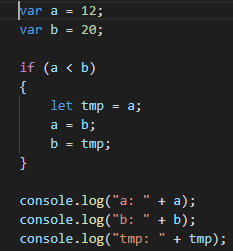
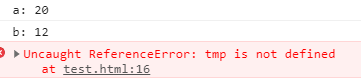
**Tổng hợp kiến thức es6**

1. **Block scoped let, const**
   1. **Block scoped**

* Là phạm vi trong một khối, nghĩa là chỉ hoạt động trong phạm vi được khai báo bởi {}
  1. **Let**
* Dùng để khai báo biến chỉ có tác dụng trong khối khai báo
* Cú pháp: let var\_name = var\_value
* Let thường dùng để khai báo biến mang tính tạm thời



Kết quả:



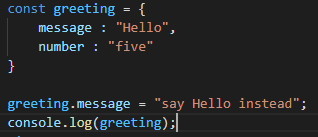
* 1. **Const**
* Biến này tương tự let, cũng có scope là block scoped
* Biến được khai báo kiểu này không thể tái khai báo hay cập nhật giá trị mới



Kết quả:



* Đối với object được khai báo kiểu const thì tuy không thể tái khai báo hay cập nhật giá trị của biến nhưng vẫn có thể cập nhật giá trị thuộc tính của biến đó

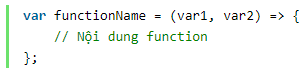


Kết quả:

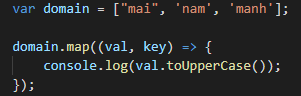


1. **Arrow function**

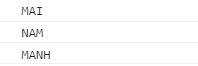
* Cú pháp:



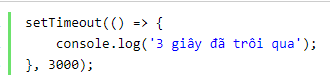
* Trường hợp trong thân hàm chỉ có một lệnh duy nhất thì có thể bỏ dấu {}
* Trường hợp truyền vào chỉ một tham số thì có thể bỏ cặp ()
* Trường hợp không có tham số thì dùng cặp () rỗng
* Một số ví dụ arrow function
* Với hàm map:



Kết quả:



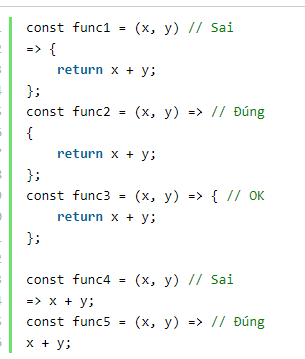
* Với hàm setTimeout:



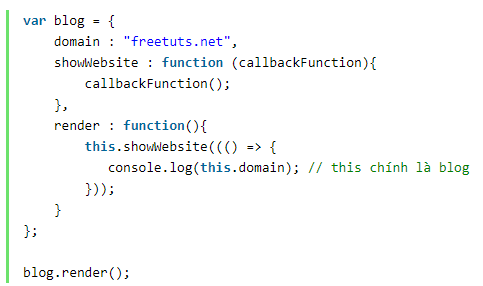
* Lỗi cú pháp với Arrow function
* Đóng arrow function



* Ràng buộc mũi tên

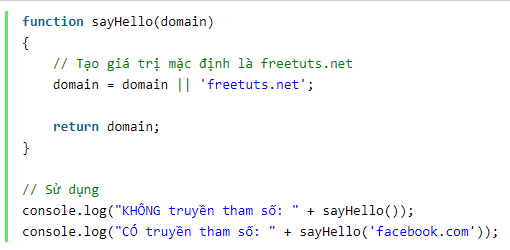


* Dùng arrow function khắc phục nhược điểm phạm vi hoạt động của đối tượng this trong closure function

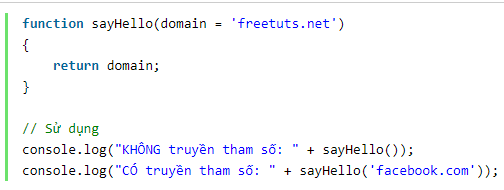


1. **Default Parameter**

* Là giá trị mặc định của tham số khi truyền vào các function
* Ví dụ:
* Trong ES5 để tạo giá trị mặc định thì sử dụng cặp dấu ||



* Trong ES6 thì gán ngay tại vị trí khai báo

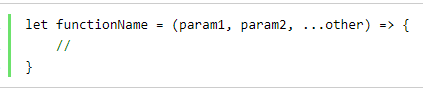


Kết quả cả hai đều giống nhau:



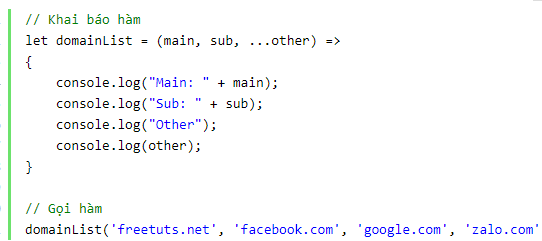
1. **Rest**

* Dịch theo tiếng anh thì rest parameter nghĩa là tham số còn lại, có nghĩa là có thể khai báo một hàm với số lượng tham số không xác định
* Khai báo:

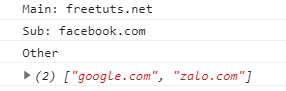


Ở đây biến other chính là tham số còn lại

* Ví dụ:



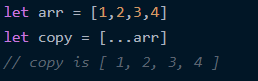
Kết quả:



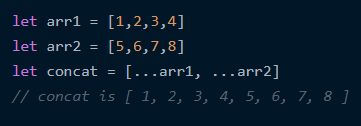
* Nếu tham số truyền vào vừa đủ thì rest parameter sẽ có giá trị là mảng rỗng

1. **Spread**

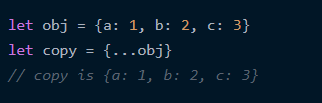
* Spread operator cho phép chuyển đổi một chuỗi thành nhiều argument (trong trường hợp gọi với hàm) hoặc thành nhiều phần tử (cho array). Bên cạnh đó nó cũng cho phép làm nhiệm vụ destructure
* Operator này có syntax là 3 dấu chấm …
* Các trường hợp dùng Spread operator
* Copy một array



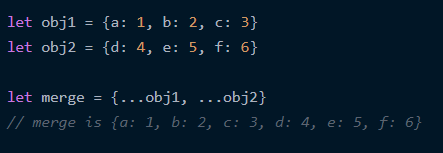
* Nối array



* Copy một object



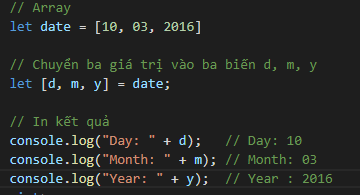
* Kết hợp object



* Dùng để loop trong logging

1. **Destructuring Assignment**

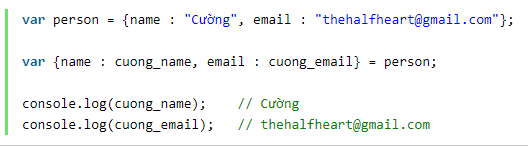
* Là cách tách các phần tử của Array hoặc Object thành nhiều biến chỉ bằng một đoạn code duy nhất



Kết quả:



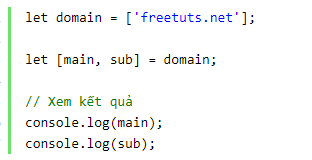
* Một số ví dụ nâng cao khác
* Lấy giá trị theo key object



Kết quả:



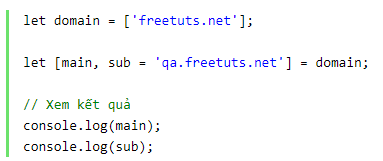
* Lỗi undefined



Kết quả:



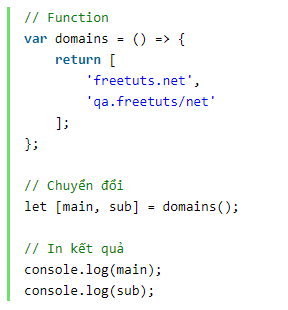
* Giá trị mặc định



Kết quả:



* Sử dụng function



Kết quả:



1. **Template Literals**

* Là một cú pháp mới dùng để khai báo biến. Khi dùng cú pháp này thì:
* Khai báo biến nhiều dòng một cách đơn giản
* Binding biến vào template dễ dàng
* Cú pháp khai báo: Thay dấu ‘ hoặc “ bằng `



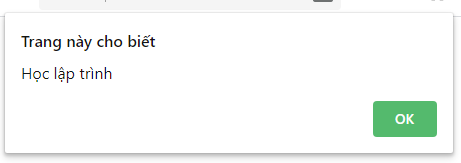
* Cú pháp binding



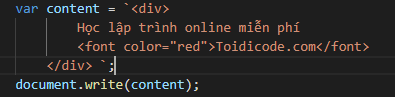
* Một số ví dụ
* Khai báo biến dưới dạng template literals



Kết quả:



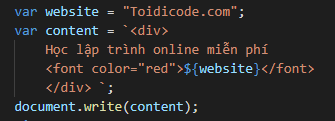
* Khai báo biến nhiều dòng



Kết quả:



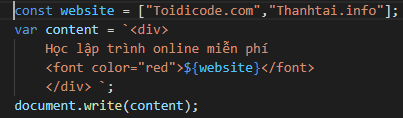
* Binding giá trị biến vào template literals



Kết quả:



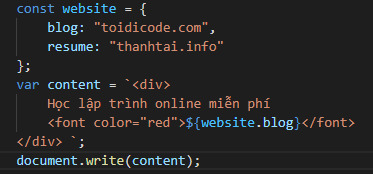
* Binding giá trị mảng vào template literals



Kết quả:



* Binding giá trị object vào template literals



Kết quả:

