1. Codes and Key Data

A. Script Organization

- Scripts should be clearly structured, ideally categorized into sections.
    - Q1: "I organize my scripts by function, keeping related methods together."
    - Q3: "Use slashes // to categorize in sections for codes."
    - Q2: "Marking every little thing clutters the script."
- Scripters prefer some level of modularity but differ in how strict this should be.
    - Q1: "I try to keep things modular, but sometimes a script is just easier when everything is together."
    - Q3: "Modular scripts are easier to track, but not always necessary."
    - Q2: "I prefer modularity for usability."

B. Documentation and Comments

- Most interviewees support introductory blocks explaining the script's purpose.
    - Q1: "I don’t always write an introduction, but when I do, I make it detailed so I don’t forget what the script does."
    - Q2: "If I get another script like this one, I can know what to look for."
    - Q3: "Side notes guide people."
- Yanfly’s documentation style is widely praised.
    - Q1: "Yanfly’s documentation is great because it explains everything upfront."
    - Q2: "Yanfly my beloved. Organized and very good briefing."

C. Debugging & Error Handling

- Consensus that core systems should have debugging, but standalone scripts may not need it.
    - Q1: "I add debug messages when I’m testing, but I usually remove them later."
    - Q3: "Core engine scripts need debugging; individual scripts, not always."

    - Q2: "If the script itself causes errors, that’s on the creator."
- Debugging preferences vary:
    - Q1: "I prefer to test my scripts through trial and error rather than debugging tools."
    - Q2: "Users should debug their own modifications."
    - Q3: "Debugging is needed for complex systems."

D. Warnings & User Control

- Warnings are important but should be relevant.
    - Q1: "I put warnings if removing something might break the script."
    - Q3: "Warnings prevent breaking the script."

E. Standardization vs. Customization

- A standardized script layout is useful, but not mandatory.
    - Q1: "I don’t follow a specific format, but I do keep things consistent."
    - Q3: "Standardization is useful for bigger scripts but unnecessary for small ones."
    - Q2: "Every script introduction is necessary."
- Language preferences vary based on intent.
    - Q1: "If the script is meant to be shared, I write in English. If it’s just for me, I don’t bother."
    - Q3: "If it’s meant to be shared worldwide, yes, translate it."
    - Q2: "User responsible for translation."

2. Patterns Identified Across Key Data Pieces

A. Universal Expectations for Script Layout

1. Scripts should be structured and categorized.
    - All interviewees agree that categorization improves readability.
2. Introductory blocks help, but not everyone writes them.
    - Those who share scripts prioritize introductions more.

3. Modularity is preferred but depends on script complexity.
    - Short scripts can be less modular; large systems need structure.
4. Warnings should exist but not overwhelm.
    - Only necessary warnings should be included.

B. Debugging

- Most agree that debugging is useful, but how it's implemented varies.
    - Q1 removes debug messages after testing.
    - Q2 says debugging should be left to the user.
    - Q3 says debugging should be required for core scripts.

C. Standardization

- A rigid format isn't required, but clarity is necessary.
- English is only needed for shared scripts.
- Users should be able to modify scripts without too much restriction.

3. Reflection on Patterns & Insights for the Research Question

Research Question: What features do RPG Maker VX Ace scripters believe to be part of a universal or ideal set-up for an RPGM script?

- Most scripters value organization, modularity, and documentation.
- There is no single "correct" way to structure a script, but clear categories help.
- Debugging expectations depend on the script’s purpose (core vs. standalone).
- Standardization exists, but flexibility is valued.
- Language and warnings depend on intent (shared vs. private scripts).

These patterns suggest that a universal script format isn't about strict rules, but about widely accepted practices that enhance clarity and usability.

4. Building Evidence-Based Claims

Claim 1: A Well-Structured Script Enhances Readability & Usability

- Evidence: All interviewees agreed that categorization and introductory comments improve clarity.

Claim 2: A Universal Script Structure Should Be Flexible

- Evidence: While standardization is useful, Q1 and Q2 emphasize flexibility, especially for private scripts.

Claim 3: Debugging Expectations Depend on Context

- Evidence: Q1 removes debug messages after testing, while Q3 sees debugging as essential for large systems.

Claim 4: Standardization Should Focus on Functionality Over Format

- Evidence: Q3, Q2, and Q1 all favor usability over strict formatting.

5. Interpretive Frame

The framework for my research is Applied Literacy & Usability, as this research question is analyzing the rhetorical side of code over the practical side, and thus I don't need to spend time interpreting the code itself, only how other's view it.

6. Coding Categories for Primary Data

Category	Definition	Example
Script Organization	How well the script is structured and categorized.	Section headers, grouping methods logically
Documentation & Comments	Presence and clarity of explanations.	Introductory blocks, inline comments
Debugging & Error Handling	Whether debugging features exist and how they are used.	Debug print statements, error-catching mechanisms
Warnings & User Control	How well scripts prevent accidental breaking.	Warning blocks, edit restrictions
Standardization and Customization	Whether scripts should follow standardized layouts.	Structure of blocks, comments, and translation help