

LẬP TRÌNH JAVASCRIPT DÀNH CHO NODESJS

1. Khai báo biến, hằng:

```
var x = 5;  
var y = 6;  
var z = x + y;  
const pi = 3.14;
```

2. Toán tử:

a. Toán tử số học

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

b. Toán tử khác

Operator	Example	Same As
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$

3. Các phép toán so sánh:

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator



4. Toán tử logic:

Operator	Description
&&	logical and
 	logical or
!	logical not

5. Kiểu dữ liệu

```
var length = 16;           // Number
var lastName = "Johnson"; // String
var x = {name:"John", age:22}; // Object
```

```
typeof ""           // Returns "string"
typeof "Joh         // Returns "string"
typeof 3.14         // Returns "number"
typeof (3 + 4)      // Returns "number"
typeof true         // Returns "boolean"
typeof false        // Returns "boolean"
typeof x            // Returns "undefined" (if x has no value)
```

a. undefined

```
var car;           // Value is undefined, type is undefined
car = undefined;   // Value is undefined, type is undefined
```

b. Empty

```
var car = ""; // The value is "", the typeof is "string"
```

c. Null

```
var person = {firstName:"John", lastName:"Doe", age:50};
person = null; //value is null, type is an object
```

```
var person = {firstName:"John", lastName:"Doe", age:50};
person = undefined; // both value and type is
undefined
```

```
typeof undefined // undefined
typeof null      // object
```

6. Functions (Hàm)


```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Ví dụ:

```
var x = myFunction(4, 3); // Function is called,
result is 12
```

```
function myFunction(a, b) {
  return a * b;
}
```

7. Objects (Đối tượng)

Object	Properties	Methods
	<p>car.name = Fiat</p> <p>car.model = 500</p> <p>car.weight = 850kg</p> <p>car.color = white</p>	<p>car.start()</p> <p>car.drive()</p> <p>car.brake()</p> <p>car.stop()</p>

a. Khai báo:

```
var car = {type:"Fiat", model:"500", color:"white"};
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue",
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

b. Truy cập thuộc tính

objectName.propertyName

hoặc

objectName["propertyName"]

c. Truy cập phương thức

objectName.methodName()

8. Hàm xử lý chuỗi

a. Đếm chiều dài chuỗi

```
var txt = "ABCDEFGHJKLMNOPQRSTUVWXYZ";  
var sln = txt.length;
```

b. Tìm kí tự trong chuỗi

```
var str = "Please locate where 'locate' occurs!";  
var pos = str.indexOf("locate");
```

```
var str = "Please locate where 'locate' occurs!";  
var pos = str.lastIndexOf("locate");
```

c. Cắt chuỗi với Slice

```
var str = "Apple, Banana, Kiwi";  
var res = str.slice(7, 13); // Banana
```

9. Array (Mảng)

a. Tạo mảng

```
var array_name = [item1, item2, ...];
```

hoặc

```
var cars = new Array("Saab", "Volvo", "BMW");
```

Ví dụ:

```
var cars = [  
    "Saab",  
    "Volvo",  
    "BMW"  
];
```

b. Truy cập phần tử trong mảng

```
var name = cars[0];
```

c. Thay đổi giá trị phần tử trong mảng

```
cars[0] = "Opel";
```

d. Lặp phần tử trong mảng



10. Hàm xử lý mảng

a. Lấy phần tử cuối ra khỏi mảng

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();
```

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.pop();    // the value of x is
"Mango"
```

b. Lấy phần tử đầu ra khỏi mảng

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();          // Removes the first
element "Banana" from fruits
```

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.shift();  // the value of x is
"Banana"
```

c. Thêm phần tử vào cuối mảng

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Lemon");    // adds a new element
(Lemon) to fruits
```

d. Thêm phần tử vào đầu mảng

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon"); // Adds a new element
"Lemon" to fruits
```


e. Thêm phần tử vào vị trí bất kì trong mảng

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.splice(2, 0, "Lemon", "Kiwi");
```

f. Sắp xếp mảng

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.sort();           // Sorts the elements of fruits
```

Sắp xếp ngược:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.sort();           // First sort the elements of  
fruits                   fruits  
fruits.reverse();        // Then reverse the order of  
the elements
```

11. Câu lệnh điều kiện

a. if ... else

```
if (condition) {  
    // block of code to be executed if the condition is true  
}  
  
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```



b. switch...case

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```

Ví dụ:

```
switch (new Date().getDay()) {  
    case 6:  
        text = "Today is Saturday";  
        break;  
    case 0:  
        text = "Today is Sunday";  
        break;  
    default:  
        text = "Looking forward to the Weekend";  
}
```

12. Vòng lặp

a. for

```
for (i = 0; i < 5; i++) {  
    console.log(i);  
}
```

```
var i ;  
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
  
for (i = 0; i < fruits.length; i++) {  
    console.log(fruits[i]);  
}
```

b. while

```
while (condition) {  
    // code block to be executed  
}
```

Ví dụ:

```
while (i < 10) {  
    console.log(i);  
    i++;  
}
```

c. do...while

```
do {  
    // code block to be executed  
}  
while (condition);
```

13. Errors và try...catch

```
try {  
    // Block of code to try  
  
}  
catch(err) {  
    // Block of code to handle errors  
  
}
```