i, array)); // Apply callback
to each element
      }
      return result;
  }

  // Example usage
  console.log(customMap([1, 2, 3], num => num * 2));
  // Output: [2, 4, 6]
  ```
</details>

<details>
  <summary><strong>Write a function that uses closures to create a
counter.</strong></summary>

  ```js
  function createCounter() {
      let count = 0;
      return function() { // Closure retains access to `count`
          return ++count;
      };
  }
~~~

~~~
  // Example usage
  const counter = createCounter();
  console.log(counter()); // Output: 1
  console.log(counter()); // Output: 2
  console.log(counter()); // Output: 3
  ```
</details>

<details>
  <summary><strong>Implement a function that limits how many times
another function can be called (Closure + HOF).</strong></summary>

  ```js
  function limit(fn, limit) {
      let calledtimes = 0;
      return function () {
          if (calledtimes < limit) {
              calledtimes++;
              fn();
          }
      };
  }

  // Example usage
  let fn = limit(() => console.log("hello"), 3);
  fn(); // "hello"
  fn(); // "hello"
  fn(); // "hello"
  fn(); // (No output, limit reached)
  ```
</details>

---
~~~