**Title:** Test Plan for Object Operations in JavaScript

**By whom:** Van Minh Le

**Date:** 23.03.2025

**Version:** 1.0

**1. Introduction**

This test plan defines the strategy, process, and workflow for testing JavaScript object operations. The goal is to verify that object manipulations (creation, updating, and deletion of properties) function correctly in a browser environment.

**1.1. Scope**

**1.1.1. In Scope**

The following functionalities will be tested:

* Creating an object myMovie with specific properties.
* Adding new properties (rating, summary).
* Updating existing properties (rating, year).
* Removing a property (summary).
* Verifying changes using JavaScript code execution in a browser.

**1.1.2. Out of Scope**

* Performance testing of object operations.
* Complex object structures or nested properties.
* UI-based verification.

**1.2. Quality Objective**

The objective of this test is to ensure:

* The object myMovie is created correctly with initial properties.
* New properties are successfully added.
* Updates to properties are applied correctly.
* The property removal is correctly executed.
* All test cases pass before submission.

**1.3. Roles and Responsibilities**

|  |  |
| --- | --- |
| **Role** | **Responsibility** |
| Developer | Implements functions and ensures correctness. |
| Tester | Executes test cases and records results. |
| Reviewer | Reviews test results and verifies code quality. |

**2. Test Methodology**

**2.1. Test Levels**

* **Unit Testing**: Each object operation is tested individually.
* **Integration Testing**: Ensure object manipulations work together correctly.
* **System Testing**: Verify all functionalities meet the requirement.

**2.2. Suspension Criteria and Resumption Requirements**

* Testing is suspended if critical bugs block further execution.
* Testing resumes once the issues are fixed and verified.

**2.3. Test Completeness**

The testing is considered complete when:

* 100% of the required functionalities are tested.
* All test cases pass without critical defects.
* Code meets JavaScript coding guidelines.

**3. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Status** |
| TC-001 | Create object myMovie with initial properties | { Title: "The longest day", Year: 1964 } | **title: The longest day year: 1964** | Pass |
| TC-002 | Add rating and summary properties | { rating: 5, summary: "World War II movie about the Normandy landings" } | **title: The longest day year: 1964 rating: 5 summary: World War II movie about the Normandy landings** | Pass |
| TC-003 | Update rating to 4 and year to 1962 | { rating: 4, year: 1962 } | **title: The longest day year: 1962 rating: 4 summary: World War II movie about the Normandy landings** | Pass |
| TC-004 | Remove summary property |  | **title: The longest day year: 1962 rating: 4** | Pass |

**4. Resource & Environment Needs**

**4.1. Testing Tools**

* Code Editor: VS Code
* Debugging: Chrome Developer Tools
* Version Control: GitHub

**4.2. Test Environment**

* OS: Windows
* Browser: Chrome (latest version)
* JavaScript: ECMAScript 6+

**5. Terms/Acronyms**

|  |  |
| --- | --- |
| **TERM/ACRONYM** | **DEFINITION** |
| API | Application Program Interface |
| AUT | Application Under Test |
| JS | JavaScript |

This test plan ensures that all object manipulations work correctly before submission. All test cases will be executed, and failures will be debugged and fixed accordingly.