**Hằng (constant)**

**const** PI **=** 3.14

Hằng giúp tạo các biến không thể thay đổi giá trị (immutable variables). Lưu ý rằng hằng chỉ ngăn chặn việc thay đổi giá trị của biến chứ không ngăn chặn thay đổi nội dung của biến, ví dụ như đổi 1 thuộc tính (property) của 1 object.

**Biến giới hạn phạm vi (Block-Scoped Variables)**

**function** varTest() {

**let** x **=** 1

**if** (**true**) {

**let** x **=** 2

console.log(x) *// 2*

}

console.log(x) *// 1*

}

Trong ví dụ trên, biến x bên ngoài và trong block if được xem là 2 biến khác nhau nếu khai báo biến dùng let. Nếu dùng var khai báo biến thì 2 biến x trên được xem là 1 biến và giá trị in ra tất cả bằng 2.

Biến giới hạn phạm vi là 1 bước tiến lớn của JS, được khuyến cáo sử dụng thay thế var, chúng giúp cho việc sử dụng biến và debug dễ dàng hơn khi ta biết được phạm vi hoạt động của từng biến một cách dễ dàng.

**Arrow (Fat Arrow) function**

**let** odds **=** evens.map(v **=>** v **+** 1)

Tương đương với

**let** odds **=** evens.map(**function**(v){

**return** (v **+** 1)

});

var chuoi1 = "Hello";

var chuoi2 = "TDC-Cao đẳng công nghệ";

function soSanh()

{

if(chuoi1 > chuoi2)

{

console.log("chuỗi 1 dài hơn chuỗi 2")

}

else

{

console.log("chuỗi 1 ngắn hơn chuỗi 2")

}

}

soSanh();

function noiChuoi()

{

var chuoi = "Welcome";

var chuoi1 = " to TDC";

chuoi = chuoi.concat(chuoi1, "!!!");

console.log("Kết quả sau khi nối chuỗi: " +chuoi);

}

noiChuoi();

function timKiemThayThe()

{

var chuoi = chuoi1 + chuoi2;

console.log(chuoi.replace("TDC-Cao đẳng công nghệ"," Trường cao đẳng công nghệ thủ đức"));

}

timKiemThayThe();

function viTri()

{

console.log("vị trí xuất hiện chuỗi: " + chuoi2.indexOf("Cao"));

}

viTri();

