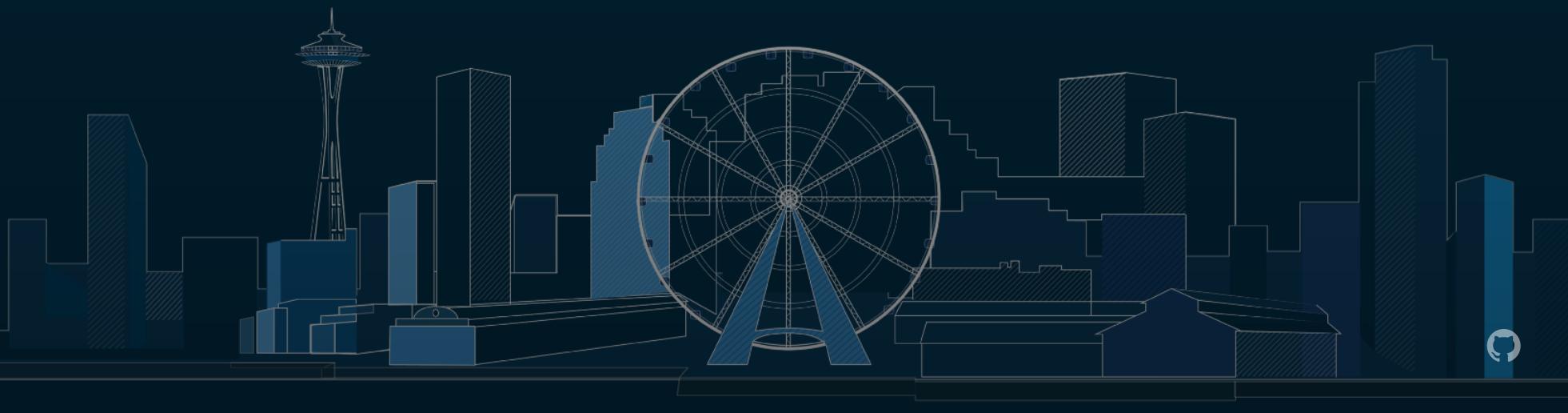
TypeScript

Generics



Teamwork

Mina Sameh Wadie
Ahmed Hamdy Mohammed
Ahmed Hamdy Ameen

What Are Generics?

They are a feature of statically typed languages, they help a lot and allow us to:

- Pass types as parameters to other types, functions or classes.
- Reuse components and parts.

They allow us to create components that can work over a variety of types instead of just one.

Read more about Generics

Hello world in generics

```
function returnWhatIpassIn(item: string): string {
  return item
}

const StringItem = returnWhatIpassIn('string') // string
const NumItem = returnWhatIpassIn(3) // error
```

Using generics

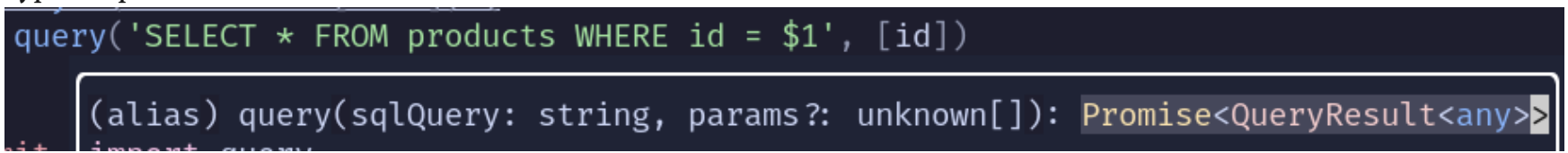
```
function returnWhatIpassIn<TItem>(item: TItem): TItem {
  return item
}
const StringItem = returnWhatIpassIn('string') // string
const NumItem = returnWhatIpassIn(3) // number
```

Arrays

```
function returnWhatIPassInArray<TItem>(item: TItem[]): TItem[] {
  return item
}
const StringItem = returnWhatIPassInArray(['string']) // Array<string>
const NumItem = returnWhatIPassInArray([3]) // Array<number>
const Item = returnWhatIPassInArray(1) // Error
```

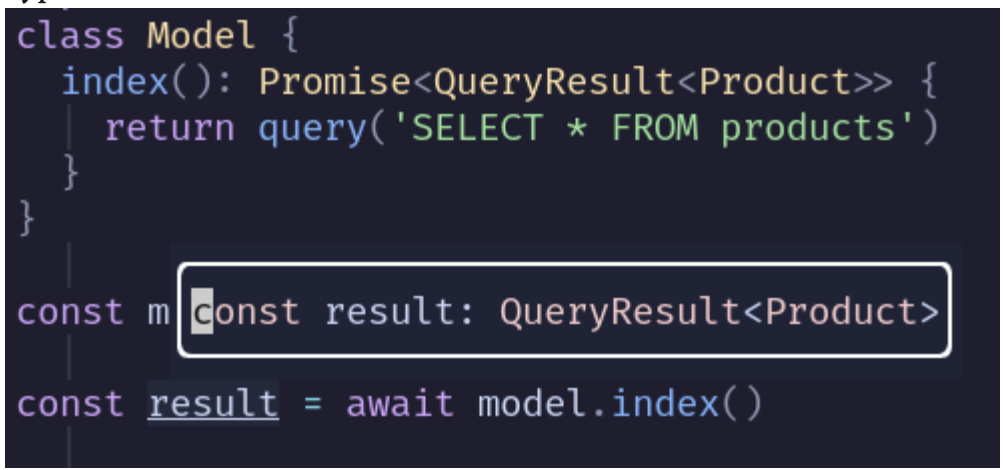
Example with nodejs pg

PG has a type called QueryResult, which by default returns any, That any means we don’t get to use typescript features.



Example with nodejs pg

However it is a generic that accepts types, meaning we can fix this and get typescript features by giving it a type.



and we get autocomplete and type checking, the features of ts that we love.

Example with nodejs pg

Say we want to return one row, instead of writing:

```
import { QueryResult } from 'pg'
import { Product } from './types'

function show(id: string): Promise<QueryResult<Product>['rows'][0]> {
    const result = await query('SELECT * products WHERE id = $1;', [id])
    return result.rows[0]
}
```

As you can see we get typescript type checking, note that the result might be undefined if not found

Example with nodejs pg

Instead of having to write `` `Promise<QueryResult<Product>['rows'][0]>` `` every time, we can use a generic:

```
export type queryReturnRow<Type> = Promise<QueryResult<Type>['rows'][0] | undefined>;
export type ProductRow = queryReturnRow<Product>;
export type UserRow = queryReturnRow<User>; // Reuseability.

function show(id: string): ProductRow {
    const result = await query('SELECT * products WHERE id = $1;', [id]);
    return result.rows[0];
}
```