

Giới thiệu về Typescript

Đỗ Thành Long

dtlong@opengis.vn

<https://opengis.vn>



OPENGIS
Discover the world, Learn with maps

 <https://opengis.vn>

1. Lộ trình



- Buổi 1: Giới thiệu Ionic Framework và GIS mobile.
- Buổi 2: Ôn tập HTML, CSS.
- Buổi 3: JavaScript, Overview Ionic.
- **Buổi 4: Giới thiệu TypeScript (hôm nay).**
- Buổi 5: Giới thiệu Angular và NgModules.

Tại sao lại học TypeScript???



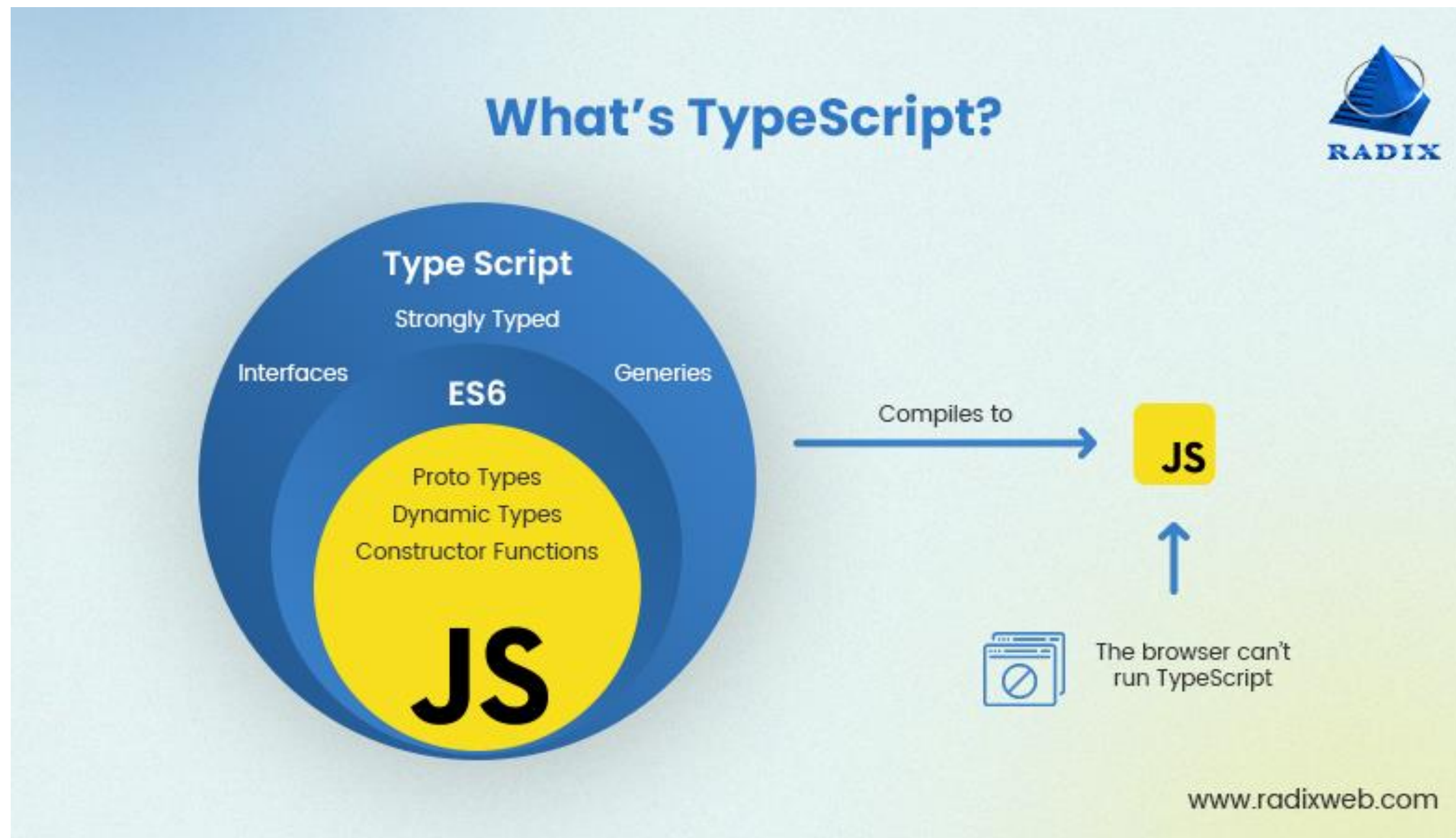
2. TypeScript là gì?



- TypeScript là ngôn ngữ lập trình mã nguồn mở, do Microsoft phát triển (2012).
- Là **siêu tập (superset)** của JavaScript: mọi mã JavaScript đều là TypeScript.

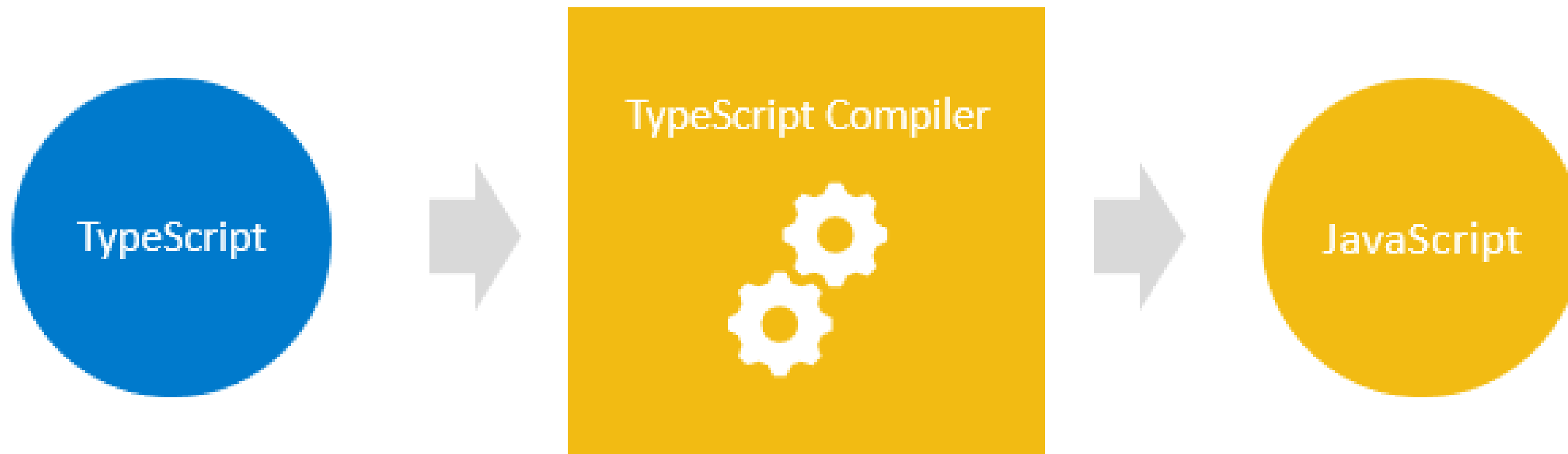


2. TypeScript là gì?



Là **siêu tập (superset)** của JavaScript: mọi mã JavaScript đều là TypeScript.

2. TypeScript là gì?



2. TypeScript là gì?

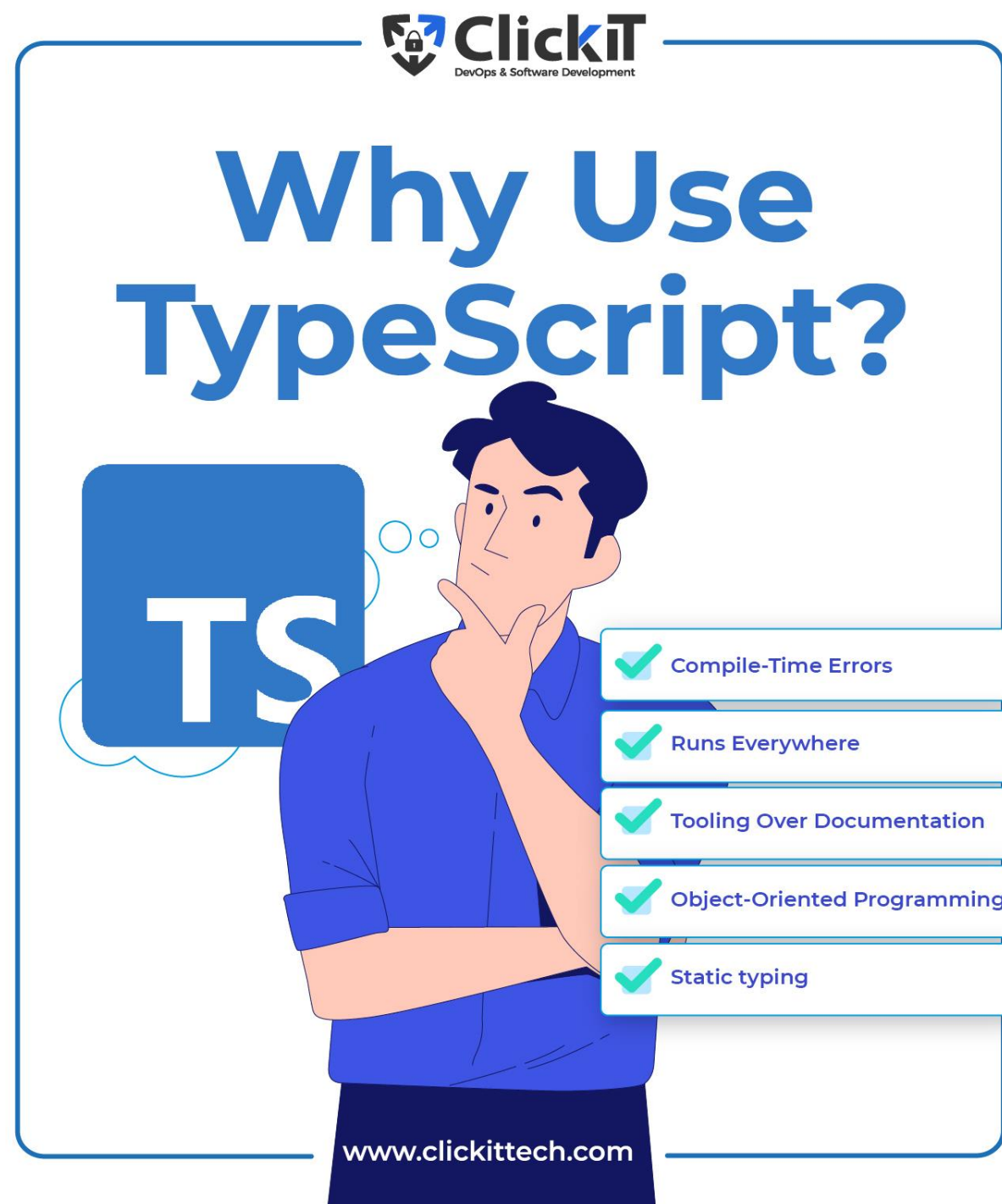


- Thêm **kiểu dữ liệu tĩnh (static typing)** vào JavaScript.
- Biên dịch thành JavaScript bằng tsc để chạy trên trình duyệt hoặc Node.js.

typescript

```
let message: string = "Hello, TypeScript!";  
console.log(message);
```


3. Tại sao dùng TypeScript?

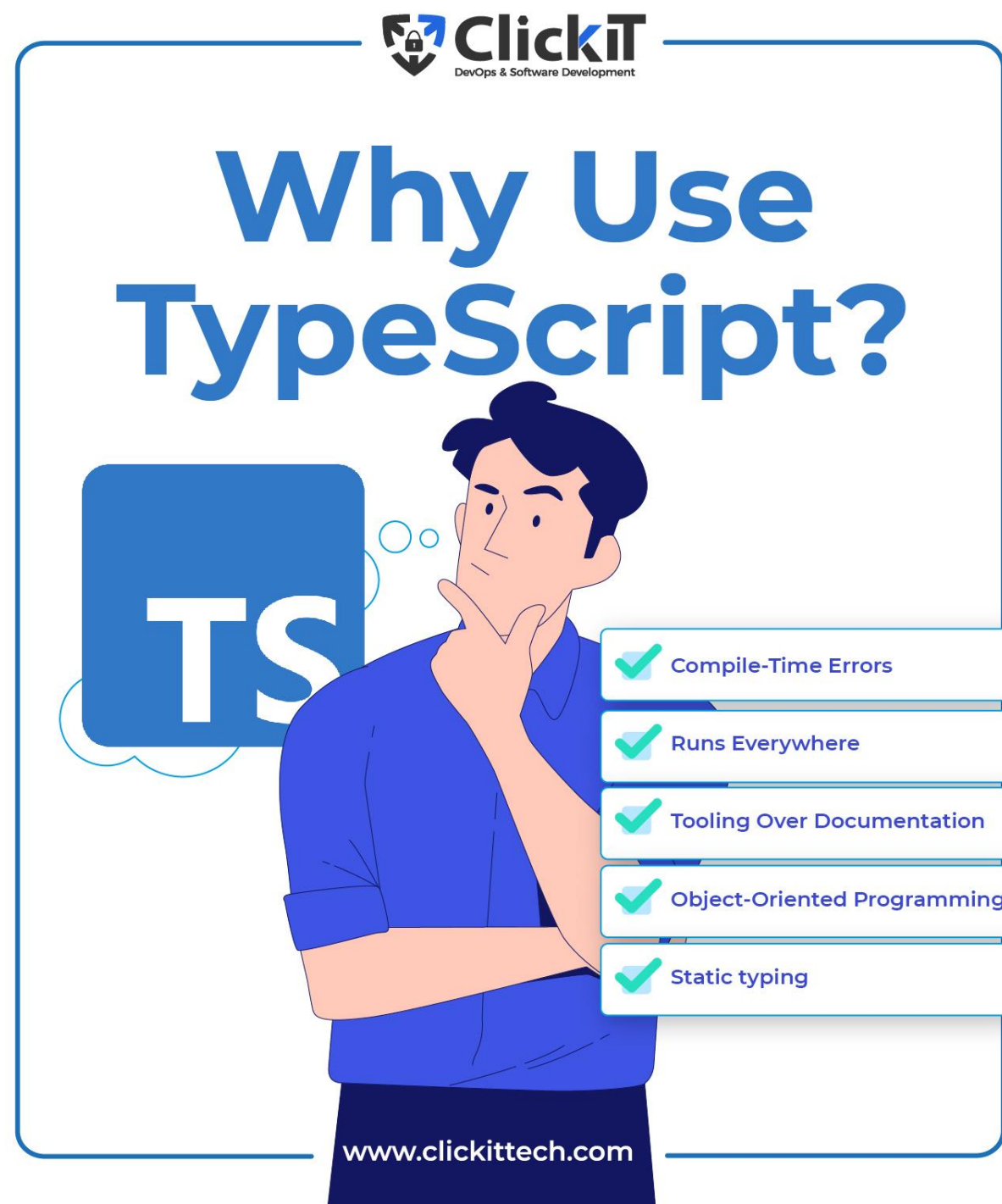


Phát hiện lỗi sớm: Kiểm tra lỗi khi viết code thay vì khi chạy.

Dễ bảo trì: Mã nguồn rõ ràng, dễ đọc nhờ kiểu dữ liệu.


Hỗ trợ dự án lớn: Phù hợp với ứng dụng phức tạp như GIS mobile.

3. Tại sao dùng TypeScript?




Ionic dùng TypeScript để tích hợp với Angular.
Đảm bảo ứng dụng GIS mobile (quản lý tọa độ, bản đồ) hoạt động chính xác.

3. Tại sao dùng TypeScript?



Why Use TypeScript?



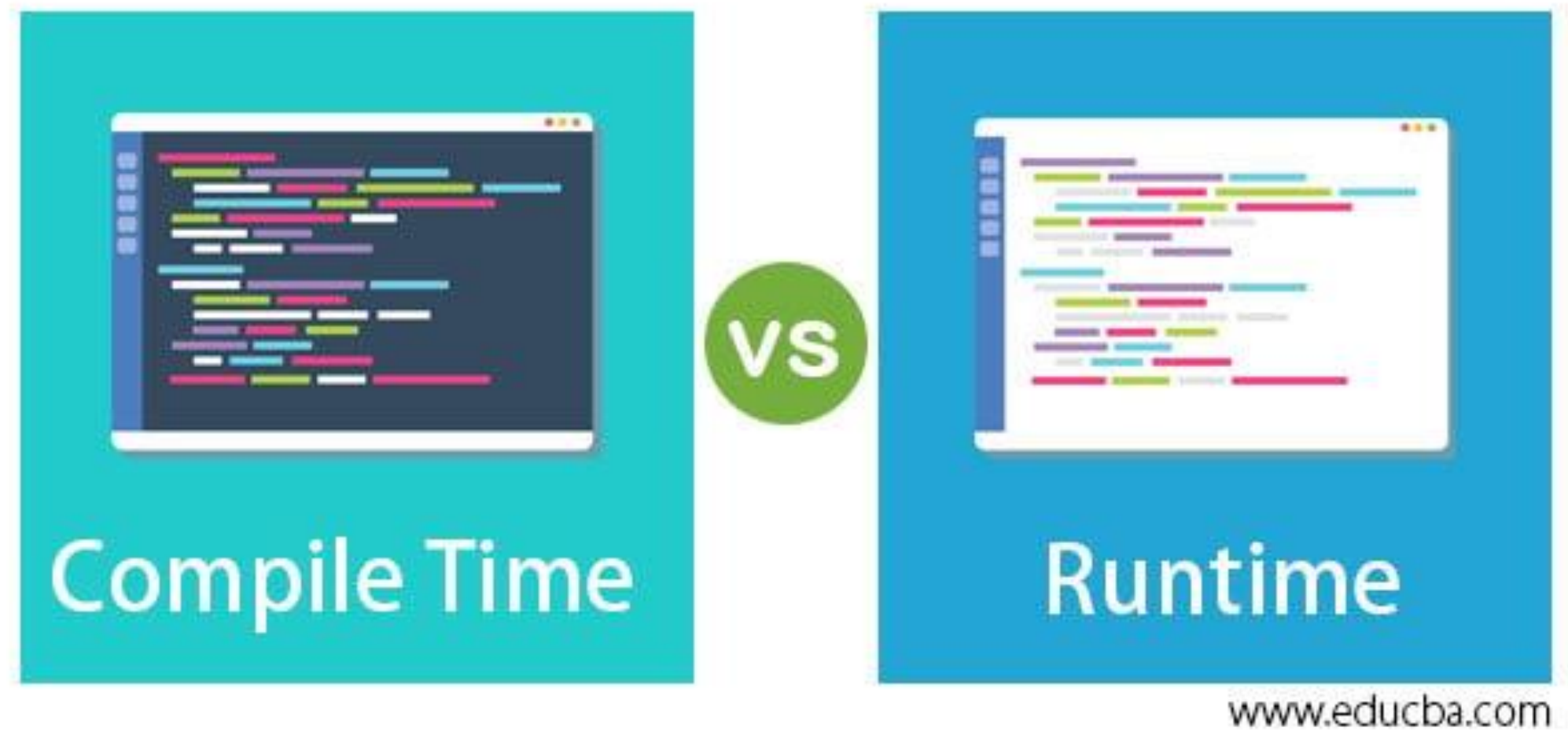
- ✓ Compile-Time Errors
- ✓ Runs Everywhere
- ✓ Tooling Over Documentation
- ✓ Object-Oriented Programming
- ✓ Static typing

www.clickittech.com



Cách xử lý lỗi

3. Tại sao dùng TypeScript?



- Xảy ra trong quá trình biên dịch mã nguồn thành mã máy.
- Thường là lỗi cú pháp, lỗi kiểu dữ liệu, lỗi logic cơ bản.
- Dẫn đến việc không thể hoàn thành quá trình biên dịch và không tạo ra file thực thi.
- Lỗi sẽ được phát hiện và báo cáo khi biên dịch mã nguồn.

- Xảy ra khi chương trình đang chạy và gặp phải tình huống không xử lý được.
- Thường là lỗi logic phức tạp, truy cập vào vùng nhớ không hợp lệ, chia cho 0,...
- Dẫn đến việc chương trình dừng hoạt động, crash hoặc hiện thông báo lỗi cho người dùng.
- Lỗi thường không được phát hiện trước khi chương trình chạy.

3. Tại sao dùng TypeScript?

javascript



```
function add(a, b) { return a + b; }  
add(5, "10"); // "510" (lỗi không mong muốn)
```

4. Khác biệt so với Javascript

- Kiểu dữ liệu:

- **JavaScript:** Không khai báo kiểu (dynamically typed).

```
javascript
```

```
let x = 5; x = "text"; // Hợp lệ nhưng nguy hiểm
```

- **TypeScript:** Yêu cầu khai báo kiểu (statically typed).

```
typescript
```

```
let x: number = 5; x = "text"; // Lỗi biên dịch
```

- Ý nghĩa: TypeScript giúp tránh lỗi do thay đổi kiểu không mong muốn.

Kiểu dữ liệu

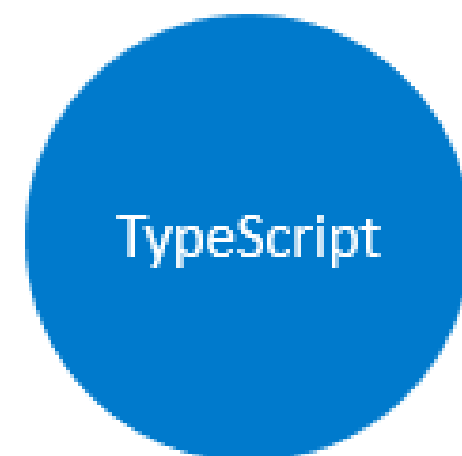
4. Khác biệt so với Javascript

- JavaScript: Chạy trực tiếp.
- TypeScript: Biên dịch bằng `tsc`.

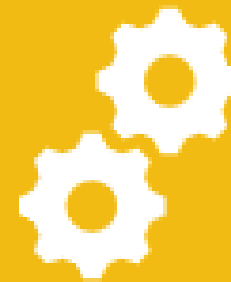
```
bash
```

```
tsc file.ts # Tạo file.js  
node file.js
```

Thực thi



TypeScript Compiler



JavaScript

TypeScript lý tưởng cho dự án lớn như Ionic

4. Biến trong TypeScript

- Khai báo biến:

- Dùng `let` (có thể thay đổi) hoặc `const` (cố định).
- Thêm kiểu bằng `: kiểu_dữ_liệu`.

- Kiểu cơ bản:

- `number`: Số (10, 3.14).
- `string`: Chuỗi ("Hello").
- `boolean`: True/False.
- `any`: Bất kỳ kiểu nào (ít dùng).

kiểu dữ liệu tĩnh
(static typing)



4. Biến trong TypeScript

typescript

```
let age: number = 25;  
const name: string = "John";  
let isActive: boolean = true;  
let data: any = 10; data = "text";
```

kiểu dữ liệu tĩnh
(static typing)



5. Kiểu dữ liệu nâng cao

Mảng

◦ Array:

- Cách 1: `kiểu[]`.
- Cách 2: `Array<kiểu>`.

typescript

```
let numbers: number[] = [1, 2, 3];  
let names: Array<string> = ["John", "Jane"];
```

5. Kiểu dữ liệu nâng cao

Đối tượng

- **Interface:** Định nghĩa cấu trúc đối tượng.

typescript

```
interface Point {  
  x: number;  
  y: number;  
}  
let point: Point = { x: 1, y: 2 };
```

5. Kiểu dữ liệu nâng cao

- **Union Type:** Nhiều kiểu cho một biến.

typescript



```
let id: number | string = 123; id = "ABC";
```

6. Hàm trong TypeScript

- **Cú pháp cơ bản:**

- Khai báo kiểu cho tham số và giá trị trả về.

typescript



```
function add(a: number, b: number): number {  
    return a + b;  
}  
console.log(add(3, 4)); // 7
```

6. Hàm trong TypeScript

- Tham số tùy chọn: Dùng `?`.

typescript

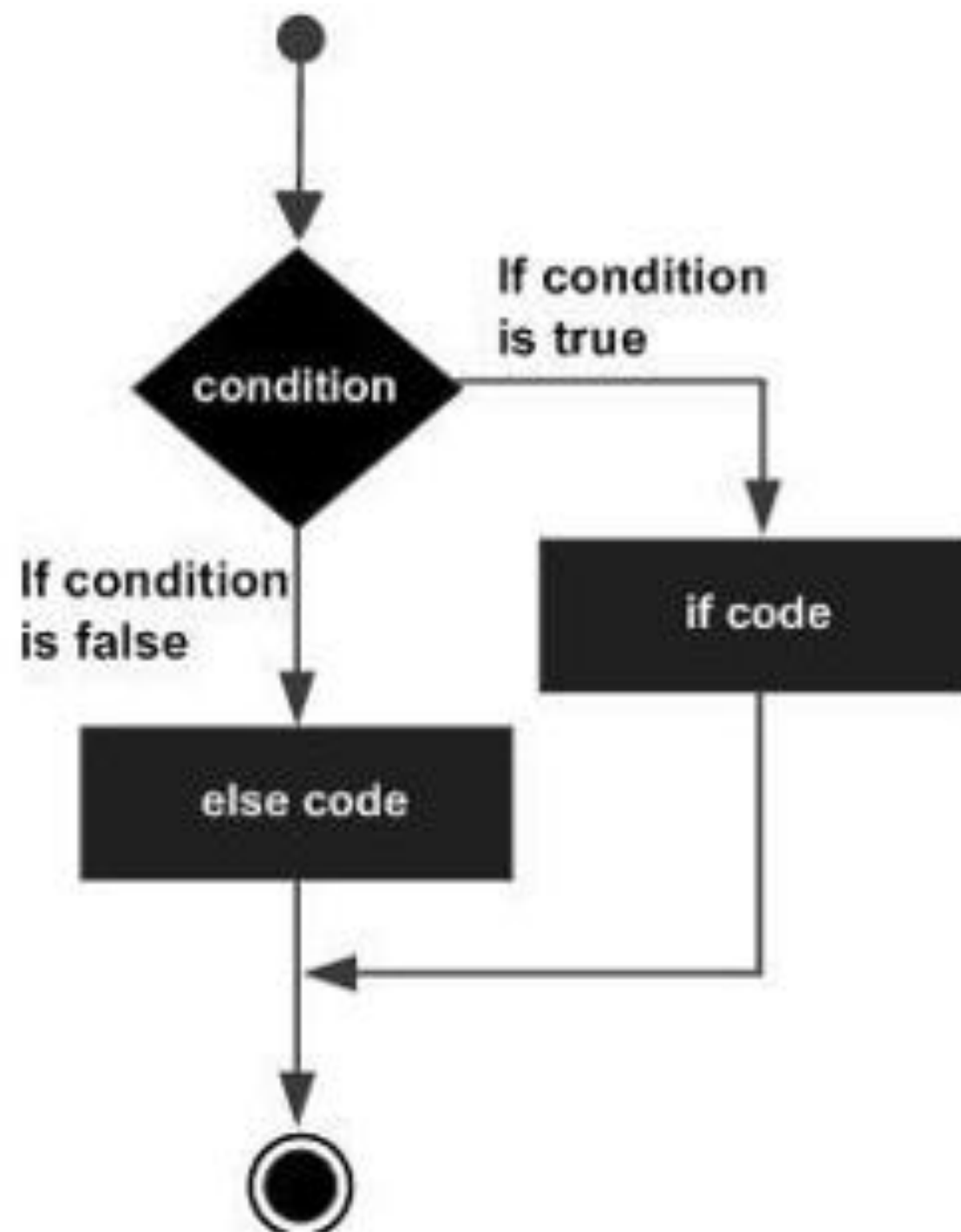
```
function greet(name: string, greeting?: string): string {  
    return `${greeting || "Hello"}, ${name}`;  
}
```

- Tham số mặc định:

typescript

```
function sayHi(name: string = "Guest"): string {  
    return `Hi, ${name}`;  
}
```


7. Cấu trúc điều khiển - If-else



◦ Cú pháp:

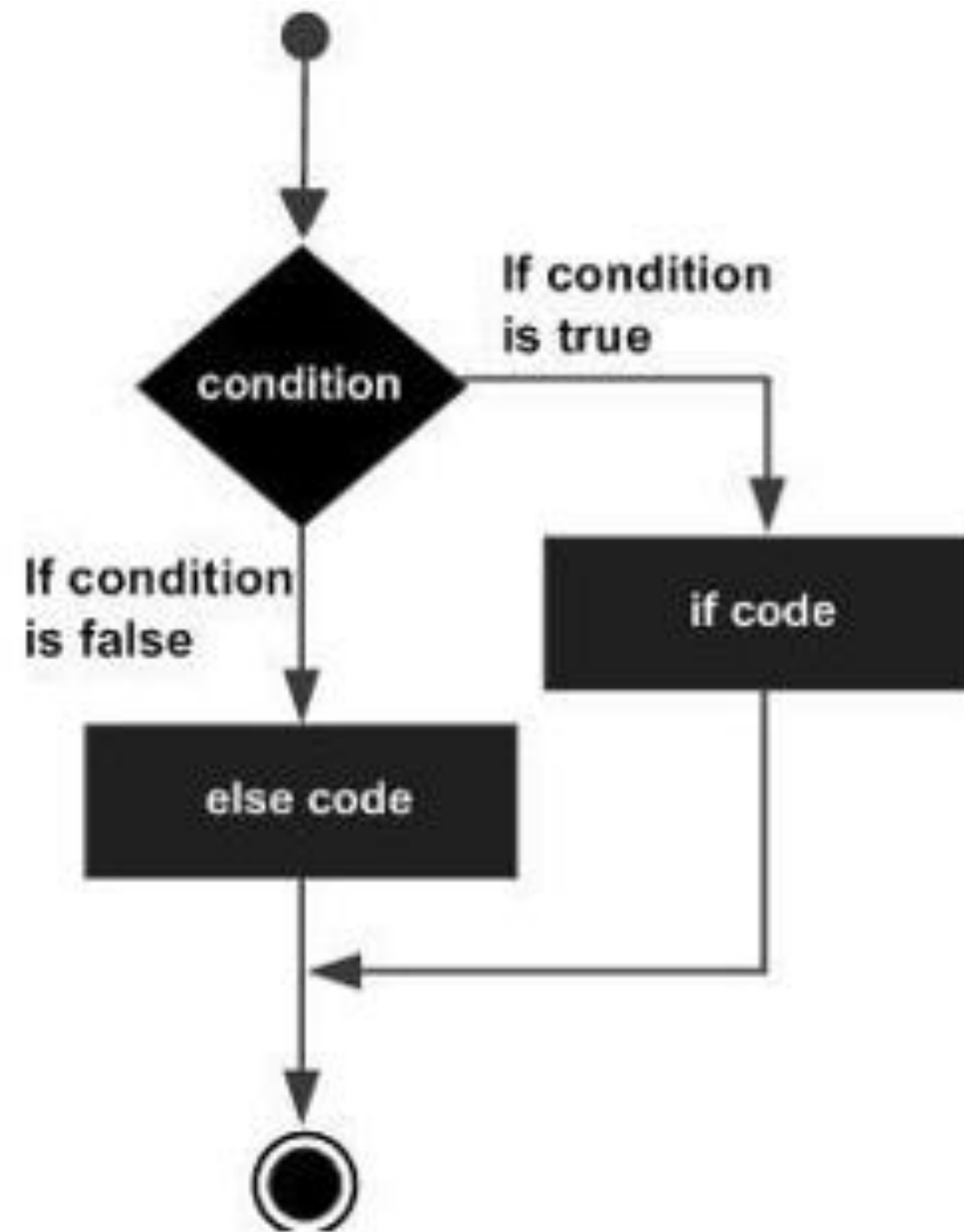
```
typescript

function checkScore(score: number): string {
  if (score >= 8) {
    return "Giỏi";
  } else if (score >= 5) {
    return "Trung bình";
  } else {
    return "Yếu";
  }
}
```

◦ Ý nghĩa: Ra quyết định dựa trên điều kiện.

◦ Ví dụ: `console.log(checkScore(7)); // "Trung bình"`

7. Cấu trúc điều khiển - If-else



- Kiểm tra tọa độ có hợp lệ không ($x, y > 0$).

typescript

```
function isValidPoint(x: number, y: number): boolean {  
  if (x >= 0 && y >= 0) return true;  
  return false;  
}
```

8. Cấu trúc điều khiển - Switch

- **Cú pháp:**

typescript

```
function getDay(day: number): string {  
  switch (day) {  
    case 1: return "Thứ Hai";  
    case 2: return "Thứ Ba";  
    default: return "Không xác định";  
  }  
}
```

- **Ý nghĩa:** Thay thế nhiều if-else khi so sánh giá trị cụ thể.

8. Cấu trúc điều khiển - Loop

- For Loop:

typescript



```
for (let i: number = 0; i < 5; i++) {  
    console.log(`Số: ${i}`);  
}
```

8. Cấu trúc điều khiển – While Loop

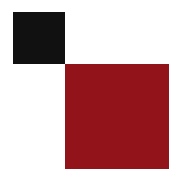
◦ While Loop:

typescript

```
let count: number = 0;
while (count < 3) {
  console.log(`Đếm: ${count}`);
  count++;
}
```



OPENGIS



THANK YOU

