You can format a `Date` in **TypeScript** to include:

- **Day name abbreviation** (e.g., `Mon`)
- **Day number** (e.g., `31`)
- **Month abbreviation** (e.g., `Dec`)

using `toLocaleDateString()` with the right `Intl.DateTimeFormatOptions`.

Complete TypeScript Example

```
// TypeScript: Format date with abbreviated day name, day number, and abbreviated month
function formatDate(date: Date, locale: string = 'en-US'): string {
    if (!(date instanceof Date) || isNaN(date.getTime())) {
        throw new Error("Invalid Date object provided.");
    }

    // Options for abbreviated weekday, numeric day, and abbreviated month
    const options: Intl.DateTimeFormatOptions = {
        weekday: 'short', // e.g., Mon, Tue
        day: 'numeric',   // e.g., 31
        month: 'short'    // e.g., Dec
    };

    return date.toLocaleDateString(locale, options);
}

// Example usage:
try {
    const today = new Date();
    console.log(formatDate(today));           // e.g., "Wed, Jan 31"
    console.log(formatDate(today, 'en-GB'));  // e.g., "Wed, 31 Jan"
    console.log(formatDate(today, 'it-IT'));  // e.g., "mer 31 gen"
} catch (err) {
    console.error(err);
}
```

Key Points

- `weekday: 'short'` → abbreviated day name (`Mon`, `Tue`, `Wed`).
- `day: 'numeric'` → numeric day of the month.
- `month: 'short'` → abbreviated month name (`Jan`, `Feb`, `Mar`).
- **Locale** affects language and order (e.g., `'en-US'` vs `'it-IT'`).
- Includes **input validation** to avoid runtime errors.

✅ This works in **TypeScript** and **JavaScript** in browsers and Node.js (v13+ with full `Intl` support).

If you want, I can also give you a **version that always outputs in English abbreviations regardless of system locale**. Would you like me to prepare that?