TypeScript Migration Roadmap for Backend Server

Overview

This roadmap outlines the step-by-step process to migrate the backend server from JavaScript (with Babel) to TypeScript and resolve all **deprecated** dependencies.

Current State Analysis

Current Stack

- **Runtime**: Node.js 18+
- **Transpiler**: Babel 7.x
- **Framework**: Express 4.16.3
- **GraphQL**: Apollo Server Express 3.12.1
- **Database**: MongoDB with Mongoose 5.9.13
- **Testing**: Jest 26.0.1, Mocha 7.1.2
- **Code Style**: ES6 modules (import/export)

Deprecated/Outdated Dependencies Identified

Critical Deprecations

1. **`babel-core`** → Should be `@babel/core` (already have, but old version)
2. **`request`** → Deprecated, replace with `axios` or native `fetch`
3. **`subscriptions-transport-ws`** → Deprecated, already using `graphql-ws` (can remove)
4. **`graphql-server-express`** → Deprecated, using `apollo-server-express` (can remove)
5. **`@babel/polyfill`** → Deprecated, use `core-js` directly
6. **`babel-eslint`** → Deprecated, use `@babel/eslint-parser`
7. **`body-parser`** → Now built into Express 4.16.0+
8. **`path`** → Not needed (Node.js built-in)
9. **`promise`** → Not needed (native Promise)
10. **`prop-types`** → React-specific, shouldn't be in backend
11. **`http`** → Security placeholder package, not needed
12. **`underscore`** → Redundant with `lodash`

Outdated Versions (Major Updates Needed)

- `mongoose`: 5.9.13 → 8.x
- `express`: 4.16.3 → 4.21.x

- `axios`: 0.21.1 → 1.x
- `jest`: 26.0.1 → 29.x
- `mocha`: 7.1.2 → 10.x
- `winston`: 3.2.1 → 3.15.x
- `cors`: 2.8.3 → 2.8.5
- `dotenv`: 8.2.0 → 16.x
- `pm2`: 4.4.0 → 5.x
- `stripe`: 8.89.0 → 14.x
- `mongodb`: 3.5.7 → 6.x
- `apollo-server-express`: 3.12.1 → 4.x (or stay on 3.x latest)
- `graphql`: 16.0.0 → 16.9.x (latest stable)

Migration Phases

Phase 1: Preparation & Setup

1.1 Create TypeScript Configuration

- [ ] Create `tsconfig.json` with appropriate compiler options
- [ ] Configure path aliases to match current `~` root import pattern
- [ ] Set up TypeScript strict mode gradually

- [ ] Configure build output directory (`dist/`)

1.2 Update Package Dependencies

- [ ] Remove deprecated packages:
    - `babel-core`
    - `subscriptions-transport-ws`
    - `graphql-server-express`
    - `@babel/polyfill`
    - `babel-eslint`
    - `body-parser` (use Express built-in)
    - `path`
    - `promise`
    - `prop-types`
    - `http`
    - `underscore`

- [ ] Update to latest stable versions:
    - `mongoose`: 5.9.13 → 8.2.0
    - `express`: 4.16.3 → 4.21.0
    - `axios`: 0.21.1 → 1.7.7
    - `jest`: 26.0.1 → 29.7.0
    - `mocha`: 7.1.2 → 10.4.0
    - `winston`: 3.2.1 → 3.15.0
    - `cors`: 2.8.3 → 2.8.5
    - `dotenv`: 8.2.0 → 16.4.5
    - `pm2`: 4.4.0 → 5.3.1
    - `stripe`: 8.89.0 → 14.21.0
    - `mongodb`: 3.5.7 → 6.3.0
    - `apollo-server-express`: 3.12.1 → 3.12.2 (or evaluate 4.x)
    - `graphql`: 16.0.0 → 16.9.0
- [ ] Replace `request` with `axios` or native `fetch`

- [ ] Add TypeScript dependencies:
    - `typescript`: ^5.6.0
    - `@types/node`: ^20.14.0
    - `@types/express`: ^4.17.21
    - `@types/cors`: ^2.8.17
    - `@types/bcryptjs`: ^2.4.6
    - `@types/jsonwebtoken`: ^9.0.6
    - `@types/lodash`: ^4.17.7
    - `@types/mocha`: ^10.0.6
    - `@types/jest`: ^29.5.12
    - `@types/mongoose`: ^5.11.97
    - `@types/ws`: ^8.5.10
    - `ts-node`: ^10.9.2 (for development)
    - `ts-node-dev`: ^2.0.0 (replaces nodemon + babel-node)
    - `ts-jest`: ^29.1.5 (for Jest TypeScript support)

1.3 Update ESLint Configuration

- [ ] Replace `babel-eslint` with `@babel/eslint-parser`
- [ ] Add `@typescript-eslint/parser` and `@typescript-eslint/eslint-plugin`

- [ ] Update ESLint config for TypeScript
- [ ] Configure ESLint to handle both `.js` and `.ts` files during migration

1.4 Update Build Scripts

- [ ] Update `package.json` scripts:
    - Replace `babel-node` with `ts-node` or `ts-node-dev`
    - Update build script to use `tsc` instead of `babel`
    - Update test scripts for TypeScript support

Phase 2: TypeScript Infrastructure

2.1 Create Type Definitions

- [ ] Create `src/types/` directory structure
- [ ] Define base types:
    - `types/common.ts` - Common types (User, Post, etc.)
    - `types/graphql.ts` - GraphQL context, resolvers
    - `types/mongoose.ts` - Mongoose model types
    - `types/environment.ts` - Environment variables

- [ ] Create type definitions for existing models

2.2 Update Module Resolution

- [ ] Configure TypeScript path mapping for `~` imports
- [ ] Update `tsconfig.json` paths to match Babel root import plugin
- [ ] Test module resolution works correctly

2.3 Update Jest Configuration

- [ ] Update `jest.config.js` to use `ts-jest`
- [ ] Configure Jest to handle TypeScript files
- [ ] Update module name mapper for TypeScript paths
- [ ] Ensure test mocks work with TypeScript

Phase 3: Incremental Migration

3.1 Start with Utilities (Low Risk) -

- [ ] Migrate `app/data/utils/` directory:
    - `logger.ts`
    - `constants.ts`
    - `authentication.ts`
    - `requireAuth.ts`
    - `rateLimiter.ts`
    - `send-grid-mail.ts`
    - `presence/cleanupStalePresence.ts`
- [ ] Add proper type annotations
- [ ] Test each utility after migration

3.2 Migrate Models (Medium Risk)

- [ ] Migrate `app/data/resolvers/models/` directory:
    - Start with simpler models (UserModel, PostModel, etc.)
    - Add Mongoose schema types
    - Update model exports with proper types
    - Test database operations after each model

3.3 Migrate GraphQL Schema (Medium Risk)

- [ ] Migrate `app/data/type_definition/`:
    - Convert to TypeScript
    - Add type annotations for GraphQL types
    - Ensure GraphQL schema generation works
- [ ] Migrate `app/data/types/` (GraphQL types)
- [ ] Migrate `app/data/inputs/`
- [ ] Migrate `app/data/resolvers/scalars/`

3.4 Migrate Resolvers (Higher Risk)

- [ ] Migrate `app/data/resolvers/queries/`:
    - Start with simpler queries
    - Add proper return types
    - Test each query after migration
- [ ] Migrate `app/data/resolvers/mutations/`:
    - Start with simpler mutations
    - Add proper input/output types
    - Test each mutation after migration
- [ ] Migrate `app/data/resolvers/relationship/`
- [ ] Migrate `app/data/resolvers/subscriptions.ts`

3.5 Migrate Main Server File (High Risk)

- [ ] Migrate `app/server.js` → `app/server.ts`

- [ ] Add proper types for Express, Apollo Server
- [ ] Update WebSocket server types
- [ ] Test server startup and all endpoints

Phase 4: Dependency Updates & Fixes

4.1 Mongoose Migration (5.x → 8.x)

- [ ] Review Mongoose 8.x breaking changes
- [ ] Update connection options (remove deprecated options)
- [ ] Update model definitions:
    - Remove `useNewUrlParser`, `useUnifiedTopology`, `useCreateIndex`, `useFindAndModify`
    - Update schema definitions if needed
    - Fix any deprecated query methods
- [ ] Test all database operations
- [ ] Update Mongoose types

4.2 Express & Middleware Updates

- [ ] Remove `body-parser` dependency (use Express built-in)
- [ ] Update Express middleware usage
- [ ] Test all routes and middleware

4.3 Apollo Server Updates

- [ ] Review Apollo Server 3.x → 4.x migration (if upgrading)
- [ ] Update Apollo Server configuration
- [ ] Test GraphQL endpoint and subscriptions
- [ ] Verify WebSocket connections work

4.4 Replace Deprecated `request` Package

- [ ] Find all usages of `request` package
- [ ] Replace with `axios` or native `fetch`
- [ ] Update error handling
- [ ] Test all HTTP requests

4.5 Testing Framework Updates

- [ ] Update Jest configuration for TypeScript
- [ ] Update Mocha configuration (if still using)
- [ ] Migrate test files to TypeScript
- [ ] Ensure all tests pass

Phase 5: Code Quality & Optimization

5.1 Enable TypeScript Strict Mode

- [ ] Gradually enable strict mode options:
    - `strict: true`
    - `noImplicitAny: true`
    - `strictNullChecks: true`
    - `strictFunctionTypes: true`
    - `strictBindCallApply: true`
    - `strictPropertyInitialization: true`
    - `noImplicitThis: true`
    - `alwaysStrict: true`
- [ ] Fix all type errors
- [ ] Add proper null checks

5.2 Add Type Safety

- [ ] Add types for all function parameters and return values
- [ ] Add types for GraphQL resolvers
- [ ] Add types for database models
- [ ] Add types for environment variables
- [ ] Add types for API responses

5.3 Remove Babel Dependencies

- [ ] Remove all Babel-related packages:
    - `@babel/cli`
    - `@babel/core`
    - `@babel/node`
    - `@babel/preset-env`
    - `@babel/preset-react`
    - All `@babel/plugin-*` packages
    - `@babel/register`
    - `@babel/runtime`
    - `@babel/transform-*` packages
    - `babel-loader`
    - `babel-plugin-root-import`
- [ ] Remove `.babelrc` file
- [ ] Update all scripts to use TypeScript

5.4 Update Documentation

- [ ] Update README with TypeScript instructions
- [ ] Document new build process
- [ ] Update development setup instructions
- [ ] Document type definitions structure

Phase 6: Testing & Validation

6.1 Comprehensive Testing

- [ ] Run all unit tests
- [ ] Run all integration tests
- [ ] Test all GraphQL queries
- [ ] Test all GraphQL mutations
- [ ] Test all GraphQL subscriptions
- [ ] Test authentication flows
- [ ] Test database operations
- [ ] Test external API integrations

6.2 Performance Testing

- [ ] Compare build times (Babel vs TypeScript)
- [ ] Compare runtime performance
- [ ] Check bundle sizes
- [ ] Optimize if needed

6.3 Code Review

- [ ] Review all TypeScript code
- [ ] Check for any `any` types
- [ ] Ensure proper error handling
- [ ] Verify type safety throughout

Phase 7: Deployment & Cleanup (Week 7)

7.1 Update Deployment Configuration

- [ ] Update Dockerfile for TypeScript build
- [ ] Update PM2 ecosystem config
- [ ] Update CI/CD pipelines
- [ ] Test production build

7.2 Final Cleanup

- [ ] Remove all `.js` files (after migration complete)
- [ ] Remove Babel configuration files
- [ ] Clean up unused dependencies
- [ ] Update `.gitignore` for TypeScript
- [ ] Remove old build artifacts

7.3 Production Deployment

- [ ] Deploy to staging environment
- [ ] Run smoke tests
- [ ] Monitor for errors
- [ ] Deploy to production
- [ ] Monitor production metrics

Risk Mitigation

High-Risk Areas

1. **Mongoose 5.x → 8.x**: Major version jump with breaking changes
2. **Apollo Server**: Potential breaking changes in 4.x
3. **GraphQL Subscriptions**: WebSocket implementation changes
4. **Authentication**: Token verification and user context

Mitigation Strategies

1. **Incremental Migration**: Migrate one module at a time
2. **Comprehensive Testing**: Test after each migration step
3. **Feature Flags**: Use feature flags for gradual rollout
4. **Rollback Plan**: Keep old code until migration is verified
5. **Staging Environment**: Test thoroughly in staging before production

---

Success Criteria

- [ ] All JavaScript files migrated to TypeScript
- [ ] All deprecated packages removed

- [ ] All dependencies updated to latest stable versions
- [ ] All tests passing
- [ ] No TypeScript errors in strict mode
- [ ] Build process working correctly
- [ ] Production deployment successful
- [ ] No runtime errors in production
- [ ] Performance improved

Resources & References

- TypeScript Handbook
- Mongoose 8.x Migration Guide
- Apollo Server Migration Guide
- Express TypeScript Guide
- Jest TypeScript Setup
