Text Encoding Converter

Project Abstract — projectech.in (built to order) · K-1630

Introduction / Background

Legacy documents, exports from old systems and files from other locales routinely arrive in the wrong encoding, turning readable text into mojibake. This project builds a text encoding converter as a single-file web app: 14+ legacy source decodings, hand-written target encoders, configurable strategies for unencodable characters, mojibake detection with one-click repair, round-trip integrity verification, and batch conversion with per-file integrity reports.

Problem Statement

Encoding bugs are silent data corruption: a file decoded as Windows-1252 instead of UTF-8 looks broken but is perfectly recoverable if you know the original bytes. Most converter tools hide the byte level entirely, so users cannot verify what happened. A converter with hex dumps, a character inventory and a round-trip prover makes every conversion auditable.

Objectives

- Decode 14+ legacy encodings via the platform TextDecoder.
- Hand-write target encoders for UTF-8, UTF-16 LE/BE, Windows-1252, Latin-1, ASCII.
- Offer replace/drop/escape/transliterate strategies for unencodable characters.
- Detect mojibake and repair it in one click.
- Verify round-trip integrity and batch-convert with integrity reports.

Proposed Methodology

Source bytes are decoded with TextDecoder (fatal:false) after BOM sniffing; auto-detect combines BOM checks, strict UTF-8 validation and CJK byte-pattern heuristics. Target encoders are hand-written, including the Windows-1252 0x80–0x9F mapping table; unencodable code points follow the selected strategy. Mojibake detection tests whether bytes valid as strict UTF-8 were decoded as legacy. The verifier re-decodes output and diffs against source text.

System Architecture / Working

Single-file web app: queue layer (5 real multi-encoding sample files as ArrayBuffers), decode layer (encoding selector, auto-detect, hex dump), analysis layer (character inventory, mojibake banner), encode layer (target selector, strategy selector, BOM option, output hex dump), and batch layer (queue conversion with integrity report). No dependencies, no network.

Hardware / Software Requirements

- Hardware: any laptop/desktop with a modern browser (2 GB RAM minimum).
- Software: current Chrome, Edge, Firefox or Safari (TextDecoder support); no installation.
- Input: text files in any supported encoding.

Expected Outcomes

- A working converter that provably preserves text across encodings.
- Readable hand-written encoder implementations for five targets.
- A mojibake repair flow that rescues misdecoded files.
- Complete viva kit: report, PPT and Q&A.

Applications

- CS/IT coursework on data representation and Unicode.
- Data migration projects recovering legacy archives.
- Localization workflows normalizing file encodings.

Future Scope

- Streaming conversion for very large files.
- Encoding detection via statistical language models.
- Additional targets (EBCDIC, ISO-2022-JP).
- CLI build for scripted batch pipelines.
