Les 4: Huiswerkopdracht

JavaScript naar TypeScript Converter

Deadline: Voor het begin van Les 5 | **Type:** JS naar TS omzetten
Benodigdheden: Cursor, Terminal, Node.js
Geschatte tijd: 1,5 - 2 uur

WAT GA JE DOEN?

Je krijgt een werkend JavaScript project met utility functies voor een e-commerce systeem. Jouw taak: **alles omzetten naar TypeScript** zonder any types te gebruiken.

Je leert: interfaces schrijven, union types gebruiken, functies typen, en TypeScript errors lezen en oplossen.

De tests staan al in TypeScript — die vertellen je wat de verwachte types zijn!

STAP 0: PROJECT DOWNLOADEN

■ Download `les4-huiswerk-js-converter.zip` van Teams

■ Unzip het bestand en open de folder in Cursor:

```
unzip les4-huiswerk-js-converter.zip
```

```
cd les4-huiswerk-js-converter
```

```
cursor .
```

■ Installeer dependencies:

```
npm install
```

Projectstructuur:

```
les4-huiswerk-js-converter/
src/
users.js <-- omzetten naar .ts
products.js <-- omzetten naar .ts
orders.js <-- omzetten naar .ts
utils.js <-- omzetten naar .ts
__tests__/
users.test.ts (niet aanpassen!)
products.test.ts
orders.test.ts
utils.test.ts
tsconfig.json
package.json
```

Belangrijk: De test bestanden (`__tests__/`) bevatten al TypeScript interfaces. Bekijk ze als hint voor de verwachte types!

DE 4 BESTANDEN

Bestand 1: src/users.js → users.ts

Functies:

• `createUser(name, email, age)` — maakt een nieuwe user aan

• `findUser(users, id)` — zoekt een user op id

• `updateUser(users, id, updates)` — update user gegevens

• `filterActiveUsers(users)` — geeft alle actieve users terug

Wat je moet doen:

- ■ Maak een `User` interface met: `id`, `name`, `email`, `age`, `isActive`
- ■ Type alle function parameters en return types
- ■ Geen `any`!

Bestand 2: src/products.js → products.ts

Functies:

• `createProduct(name, price, category)` — maakt een nieuw product aan

• `applyDiscount(product, discountPercent)` — past korting toe

• `getExpensiveProducts(products, minPrice)` — filtert dure producten

• `formatPrice(price)` — formatteert prijs als string

Wat je moet doen:

- ■ Maak een `Product` interface met: `id`, `name`, `price`, `category`, `discount` (optioneel)

- Maak een union type: `"electronics" | "clothing" | "books" | "other"`
- Type alle functies compleet — geen `any`!

Bestand 3: src/orders.js → orders.ts

Functies:

• `createOrder(user, products)` — maakt een order aan

• `calculateTotal(order)` — berekent totale prijs

• `getOrderStatus(order)` — geeft de status terug

• `getOrdersByUser(orders, userId)` — filtert orders per user

Wat je moet doen:

- Maak een `Order` interface met: `id`, `userId`, `products`, `total`, `status`, `createdAt`
- Maak een union type: `"pending" | "processing" | "shipped" | "delivered"`
- `products` moet `Product[]` zijn, `userId` moet een `string` zijn

Bestand 4: src/utils.js → utils.ts

Functies:

• `formatDate(date)` — formatteert een Date naar "DD-MM-YYYY"

• `generateId()` — genereert een willekeurige string ID

• `validateEmail(email)` — checkt of email geldig is

• `sortBy(items, key)` — sorteert een array op een eigenschap

Wat je moet doen:

- `formatDate`: ontvangt `Date`, geeft `string` terug
- `generateId`: geeft `string` terug

■ `validateEmail`: geeft `boolean` terug

■ `sortBy`: **bonus!** Probeer generics: `function sortBy<T>(items: T[], key: keyof T): T[]`

VEREISTEN

Regel	Status
Geen `any` types	■
Geen `// @ts-ignore` comments	■
Alle functies volledig getypt (parameters + return)	■
Interfaces voor alle objecten (User, Product, Order)	■
Union types voor ProductCategory en OrderStatus	■
Optional properties waar nodig (?)	■
`npm run check` geeft 0 errors	■
`npm test` is groen	■

STAP 2: VERIFY & TEST

Als je klaar bent, run deze commands:

`npm run check` (moet 0 errors tonen)

`npm test` (alle tests moeten slagen)

De tests zijn al geschreven in TypeScript! Ze checken of jouw types correct zijn. Als tests mislukken, kijk naar de foutmelding — die vertelt je precies wat er fout is.

TIPS & TRICKS

1. Begin met interfaces

Definieer eerst de shapes van je data. Dan schrijf je veel makkelijker de functies.

2. Kijk naar de tests als hint

Open de test files in `__tests__/` — die laten zien welke types verwacht worden.

3. Hover voor informatie

Hover over rode squiggles in Cursor voor error details. Lees de TypeScript error messages!

4. Union types zijn beter dan strings

`type Status = "pending" | "shipped"` is veel beter dan `status: string`

5. Optional properties met ?

Niet alles is verplicht: `discount?: number` maakt een veld optioneel.

6. Ga stap voor stap

Rename .js naar .ts, type een bestand af, check met `npm run check`, ga verder.

STAP 3: GIT COMMIT & INLEVEREN

Terminal commando's:

```
git add .
```

```
git commit -m "feat: convert JavaScript utilities to TypeScript"
```

```
git push
```

Branch: `feature/les4-typescript-converter`

INLEVER CHECKLIST

- ■ Alle 4 bestanden omgezet naar .ts
- ■ Interfaces geschreven voor User, Product, Order
- ■ Union types voor ProductCategory en OrderStatus
- ■ Alle functies volledig getypt
- ■ Geen `any` of `@ts-ignore`
- ■ `npm run check` = 0 errors
- ■ `npm test` = groen
- ■ Git commit + push gedaan

Alles afgevinkt? Goed bezig! Je bent klaar voor Les 5: TypeScript voor React.

HULP NODIG?

`Type 'X' is not assignable to type 'Y'` → Je gebruikt het verkeerde type

`Property 'X' does not exist on type 'Y'` → Voeg property toe aan interface

`Object is possibly 'undefined'` → Check eerst of het bestaat (`if (x) {...}`)

Tests falen? → Kijk in de test file welk type verwacht wordt

Echt vast? → Hover over de error in Cursor, lees de melding, en kijk terug naar de Les 4 slides