*//khởi tạo hàm*

*//!declaration function*

*//todo hàm Declaration có hỗ trợ hoisting*

function sum(){

     console.log(1+2);

}

*//!expression function*

var sum1 = () =>{

     console.log(3+4);

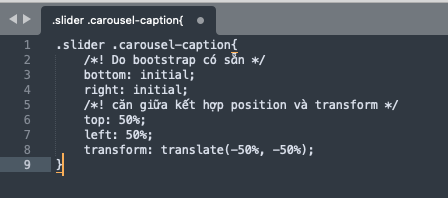
}

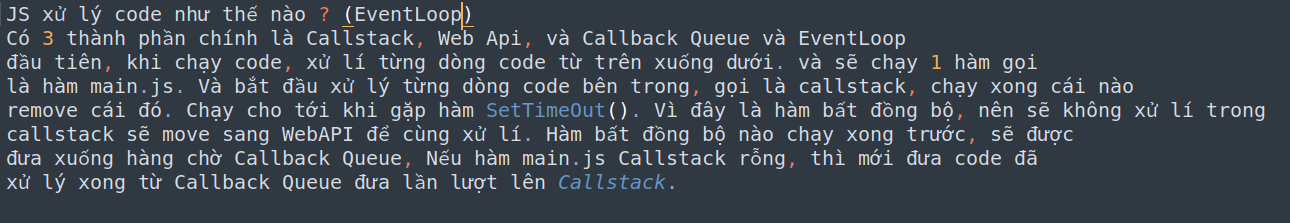
*//! hoisting là log ra trước khi khai báo*

console.log(a);

var a = 5;

console.log(a);

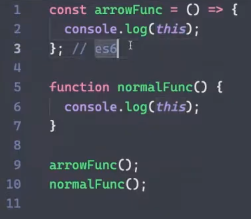




X) Arrow function và Normal function khác nhau chỗ nào

**Nghĩ gì về context ( ngữ cảnh)**

**Em biết gì về con trỏ this trong javascript**



this ở arrow sẽ trỏ đến nơi thực thi ArrowFunc đó

this ở normal sẽ trỏ đến nơi gọi function đó (window)

X) Var, Const và let khác nhau chỗ nào

- Var có thể khai báo 1 biến nhiều lần.

**Khác nhau về cơ chế Hoisting:**

* Var cho phép Hoisting ( ngược lại Let và Const thì không)

**Khác nhau về cơ chế Scope:**

* Var phạm vi sử dụng là Local Scope và Global
* 2 cái còn lại là sử dụng trong block Scope

X) Closure

*Tập hợp 1 function và môi trường khởi tạo function đó, cho phép truy cập vào biến cảu function cha.*

*Ví dụ khi muốn truyền tham số vào event onClick*

*{this.showMessage(“Hello”)}*

handleChangeColor = (*image*) => {

*//! đây là clorsure*

return () => {

this.setState({

imgSource: *image*,

});

}

}

<button *onClick*={this.handleChangeColor(redCar)} *className*="btn btn-danger">Red</button>

<button *onClick*={this.handleChangeColor(silverCar)} *className*="btn btn-secondary">Silver</button>

<button *onClick*={this.handleChangeColor(blackCar)}

Khi truyền event có params thì phải bọc 1 closure

X) setState là 1 phương thức bất đồng bộ, muốn kiểm tra setState đúng thì phải làm 1 hàm callback gọi ở bên trong mới kiêm tra được

