LOCALIZATION ENGINEERING • CLASS 1 • MODULE 1

# Reclaiming Control of Your Files & Workbench

Moving from passive document creation to intentional digital architecture for global software delivery.

PATH: /assets/loc/module1_intro.json

# Unmasking Hidden File Extensions

Consumer operating systems hide file extensions by default. In localization pipelines, masked extensions lead to broken automated builds and lost files.

⚠️ **Masked Trap: strings.json displayed on desktop.**

❌ **Underlying Reality: strings.json.txt breaks build scripts.**

✅ **Engineering Rule: Always reveal full extensions in OS settings.**



## Visibility Mandate

Unmasking extensions is your first step toward total file authority.

LESSON 1.1 • DIRECTORY NAVIGATION

# Navigating Digital Neighborhoods

### Absolute Paths

Fixed host machine paths break across team environments.

### Relative Paths

Flexible root referencing works on all developer machines.

### Data Containers

Structured plain text files isolate content from design.

LESSON 1.1 • FILE TYPOLOGIES

# Binary Assets vs. Plain Text Data

## Binary Assets

.docx • .png • .xlsx

Compiled data streams containing proprietary styling, compressed layers, and formatting. Cannot be parsed line-by-line by build scripts.

## Plain Text Data

.json • .vtt • .xml • .csv

Unstyled character streams mapping directly to explicit bytes. Essential for flexible automated translation pipelines.

LESSON 1.2 • PROTECTING MEANING

# How "Smart" Word Processors Corrupt Code

Traditional editors silently alter characters to improve visual layout, destroying structural code syntax.

**Curly "Smart" Quotes (“ ”)**

Breaks JSON string boundaries instantly.

**Auto-Capitalization**

Alters explicit key names like btn_save.

```
// WORD PROCESSOR OUTPUT (CORRUPTED)
{ “btn_start”: “Naciśnij “START”” }

// PLAIN TEXT EDITOR OUTPUT (VALID)
{ "btn_start": "Naciśnij \"START\"" }
```

LESSON 1.3 • TOOLING

# Setting Up Your Professional Workbench

Configuring Visual Studio Code for low-friction, high-precision localization editing.

LESSON 1.3 • VS CODE MECHANICS

# Essential Editor Controls

### Status Bar

Live metrics for encoding, line endings, and syntax type.

### Word Wrap

Prevents horizontal scrolling without adding newlines.

### Line Numbers

Enables precise error targeting during QA audits.

### Global Search

Searches multi-file directory drops instantly.

LESSON 1.3 • SHORTCUTS

# High-Frequency Navigation Commands

**Cmd+Shift+F**

Global Search Across Drop

**Alt + Z**

Toggle Viewport Word Wrap

**Cmd + S**

Save Plain Text State

HANDS-ON LAB • STEP-BY-STEP

# Lab 1: Workspace Setup & File Integrity

1. **Enable OS Visibility:** Unhide file extensions in Finder or File Explorer settings.
2. **Build Folder Tree:** Create local workspace directory structure /assets/loc/.
3. **Configure VS Code:** Set UTF-8 default, auto-guess encoding, and word wrap.
4. **Validate Integrity:** Save plain-text JSON assets without corrupting syntax.

MODULE 1 COMPLETE

# Ready for Character Encodings?

Now that your workbench is set up, Module 2 takes us into global alphabets, character maps, and fixing corrupt Mojibake.

NEXT: MODULE 2 — RESPECTING GLOBAL ALPHABETS