

TypeScript Fixes Documentation

Date: December 11, 2025

Branch: expenses-structure

Status:  All errors resolved (0 errors remaining)

Summary

Fixed all TypeScript compilation errors related to null/undefined type handling in invoice page event handlers. All 9 errors were concentrated in a single file with the same root cause.

Errors Found

Initial TypeScript Check Results

```
npx tsc --noEmit
```

Total Errors: 9

Affected Files: 1

File: app/(dashboard)/(modules)/invoices/[id]/page.tsx

Error Breakdown

Line	Error Type	Description
562	TS18047	'data.contract' is possibly 'null'
562	TS18047	'data.contract.bank' is possibly 'null'
562	TS2345	Argument of type 'string null' not assignable to 'string'
580	TS18047	'data.contract' is possibly 'null'
580	TS18047	'data.contract.bank' is possibly 'null'
580	TS2345	Argument of type 'string null' not assignable to 'string'
598	TS18047	'data.contract' is possibly 'null'
598	TS18047	'data.contract.bank' is possibly 'null'
598	TS2345	Argument of type 'string null' not assignable to 'string'

Root Cause Analysis

Category: Null/Undefined Type Narrowing Issues

Problem:

TypeScript does not maintain type narrowing inside nested function scopes (e.g., event handlers). Even though the parent component had conditional checks for `data.contract?.bank`, the onClick handlers could not guarantee that these values would be non-null at runtime from TypeScript's perspective.

Affected Code Pattern:

```
// Parent has null check
{data.contract?.bank && (
  <Button onClick={() => {
    // TypeScript loses type narrowing here
    navigator.clipboard.writeText(data.contract.bank.accountNumber); // ✘ Error
  }}>
)}
```

Fixes Applied

Fix 1: Account Number Copy Handler (Line 562)

Before:

```
onClick={() => {
  navigator.clipboard.writeText(data.contract.bank.accountNumber);
  toast.success("Account number copied to clipboard");
}}
```

After:

```
onClick={() => {
  const accountNumber = data.contract?.bank?.accountNumber;
  if (accountNumber) {
    navigator.clipboard.writeText(accountNumber);
    toast.success("Account number copied to clipboard");
  }
}}
```

Explanation:

- Used optional chaining (`?.`) to safely access nested properties
- Added explicit null check before using the value
- Stored value in local variable with proper type narrowing

Fix 2: IBAN Copy Handler (Line 580)

Before:

```
onClick={() => {
  navigator.clipboard.writeText(data.contract.bank.iban);
  toast.success("IBAN copied to clipboard");
}}
```

After:

```
onClick={() => {
  const iban = data.contract?.bank?.iban;
  if (iban) {
    navigator.clipboard.writeText(iban);
    toast.success("IBAN copied to clipboard");
  }
}}
```

Explanation:

- Same pattern as Fix 1
- Ensures IBAN value exists before copying to clipboard

Fix 3: SWIFT Code Copy Handler (Line 598)**Before:**

```
onClick={() => {
  navigator.clipboard.writeText(data.contract.bank.swiftCode);
  toast.success("SWIFT code copied to clipboard");
}}
```

After:

```
onClick={() => {
  const swiftCode = data.contract?.bank?.swiftCode;
  if (swiftCode) {
    navigator.clipboard.writeText(swiftCode);
    toast.success("SWIFT code copied to clipboard");
  }
}}
```

Explanation:

- Same pattern as Fix 1 and 2
- Ensures SWIFT code value exists before copying

Common Patterns Identified**Pattern 1: Type Narrowing in Event Handlers**

Issue: TypeScript doesn't maintain type narrowing from parent scope inside nested functions.

Solution: Re-check types inside the event handler scope using:

- Optional chaining (?.)
- Explicit null checks
- Type guards

Pattern 2: Conditional Rendering vs Runtime Type Safety

Issue: JSX conditional rendering ({value && <Component />}) doesn't guarantee TypeScript type narrowing inside nested callbacks.

Solution: Always validate nullable values at the point of use, especially in event handlers.

Best Practices Applied

1. **Avoided `@ts-ignore` / `@ts-expect-error`** - Used proper type guards instead
 2. **No non-null assertions (`!`)** - Preferred explicit checks
 3. **Optional chaining** - Used `?.` for safe property access
 4. **Local variable extraction** - Improved readability and type inference
 5. **Runtime safety** - Fixes ensure no runtime errors even if data structure changes
-

Verification

Final TypeScript Check

```
npx tsc --noEmit
```

Result: 0 errors

Test Build

Build command was not run as per task requirements, but compilation check confirms no blocking errors.

Files Modified

1. **app/(dashboard)/(modules)/invoices/[id]/page.tsx**
 - Lines 561-567: Fixed accountNumber copy handler
 - Lines 582-588: Fixed IBAN copy handler
 - Lines 603-609: Fixed swiftCode copy handler
-

Related Context

Recent Schema Changes

- Added `currencyId` to Invoice model (Prisma schema)
- Updated `sendToAgency` mutation in timesheet router
- Enhanced invoice page with payment details section

Why This Matters

These fixes were necessary after recent enhancements to the invoice page that added bank account detail display and copy-to-clipboard functionality. The fixes ensure type safety while maintaining the user experience.

Recommendations for Future Development

1. **Type Inference:** Consider creating a type guard utility for commonly checked patterns:

```
typescript
function assertBankDetails(contract: Contract | null): asserts contract is Contract &
{ bank: NonNullable<Contract['bank']> } {
    if (!contract?.bank) throw new Error("Bank details required");
}
```

2. **Event Handler Patterns:** Establish a team convention for handling nullable data in event handlers.

3. **Prisma Type Generation:** Regularly run `npx prisma generate` after schema changes to keep types in sync.

4. **Strict Null Checks:** Continue enforcing `strictNullChecks` in `tsconfig.json` for better type safety.
-

Conclusion

All TypeScript errors have been successfully resolved with proper null handling patterns. The codebase now compiles without errors while maintaining runtime safety and code readability. No workarounds or type assertions were used, ensuring long-term maintainability.

Final Status:  **COMPLETE - 0 TypeScript Errors**