**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;

1. **Toán tử:**
2. **Toán tử số học**

|  |  |
| --- | --- |
| Operator | Description |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| \*\* | Exponentiation ([ES2016](https://www.w3schools.com/js/js_es6.asp)) |
| / | Division |
| % | Modulus (Division Remainder) |
| ++ | Increment |
| -- | Decrement |

1. **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 |

1. **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 |

1. **Toán tử logic:**

|  |  |
| --- | --- |
| Operator | Description |
| && | logical and |
| || | logical or |
| ! | logical not |

1. **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)

1. **undefined**

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

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

1. **Empty**

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

1. **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

1. **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;             
}

1. **Objects (Đối tượng)**

|  |  |  |
| --- | --- | --- |
| Object | Properties | Methods |
| https://www.w3schools.com/js/objectExplained.gif | car.name = Fiat  car.model = 500  car.weight = 850kg  car.color = white | car.start()  car.drive()  car.brake()   car.stop() |

1. **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;  
  }

};

1. **Truy cập thuộc tính**

*objectName.propertyName*

hoặc

*objectName["propertyName"]*

1. **Truy cập phương thức**

*objectName.methodName()*

1. **Hàm xử lý chuỗi**
2. **Đếm chiều dài chuỗi**

var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var sln = txt.length;

1. **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");

1. **Cắt chuỗi với Slice**

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

1. **Array (Mảng)**
2. **Tạo mảng**

var *array\_name* = [*item1*, *item2*, ...];

hoặc

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

Ví dụ:

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

1. **Truy cập phần tử trong mảng**

var name = cars[0];

1. **Thay đổi giá trị phần tử trong mảng**

cars[0] = "Opel";

1. **Lặp phần tử trong mảng**
2. **Hàm xử lý mảng**
3. **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"

1. **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"

1. **Thêm phần tử vào cuối mảng**

var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.push("Lemon");    // adds a new element (Lemon) to fruits

1. **Thêm phần tử vào đầu mảng**

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

1. **Thêm phần tử vào vị trí bất kì trong mảng**

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

1. **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.reverse();     // Then reverse the order of the elements

1. **Câu lệnh điều kiện**
2. **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}

1. **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";  
}

1. **Vòng lặp**
2. **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]);  
}

1. **while**

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

Ví dụ:

while (i < 10) {  
  console.log(i);

  i++;  
}

1. **do…white**

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

1. **Errors và try…catch**

try {   
  // Block of code to try

}  
catch(err) {

// Block of code to handle errors

}

**BÀI TẬP**





