Base Number Converter — Program Documentation

Project overview

A small single-page web application that converts numbers between Binary, Octal, Decimal and Hexadecimal. The app provides a simple UI: an input field for the number, a dropdown to select the input base, and a "Convert" control that displays equivalent values in the other bases.

> Source: live demo page found at the project URL.

Features

- Convert from one base (binary, octal, decimal, hexadecimal) to the others.
- Simple browser-based UI; no server required.

Prerequisites

- Modern web browser (Chrome, Firefox, Edge, Safari).
- No build tools required if using the hosted static files.

Installation / Running locally

1. Clone or download the repository from the project hosting (GitHub Pages) link.
2. If you downloaded the files, open `index.html` in your browser. The app runs entirely client-side.

Expected file structure (typical)

```
NUMBER-SYSTEM-CONVERSION-SYSTEM-PROJECT-/
├─ index.html        # UI and input controls
├─ style.css         # Optional styling
└─ script.js         # Conversion logic and DOM handlers
```

> If your repository uses different filenames or a single HTML file with embedded CSS/JS, adjust the steps above.

How it works (algorithm)

1. Read the input string and the chosen input base (2, 8, 10, or 16).
2. Validate the input characters are valid for the selected base (e.g., only `0` and `1` for binary; `0–7` for octal; `0–9` for decimal; `0–9` and `A–F` for hex).

3. Parse the input into an integer (commonly by converting to decimal using `parseInt(input, base)` in JavaScript).
4. Produce output strings for each target base using built-in formatting functions (e.g., `number.toString(targetBase).toUpperCase()` for hex letters).

Usage (user-facing)

1. Enter the number you want to convert into the `Enter Number` field.
2. Choose the input base from the `Select Input Base` dropdown (Binary / Octal / Decimal / Hexadecimal).
3. Click the `Convert` button. The converted values will be shown for each of the other bases.

Example conversions

- Input: `1011` (Binary) → Decimal: `11`, Octal: `13`, Hex: `B`.
- Input: `255` (Decimal) → Binary: `11111111`, Octal: `377`, Hex: `FF`.

Validation & Error handling

- The UI should prevent or show a clear error if the input contains invalid characters for the chosen base.
- Edge cases to consider:
- Empty input — show a friendly prompt.
- Very large inputs — decide whether to support BigInt (JS `BigInt`) or show a size limit message.
- Negative numbers — decide whether to support a minus sign and how to show two's-complement vs sign-magnitude.

Implementation notes (suggested JS snippets)

```
// parse input to decimal
const decimal = parseInt(inputString.trim(), inputBase);
if (Number.isNaN(decimal)) {
  // show error to user
}
// produce outputs
const binary = decimal.toString(2);
const octal  = decimal.toString(8);
const hex    = decimal.toString(16).toUpperCase();
```

If you want support for arbitrarily large integers, consider using `BigInt` and manual base conversion routines.

Accessibility & UX suggestions

- Add `label` elements for the input and select controls.

- Allow pressing Enter to trigger conversion.
- Provide clear error messages with `role="alert"` and focus management.

Tests

- Manual test cases: sample conversions (see examples above).
- Automated: add a small test harness (Jest or plain JS) that verifies conversion for a table of inputs across bases.

Potential improvements

- Add paste/auto-detect: detect the base automatically (e.g., `0b` prefix for binary, `0x` for hex) with an option to override.
- Add copy-to-clipboard buttons for outputs.
- Support fractional numbers (e.g., `101.101` → fractional conversions). This requires implementing fractional base conversion logic.
- Add history or bookmarking of conversions.

License

- If the original repository includes a LICENSE file, follow that license. If not, add a suitable license (MIT is common for small web projects).

If you'd like, I can: - generate a `README.md` file ready to commit with this documentation, or - extract the actual filenames and exact code from your repo and produce a line-by-line README based on the real files.

Tell me which you'd prefer and I'll prepare the file for download or commit-ready text.