```
2586 +   private transformModuleData(data: any, language: string): any {
2587 +     if (!data || typeof data !== 'object') {
2588 +       return data;
2589 +     }
2590 +
2591 +     // If this is the old format with language at top level, extract it
2592 +     if (!Array.isArray(data) && data[language] && typeof data[language] === 'object') {
2593 +       const hasLanguageKeys = Object.keys(data).every(key =>
2594 +         ['en', 'es', 'fr'].includes(key)
2595 +       );
2596 +       if (hasLanguageKeys) {
2597 +         return this.transformModuleData(data[language] || data.en, language);
2598 +       }
2599 +     }
2600 +
2601 +     // Handle arrays
2602 +     if (Array.isArray(data)) {
2603 +       return data.map(item => this.transformModuleData(item, language));
2604 +     }
2605 +
2606 +     // Handle objects - recursively transform all fields
2607 +     const transformed = { ...data };
2608 +
2609 +     Object.keys(transformed).forEach(key => {
2610 +       const value = transformed[key];
2611 +
2612 +       if (value === null || value === undefined) {
2613 +         return;
2614 +       }
2615 +
2616 +       // Handle arrays
2617 +       if (Array.isArray(value)) {
2618 +         transformed[key] = value.map(item => this.transformModuleData(item, language));
2619 +       }
2620 +       // Handle objects
2621 +       else if (typeof value === 'object') {
2622 +         const keys = Object.keys(value);
2623 +         const allKeysAreLanguages = keys.length > 0 && keys.every(k => ['en', 'es', 'fr'].includes(k));
2624 +
2625 +         if (allKeysAreLanguages) {
2626 +           // This is a translation object, extract the language value
2627 +           transformed[key] = value[language] || value.en || Object.values(value)[0];
2628 +         } else {
2629 +           // This is a nested object, recurse into it
2630 +           transformed[key] = this.transformModuleData(value, language);
2631 +         }
2632 +       }
2633 +     });
2634 +
2635 +     return transformed;
2636 +   }
2637 +
```