JavaScript Code Output

What will happen when the following code is executed?

```
if (true) {
var x = 6;
let y=7;
const z=9;
}
console.log(x);      // 6
console.log(y);      // Reference Error Y is not defined
console.log(z);      // Reference Error z is not defined
```

What will happen when the following code is executed?

```
const a = 10;
const b = 20;
b = a || 10;
b = a || 12;
console.log(b);      // Uncaught Type Error : Assignment to constant variable.
```

What will happen when the following code is executed?

```
var x = 5;
console.log(x);      // 5
if (true) {
var x = 6;
console.log(x);      // 6
}
console.log(x);      // 6
```

What will happen when the following code is executed?

```
var x = 5;
console.log(x);      // 5
if (false) {
var x = 6;
console.log(x);      // it won't go inside the loop
}
console.log(x);      // 5
```

What will happen when the following code is executed?

```
var x = 5;
function a() {
var x = 6;
return x;
}
console.log(x);      // 5
console.log(a());    // 6
```

What will happen when the following code is executed?

```
var x = 5;
function a() {
x = 6;
return x;
}
```

```
console.log(x);      // 5
console.log(a());    // 6
```

What will happen when the following code is executed?

```
let x = 5;
function a() {
let x = 6;
return x;
}
console.log(x);      // 5
console.log(a());    // 6
```

What will happen when the following code is executed?

```
const x = 5;
function a() {
let x = 6;
return x;
}
console.log(x);      // 5
console.log(a());    // 6
```

What will happen when the following code is executed?

```
const x = 5;
function a() {
x = 6;
return x;
}
console.log(x);      // 5
console.log(a());    // "TypeError: Assignment to constant variable.
```

What will happen when the following code is executed?

```
const x = 5;
function a() {
const x = 6;
return x;
}
console.log(x);      // 5
console.log(a());    // 6
```

What will happen when the following code is executed?

```
const x = 5;
function a() {
var x = 6;
return x;
}
console.log(x);      // 5
console.log(a());    // 6
```

What will happen when the following code is executed?

```
var x = 5;
function a() {
var x = 6;
return x;
}
x;
a();    // 6
```

What will happen when the following code is executed?

```
var x = 5;
function a() {
var x = 6;
return x;
}
x = a();
console.log(x)       // 6
```

What will happen when the following code is executed?

```
var x = { num: 5 };
function a() {
  delete x.num;
  return x.num;
}
console.log(a());    // undefined
```

What will happen when the following code is executed?

```
var x = 10;
function a() {
  delete x;
  return x;
}
console.log(a());    //  10 – Delete won't work because it is not an object
```

What will happen when the following code is executed?

```
var myObject = {
  foo: "bar",
  func: function () {
    var self = this;
    console.log(this.foo);   // bar
    console.log(self.foo);   // bar
    (function () {
      console.log(this.foo); // undefined
      console.log(self.foo); // bar
    })();
  }
};
myObject.func();
```

What will happen when the following code is executed?

```
(function() {
  var a = b = 3;
})();
console.log(a);     // Uncaught ReferenceError: a is not defined
console.log(b);      // 3
var a = b = 3; is actually shorthand for: b = 3;var a = b;
```

What will happen when the following code is executed?

```
console.log(0.1 + 0.2); // 0.30000000000000004
console.log(0.1 + 0.2 == 0.3); // false
```

What will happen when the following code is executed?

```
function foo1() {
  return {
    bar: "hello"
  };
}
function foo2() {
  return
  {
    bar: "hello";
  }
}
console.log(foo1());        // { bar: "hello" }
console.log(foo2());        // Undefined – because return statement should be in same line
```

What will happen when the following code is executed?

```
(function () {
    console.log(1);
    setTimeout(function (){console.log(2)}, 1000);
    setTimeout(function (){console.log(3)}, 0);
    console.log(4);
})();
// 1 , 4, 3, 4
```

What will happen when the following code is executed?

```
console.log(
  (function f(n){
    return ((n > 1) ? n * f(n-1) : n)
  })(10)
);
// 3628800 – factorial of 10
```

What will happen when the following code is executed?

```
const a = [{ type: "a" }, { type: "b" }];
const b = [...a];
a[0].type = "c";
console.log(a[0].type); // c
console.log(b[0].type); // c  -  shallow copy
```

What will happen when the following code is executed?

```
console.log(1 + "2" + "2");  //122
console.log(1 + +"2" + "2");  // 32
console.log(1 + -"1" + "2");  // 02
console.log(+"1" + "1" + "2");  //112
console.log("A" - "B" + "2");  //NAN2
console.log("A" - "B" + 2);  // NAN
console.log(1 + undefined) );  // NAN
console.log(1 + null); // 1
console.log(1 + true); //2
console.log(1 + false);  //1
console.log(true + false); //1
console.log(true + 'false'); //truefalse
console.log(false + false); //0
console.log(3>2>1); //false
console.log(1<2<3); //true
console.log((0 || 1)); //1
console.log((1 || 2));//1
console.log((0 && 1)); //0
console.log((1 && 2));//2
console.log(false == '0') // true
console.log(false === '0') // false
```

What will happen when the following code is executed?

```
for (let i = 0; i < 5; i++) {
setTimeout(
   function () {
     console.log(i);  // 0 1 2 3 4
   }, i * 1000 );
}
```

What will happen when the following code is executed?

```
for (var i = 0; i < 5; i++) {
setTimeout(
   function () {
     console.log(i);  // 5 will print 5 times
   }, 1000 );
}
```

What will happen when the following code is executed?

```
var b = 1;
function outer() {
  var b = 2;
  function inner() {
    b++;
    var b = 8;
    console.log(b); // 8
  }
  inner();
}
outer();
```

What will happen when the following code is executed?

```
function fun() {
  var abc = "test";
  function abc() {
    return "hello";
  }
  return abc();
}
console.log(fun()); // Uncaught Type Error: abc is not a function
```

What will happen when the following code is executed?

```
function dump() {
  console.log(a);    // undefine
  console.log(b);    // reference Error cannont access b before initialization
  var a = 10;
  let b = 6;
}
dump();
```

What will happen when the following code is executed?

```
function dump() {
  var a = 10;
  console.log(a);
}
var a = 22;
console.log(a);      // 22
```

What will happen when the following code is executed?

```
var x = 100;
var obj = {
  x: 43,
  y: "test",
  z: function () {
    return this.x;
  }
};
console.log(obj.z());        // 43
```

What will happen when the following code is executed?

```
var x = 0;
var y = 23;
if (x) {
  console.log(x);   // condtion will be false it wont go inside
}
if (y) {
  console.log(y);   // 23
}
```

What will happen when the following code is executed?

```
function dump() {
 a = 10;
console.log(a);      // 10
}
dump();
```

What will happen when the following code is executed?

```
console.log(sum(10, 20));         // 30
console.log(diff(10, 20));        // undefined
function sum(a, b) {
  return a + b;
}
let diff = function (x, y) {
  return x - y;
};
```

What will happen when the following code is executed?

```
var person = {
  age: 43,
  grow:  ()=> {
    this.age++;
  }
};
person.grow();
console.log(person.age);// 43
```

What will happen when the following code is executed?

```
var person = {
  age: 43,
  grow: function () {
    this.age++;
  }
};
person.grow();
console.log(person.age);// 44
```

What will happen when the following code is executed?

```
let a = 10;
var a = 20;
console.log(a);    // Uncaught SyntaxError: Identifier 'a' has already been declared
```

What will happen when the following code is executed?

```
var p = 3;
var b = p++;
var c = ++p;
console.log(p, b, c);        // 5, 3, 5
```

What will happen when the following code is executed?

```
for (let i = 0; i < 100; i++) {
  if (i === 6) {
    break;
  }
  console.log(i);      // 0 1 2 3 4 5
}
console.log(i);      // Uncaught ReferenceError: i is not defined
```

Guess the outputs of the following codes:

// Code 1:

```
function func1(){
  setTimeout(()=>{
    console.log(x);
    console.log(y);
  },3000);
  var x = 2;
  let y = 12;
}
func1();
```

Output: Outputs 2 and 12 . Since, even though let variables are not hoisted, due to async nature of javascript, the complete function code runs before the setTimeout function. Therefore, it has access to both x and y.

// Code 2:

```
function func2(){
  for(var i = 0; i < 3; i++){
    setTimeout(()=> console.log(i),2000);
}
}
func2();
```

Output: Outputs 3 , three times since variable declared with var keyword does not have block scope. Also, inside the for loop, the variable i is incremented first and then checked.

// Code 3:

```
(function(){
  setTimeout(()=> console.log(1),2000);
  console.log(2);
  setTimeout(()=> console.log(3),0);
  console.log(4);
})();
```

Output:

2 4 3

1 // After two seconds

Even though the second timeout function has a waiting time of zero seconds, the javascript engine always evaluates the setTimeout function using the Web API and therefore, the complete function executes before the setTimeout function can execute.

Guess the outputs of the following code:

// Code 1:

```
let x= {}, y = {name:"Ronny"},z = {name:"John"};
x[y] = {name:"Vivek"};
x[z] = {name:"Akki"};
console.log(x[y]);
```

Output: Output will be {name: “Akki”}.

Adding objects as properties of another object should be done carefully.

Writing x[y] = {name:”Vivek”} , is same as writing x[‘object Object’] = {name:”Vivek”} , While setting a property of an object, javascript coerces the parameter into a string. Therefore, since y is an object, it will be converted to ‘object Object’. Both x[y] and x[z] are referencing the same property.

// Code 2:

```
function runFunc(){
  console.log("1" + 1);      // 11
  console.log("A" - 1);      // NAN
  console.log(2 + "-2" + "2");      // 2-22
  console.log("Hello" - "World" + 78); // NAN
  console.log("Hello"+ "78");      // Hello78
}
runFunc();
```

// Code 3:

```
let a = 0;
let b = false;
console.log((a == b));      // true
console.log((a === b));    // false
```

Guess the output of the following code:

```
var x = 23;
(function(){
  var x = 43;
  (function random(){
    x++;
    console.log(x); // NAN
    var x = 21;
  })();
})();
```

Output is NaN - random() function has functional scope, since x is declared and hoisted in the functional scope.

Rewriting the random function will give a better idea about the output:

```
function random(){
  var x; // x is hoisted
  x++; // x is not a number since it is not initialized yet
  console.log(x); // Outputs NaN
  x = 21; // Initialization of x
}
```

Guess the outputs of the following code:

// Code 1

```
let hero = {
  powerLevel: 99,
  getPower(){
    return this.powerLevel;
  }
}
let getPower = hero.getPower;
let hero2 = {powerLevel:42};
console.log(getPower());        // Undefined
console.log(getPower.apply(hero2));         // 42
```

Reason - The first output is undefined since when the function is invoked, it is invoked referencing the global object:
window.getPower() = getPower();

// Code 2

```
const a = function(){
  console.log(this);        // global/window object
  const b = {
    func1: function(){
      console.log(this);    // object "b"
    }
  }
  const c = {
    func2: ()=>{
      console.log(this);    // global/window object
    }
  }
  b.func1();
  c.func2();
}
a();
```

// Code 3

```
  const b = {
  name:"Vivek",
  f: function(){
    var self = this;
    console.log(this.name);      // Vivek
    (function(){
      console.log(this.name);    // Undefined
      console.log(self.name);    // Vivek
    })();
  }
}
b.f();
```

Guess the outputs of the following code:

****Note** - Code 2 and Code 3 require you to modify the code, instead of guessing the output.

// Code 1

```
(function(a){
  return (function(){
    console.log(a); // 45
    a = 23;
  })()
})(45);
```

Output: 45 .

Even though a is defined in the outer function, due to closure the inner functions have access to it.

// Code 2

```
// Each time bigFunc is called, an array of size 700 is being created,
// Modify the code so that we don't create the same array again and again
function bigFunc(element){
  let newArray = new Array(700).fill('♥');
  return newArray[element];
}
console.log(bigFunc(599)); // Array is created
console.log(bigFunc(670)); // Array is created again
```

Output : This code can be modified by using closures,

```
function bigFunc(){
  let newArray = new Array(700).fill('♥');
  return (element) => newArray[element];
}
let getElement = bigFunc(); // Array is created only once
getElement(599);
getElement(670);
```

// Code 3

```
// The following code outputs 2 and 2 after waiting for one second
// Modify the code to output 0 and 1 after one second.
function randomFunc(){
  for(var i = 0; i < 2; i++){
    setTimeout(()=> console.log(i),1000);
  }
}
randomFunc();
```

Output: Can be modified in two ways:

Using let keyword:

```
function randomFunc(){
  for(let i = 0; i < 2; i++){
    setTimeout(()=> console.log(i),1000);
  }
}
randomFunc();
```

Using closure:

```
function randomFunc(){
```

```
  for(var i = 0; i < 2; i++){
  (function(i){
     setTimeout(()=>console.log(i),1000);
   })(i);
  }
}
randomFunc();
```

Guess the outputs of the following code:

```
function abc(){
        let x = this ? class y {} : class z {};
        console.log(typeof x + ' , ' + typeof z); // "function , undefined"
}
abc();
```

Guess the outputs of the following code:

```
const num = 4;
const number = [];
for (var i=0;i<num; i++){
number.push(i +1);
}
console.log(number);      // [1,2,3,4]
```

Guess the outputs of the following code:

```
console.log(1[1])           // undefined
console.log('abn'['sfsdg'])// undefined
console.log(~1)                  // -2
console.log(2*'2')               // 4
console.log(2**2**1)             // 4
console.log(8563465465486)// 8563465465486
console.log(2-'2')               // 0
console.log(2 - - '2')           // 4
console.log(2 + + '2')           // 4
console.log(3>2>1)           // false
console.log(1<2<3)            // true
console.log([] == ![])          // true
```

Guess the outputs of the following code:

```
const person = {
  name: 'shiva'
}
const nameOf = [person,person,person];
nameOf[0].name = 'kumar';
console.log(nameOf[1].name) // Kumar – Shallow Copy
```

Guess the outputs of the following code:

```
console.log(callme());      // sunitha
console.log(xy);            // undefined
console.log(msg);           // undefined
console.log(window.xy);// undefined
console.log(window.msg);// undefined
```

```
console.log(sendme);    // undefined
var xy =10;
var msg = "ss";
function callme() {
  console.log('sunitha')
}
var sendme = function() {
  console.log('Aadya')
}
```

Guess the outputs of the following code:

```
function someFunc() {
  var x = 50;
  console.log(x);
}
var x = 100;
console.log(x);                  // 100
```

Guess the outputs of the following code:

```
var x = 0;
var y = 23;
if(x) { console.log(x) }// 0 means false it wont go inside if
if(y) { console.log(y) }          // 23
```

Guess the outputs of the following code:

```
console.log(sum(10,20));    // 30
console.log(diff(10, 20));// Reference Error
function sum(x, y) {
    return x + y;
}
let diff = function(x, y) {
    return x - y;
}
```

Guess the outputs of the following code:

```
const person = {
  age: 45,
  grow: () => {
    this.age++;
  }
}
person.grow();
console.log(person.age)      // 45
```