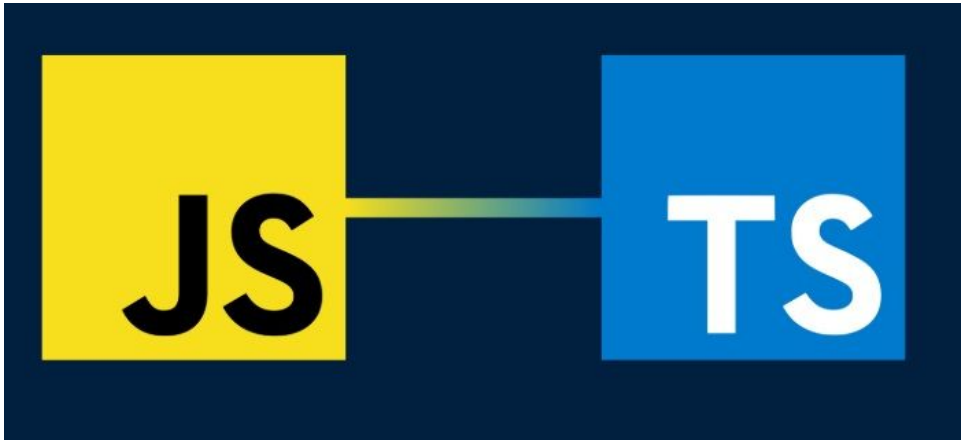
Basic Typescript

Point of Discussion

- Apa Itu TypeScript
- Instalasi
- Tipe Data Dasar dalam Typescript
- Contoh Penulisan Anotasi tipe data pada:
  - Variabel
  - Parameter Function
  - Array dan Object
  - Interface
  - Properti dalam Class
- Data Modifier

Apa itu Typescript?



- Typescript adalah superset dari ECMAScript, yang berarti semua yang bisa dilakukan di Javascript pasti bisa dilakukan di Typescript.
- File typescript berekstensi **.ts**
- File typescript ini, nantinya akan dikompilasi menjadi file javascript **.js**

Instalasi & Program Typescript Pertama

- Jalankan command: `npm install -g typescript`
- Buatlah 1 file .ts. (Contoh Disini bernama index.ts)

- Compile file ts, dengan command `tsc nama_file.ts` Nantinya, berdasar kode diatas, akan tergenerate file .js, yang berisi dibawah ini:. Kamu bisa gunakan `node nama_file.js` untuk melihat hasilnya.

Tipe Data Dasar

Typescript merupakan typed-language, dimana kita harus menspesifikkan tipe data dari variable, parameter function dan properti object.

Beberapa tipe data dasar dalam Typescript:

- Number
- String
- Boolean
- Array
- Any (Boleh memasukkan tipe data apa saja)
- Dalam typescript bisa mendefinisikan lebih dari 2 tipe data dalam 1 variable/properti/params.

Referensi: https://www.tutorialsteacher.com/typescript/typescript-number

Anotasi Tipe: Mengapa Penting?

Manfaat dari menuliskan anotasi tipe ini salah satunya memudahkan debug, karena:

- Kita tidak bisa sembarang assign suatu value yang tidak sesuai tipe datanya.
- Jumlah parameter yang dimasukkan ke suatu fungsi, harus sesuai.

```
let address: string = 'Kebayoran'
address = 105 // Type 'number' is not assignable to type 'string'.
```

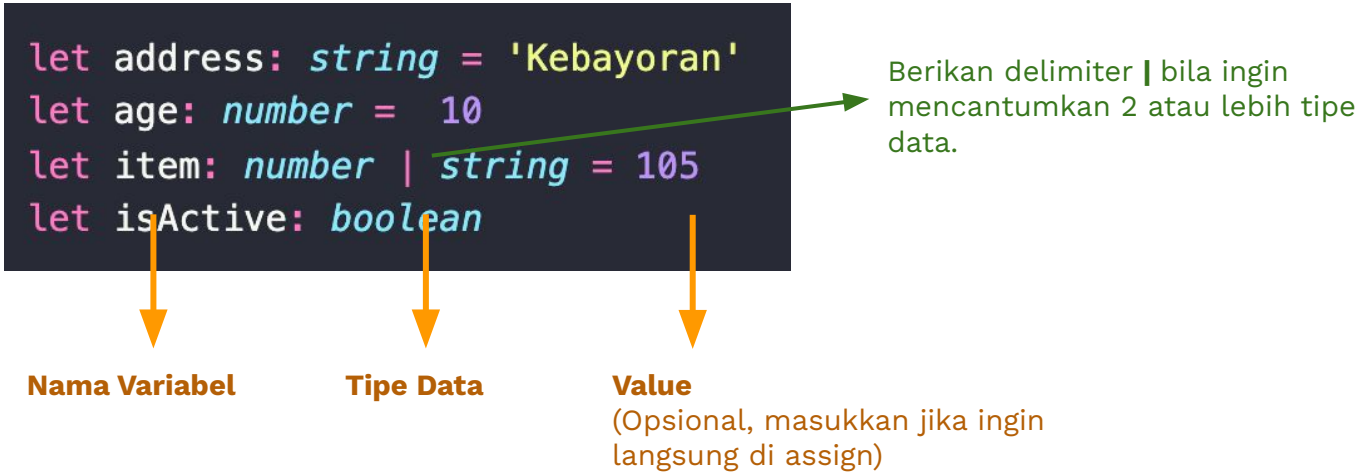
```
function getProfileTS(name:string, age: number) {
  console.log(`Hello, ${name}!, your age is ${age} old`);
}

getProfileTS('Bobi') // Expected 2 arguments, but got 1.
```

Referensi: https://www.tutorialsteacher.com/typescript/type-annotation

Cara menuliskan anotasi tipe di variabel

Kita bisa menuliskan keterangan tipe data dengan **:tipe_data** sesudah nama variabel, parameter atau property.

Contoh:



Referensi: https://www.tutorialsteacher.com/typescript/type-annotation

Cara anotasi tipe di Parameter Function

```javascript
// Javascript
function getProfile(name, age) {
  console.log(`Hello, ${name}!`);
}
getProfile('Maryam')
```

```typescript
// Typescript
function getProfileTS(name:string, age: number) {
  console.log(`Hello, ${name}!, your age is ${age} old`);
}

getProfileTS('Bobi', 12)
```

Referensi: https://www.tutorialsteacher.com/typescript/type-annotation

Cara anotasi tipe di Array

```
// Javascript
let items = ['Sepatu', 'Sendal', 'Tas']
```

```
//Typescript
let itemsTS: string[]
itemsTS= ['Sepatu', 'Sendal', 'Tas']
```

Hanya dapat meng-assign tipe data string.

```
//Typescript
let itemsTS2: (string|number|boolean)[]
itemsTS2= ['Sepatu', 'Sendal', false, 1, 2]
```

Dapat meng-assign tipe data string, number atau boolean dalam array ini.

Referensi: https://www.tutorialsteacher.com/typescript/type-annotation

Cara anotasi tipe di Object

```
// Javascript
let person = {
  name: 'Maryam',
  age: 10
}
```

```
// Typescript
let personTS : {
  name: string
  age: number
} = {
  name: 'Maryam',
  age: 10
}
```

```
// Typescript
let personTS2 : {
  name: string
  age: number | string
}

personTS2= {
  name: 'Bobi',
  age: '10'
}
```

Referensi: https://www.tutorialsteacher.com/typescript/type-annotation

Optional Property

Kita juga bisa menambahkan opsional properti..

```
// Typescript
let personTS : {
  name: string
  age: number
} = {
  name: 'Maryam'
}
```

```
Property 'age' is missing in type '{ name: string; }' but required in type
'{ name: string; age: number; }'. ts(2741)
```

Variable personTS error, karena harus mencantumkan age

Solusi:

```
// Typescript
let personTS : {
  name: string
  age?: number
} = {
  name: 'Maryam'
}
```

Tambahkan tanda **?**

di bagian definisi tipe.

Referensi: https://www.tutorialsteacher.com/typescript/type-annotation

Interface

Interface adalah sebuah kontrak, yang mana jika ada objek atau class yang mengimplementasikan interface tersebut maka objek atau class tersebut dapat menggunakan property yang dimiliki oleh interface tersebut.

Contoh deklarasi interface dengan nama Phone:

```
interface Phone {
  brand: string,
  merk: string,
  type: string,
  price: number,
  features?: string[]
}
```

Referensi: https://www.tutorialsteacher.com/typescript/typescript-interface

Interface Sebagai Anotasi Tipe

Deklarasi variabel yang mempunyai tipe interface

```
let myPhone: Phone = {
  brand: "Apple",
  merk: "Iphone",
  type: "Iphone 12 Pro",
  price: 20000000,
  features: ["iOS 14", "Super Retina Display"]
}
```

```
let phones: Phone[] = [
  {
    brand: "Apple",
    merk: "Iphone",
    type: "Iphone 12 Pro",
    price: 20000000,
    features: ["iOS 14", "Super Retina Display"]
  },
  {
    brand: "Samsung",
    merk: "Samsung",
    type: "Samsung Note Galaxy",
    price: 20000000
  }
]
```

Referensi: https://www.tutorialsteacher.com/typescript/typescript-interface

Anotasi Tipe dalam properti class

```
class Customer {
  name: string
  age: number
  money: number
  items: string[] = ['buku', 'laptop']

  constructor(name: string, age: number, money: number) {
    this.name = name
    this.age = age
    this.money = money
  }

  updateDataMoney(money: number) {
    this.money = money
  }

  addItem(itemName: any) {
    this.items.push(itemName)
  }
}

const raihan = new Customer('raihan', 12, 50000)
raihan.updateDataMoney(100000)
raihan.addItem('smartphone')
console.log(raihan)
```

Properti kelas.
Cantumkan anotasi tipe data disini.
Dan juga assign value, bila ada.

Constructor
Hanya dipanggil sekali. Ketika pembentukkan instance.
Gunakan keyword **this** untuk memanggil properti / method

Method
Merupakan function yang berada dalam class.
Gunakan keyword **this,** jika ingin memanggil properti atau method lain.

Referensi: https://www.tutorialsteacher.com/typescript/typescript-class

Anotasi Tipe dalam properti class

```
class Shop {
  customers: Customer[] = []

  addCustomer(newCustomer: Customer) {
    this.customers.push(newCustomer)
  }
}

const shopE = new Shop()
shopE.addCustomer(raihan)
console.log(shopE);
```

Class juga bisa dijadikan anotasi tipe.

Referensi: https://www.tutorialsteacher.com/typescript/typescript-class

Data Modifier

Dalam Object-Oriented Programming, terdapat konsep enkapsulasi, dimana kita bisa mengontrol visibilitas dari suatu data dalam class.

1. **Public**: Membuat data dapat diakses oleh siapa saja, dan di scope mana saja.
2. **Private**: Membuat data hanya dapat diakses dalam class miliknya saja.
3. **Protected**: Membuat data hanya dapat diakses dalam class miliknya dan turunannya.

Referensi: https://www.tutorialsteacher.com/typescript/data-modifiers

Data Modifier - Public

```
class Employee {
  public code: string;

  constructor(public name: string) {
    this.name = name
  }

  getCode() {
    return this.code
  }
}

class SalesEmployee extends Employee {
  getSalesCode() {
    return 'sales '+ this.name + this.code
  }
}

let emp = new Employee('Mawar');
emp.code = "123";
console.log(emp.name)
```

Properti dengan data modifier **public**, bisa diakses dimanapun

Referensi: https://www.tutorialsteacher.com/typescript/data-modifiers

Data Modifier - Private

```
class Employee {
  private code: string;

  constructor(private name: string) {
    this.name = name
  }

  getCode() {
    return this.code
  }
}

class SalesEmployee extends Employee {
  getSalesCode() {
    return 'sales '+ this.name + this.code
  }
}

let emp = new Employee('Mawar');
emp.code = "123";
console.log(emp.name)
```

Properti dengan data modifier **private**, bisa diakses di dalam class miliknya.

Properti dengan data modifier **private**, tidak bisa diakses di class inherit dan diluar class.

Referensi: https://www.tutorialsteacher.com/typescript/data-modifiers

Data Modifier - Protected

```
class Employee {
  protected code: string;

  constructor(protected name: string) {
    this.name = name
  }

  getCode() {
    return this.code
  }
}

class SalesEmployee extends Employee {
  getSalesCode() {
    return 'sales '+ this.name + this.code
  }
}

let emp = new Employee('Mawar');
emp.code = "123";
console.log(emp.name)
```

Properti dengan data modifier **protected**, bisa diakses di dalam class miliknya dan class inherit.

Properti dengan data modifier **protected**, tidak bisa di luar class miliknya/class inherit.

Referensi: https://www.tutorialsteacher.com/typescript/data-modifiers