GIỚI THIỆU VỀ TypeScript

TITV

Lập trình React

- JavaScript: Như đã đề cập, JavaScript là ngôn ngữ chuẩn dùng cho phát triển React. Bạn có thể sử dụng cả JavaScript hiện đại ES6+ và phiên bản cũ hơn của JavaScript (ES5) với React.

Lập trình React

- Dart: là một ngôn ngữ được phát triển bởi Google. Mặc dù thường liên quan đến Flutter cho phát triển ứng dụng di động, bạn có thể sử dụng nó cùng với React thông qua các thư viện như "react-dart."

Lập trình React

- ECMAScript là một chuẩn ngôn ngữ lập trình được sử dụng để định nghĩa ngôn ngữ kịch bản (scripting language), và nó là cơ sở cho các ngôn ngữ lập trình như JavaScript.
- (ES6, ES9, ...)

Lập trình React

- TypeScript: TypeScript là một phần mở rộng kiểu tĩnh của JavaScript. Nó thêm kiểu tĩnh vào mã JavaScript của bạn, có thể cải thiện chất lượng và dễ bảo trì mã.

Giới thiệu về TypeScript

- TypeScript là một ngôn ngữ lập trình xây dựng dựa trên JavaScript
- TypeScript có cú pháp rất giống với JavaScript
- Được Mictosoft phát triển từ 2012
- Mã nguồn mở và miễn phí

```javascript
function calculateSum(a, b) {
    return a + b;
}

console.log(calculateSum(5, 10));  // Kết quả: 15
console.log(calculateSum("5", 10));  // Kết quả: "510"
```

```typescript
function calculateSum(a: number, b: number): number {
    return a + b;
}

console.log(calculateSum(5, 10));  // Kết quả: 15
// Dòng sau sẽ báo lỗi biên dịch vì kiểu dữ liệu không ph
// console.log(calculateSum("5", 10));
```

Một số điểm bổ sung

- Kiểu tĩnh: TypeScript cho phép bạn khai báo kiểu cho các biến, hằng số, hàm và đối tượng. Điều này giúp cải thiện độ chính xác và hiệu suất của mã.

```
let myNumber: number = 10;
let myString: string = "Hello, world!";

function addNumbers(a: number, b: number): number {
  return a + b;
}

console.log(addNumbers(1, 2)); // 3
```

Một số điểm bổ sung

- Interface: Interface là một cách để mô tả các giao diện của các đối tượng. Interface có thể được sử dụng để xác định các phương thức và thuộc tính mà một

```
interface IAnimal {
  makeSound(): void;
}

class Dog implements IAnimal {
  makeSound() {
    console.log("Woof!");
  }
}

let dog: IAnimal = new Dog();
dog.makeSound(); // Woof!
```

Một số điểm bổ sung

- Generics: Generics là một cách để viết mã có thể được sử dụng với nhiều loại dữ liệu khác nhau.

```
function identity<T>(arg: T): T {
  return arg;
}

let output1 = identity<string>("myString");
let output2 = identity<number>(100);

console.log(output1); // "myString"
console.log(output2); // 100
```

Một số điểm bổ sung

- Modules: TypeScript hỗ trợ các module để tổ chức mã một cách hợp lý.

```ts
// file: my-module.ts

export function addNumbers(a: number, b: number): number {
  return a + b;
}
```

```ts
// file: my-app.ts

import { addNumbers } from "./my-module";

const result = addNumbers(1, 2);
console.log(result); // 3
```

Một số điểm bổ sung

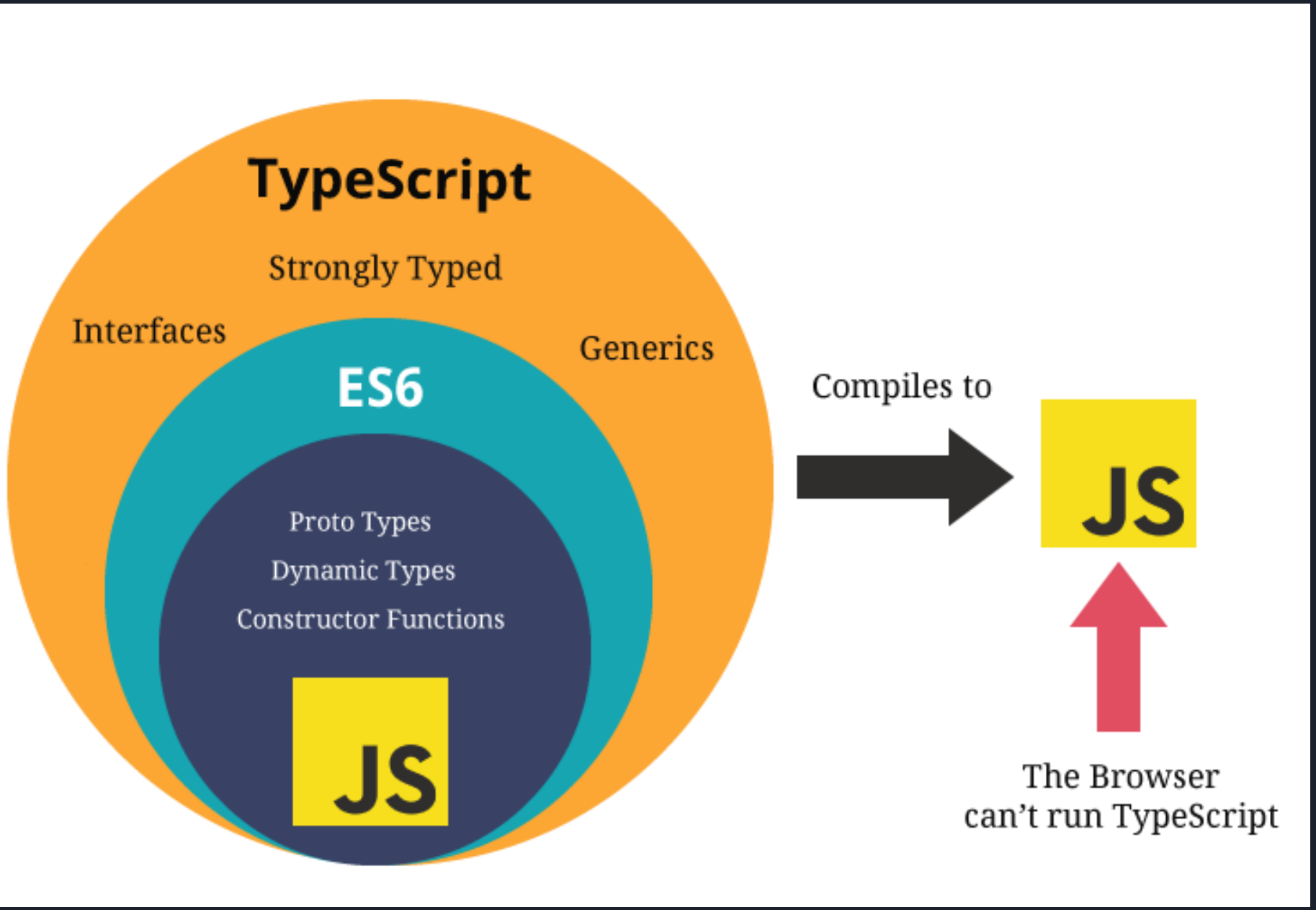
- Decorators: TypeScript hỗ trợ các decorators để thêm các chức năng bổ sung cho các lớp, hàm và thuộc

```
// file: my-decorator.ts

function logDecorator(target: any, key: string, descriptor: PropertyDescriptor) {
  console.log(`Decorated property: ${key}`);
}

// file: my-app.ts

@logDecorator
class MyClass {
  public name: string;
}

const myClass = new MyClass();
myClass.name = "John Doe";
```

Định dạng

- File TypeScript có định dạng .ts

```ts
// myFile.ts
function greet(name: string) {
  console.log(`Hello, ${name}!`);
}

const personName: string = "John";
greet(personName);
```



https://github.com/castellanosfelipe/Class-examples-Javascript-and-typescript.