JavaScript Quiz Questions:

(Prepared by: "Ismail Shah")

1.Which of the following is not a primitive data type in JavaScript?

a) Number

b) String

c) Boolean

d) Object

Answer: d) Object

Explanation: In JavaScript, the primitive data types are Number, String, Boolean, Null, and Undefined. Object is not a primitive data type.

2.What does the "typeof" operator do in JavaScript?

a) Returns the data type of a variable

b) Checks if a variable is defined

c) Assigns a value to a variable

d) Concatenates two strings

Answer: a) Returns the data type of a variable

Explanation: The "typeof" operator in JavaScript returns the data type of a variable. For example, "typeof 42" would return "number".

What is the output of the following code:
console.log(2 + "2");

a) "4"

b) "22"

c) 4

d) 22

Answer: b) “22”

Explanation: When a number is concatenated with a string, JavaScript will convert the number to a string and concatenate the two values. Therefore, the output of the code is “22”.

Which of the following is not a comparison operator in JavaScript?

a) ==

b) ===

c) !=

d) =<

Answer: d) =<

Explanation: The correct comparison operator is “<=”, not “=<“.

What is the output of the following code:
```
var x = 5;
console.log(x++);
```

a) 4

b) 5

c) 6

d) Error

Answer: b) 5

Explanation: The “++” operator increments the value of the variable by 1 after the value is returned. Therefore, the output of the code is 5.

What is the output of the following code:
```
var x = true;
console.log(!x);
```

a) true

b) false

c) undefined

d) Error

Answer: b) false

Explanation: The “!” operator inverts the value of a boolean expression. Since the value of “x” is true, “!x” will evaluate to false.

What does the “NaN” value represent in JavaScript?

a) Not a number

b) Null value

c) Undefined value

d) Boolean value

Answer: a) Not a number

Explanation: The “NaN” value in JavaScript represents “Not a Number” and is returned when a mathematical operation fails to produce a valid number.

What is the output of the following code:
var x = 5;

var y = “5”;

console.log(x == y);

a) true

b) false

c) undefined

d) Error

Answer: a) true

Explanation: The “==” operator in JavaScript performs type coercion, which means that the values are converted to a common type before being compared. In this case, the string “5” is converted to the number 5, so the values of x and y are equal.

What is the correct way to declare a variable in JavaScript?

a) var x = 5;

b) variable x = 5;

c) x = 5;

d) let x = 5;

Answer: d) let x = 5;

Explanation: In modern JavaScript, the recommended way to declare a variable is using the “let” keyword.

What does the “this” keyword refer to in JavaScript?

a) The current function

b) The global object

c) The object that the function belongs to

d) The parent object of the current object

Answer: c) The object that the function belongs to

Explanation: The “this” keyword in JavaScript refers to the object that the function belongs to. The value of “this” is determined at runtime based on how the function is called.

What is the output of the following code:
var x = [1, 2, 3];

console.log(x.length);

a) 1

b) 2

c) 3

d) 4

Answer: c) 3

Explanation: The “length” property in JavaScript returns the number of elements in an array. In this case, the array “x” has three elements, so the output is 3.

What is the output of the following code:
console.log(typeof NaN);

a) “number”

b) “string”

c) “undefined”

d) “NaN”

Answer: a) “number”

Explanation: Although NaN stands for “Not a Number”, its type in JavaScript is actually “number”.

What does the “forEach” method do in JavaScript?

a) Adds a new element to the end of an array

b) Removes an element from the beginning of an array

c) Executes a function once for each element in an array

d) Reverses the order of the elements in an array

Answer: c) Executes a function once for each element in an array

Explanation: The “forEach” method in JavaScript executes a provided function once for each element in an array.

What is the output of the following code:
console.log(2 ** 3);

a) 5

b) 6

c) 8

d) 9

Answer: c) 8

Explanation: The “**” operator in JavaScript is the exponentiation operator, which raises the first operand to the power of the second operand. In this case, 2 raised to the power of 3 is 8.

What is the correct syntax for a “for” loop in JavaScript?

a) for (var i = 0; i < 5; i++)

b) for (i = 0; i < 5; i++)

c) for (var i = 5; i > 0; i–)

d) for (i = 5; i > 0; i–)

Answer: a) for (var i = 0; i < 5; i++)

Explanation: The correct syntax for a “for” loop in JavaScript includes the initialization statement, the condition, and the increment statement, all separated by semicolons.

What is the output of the following code:

```
var x = 5;
var y = “5”;
console.log(x === y);
```

a) true

b) false

c) undefined

d) Error

Answer: b) false

Explanation: The “===” operator in JavaScript performs a strict comparison, which means that the values and the types must be equal for the comparison to return true. In this case, x is a number and y is a string

What is the difference between “==” and “===” operators in JavaScript?

a) They are interchangeable

b) “==” performs a strict comparison, while “===” performs a loose comparison

c) “===” performs a strict comparison, while “==” performs a loose comparison

d) They both perform the same type of comparison

Answer: c) “===” performs a strict comparison, while “==” performs a loose comparison

Explanation: The “===” operator in JavaScript performs a strict comparison, which means that the values and the types must be equal for the comparison to return true. The “==” operator performs a loose comparison, which means that it tries to convert the operands to the same type before making the comparison.

What is the output of the following code:

```
var x = 10;
var y = “5”;
console.log(x – y);
```

a) 5

b) 10

c) 15

d) “105”

Answer: a) 5

Explanation: When JavaScript tries to perform arithmetic operations with a string and a number, it automatically converts the string to a number before making the calculation. In this case, “5” is converted to 5, so the result of the subtraction is 5.

What is the output of the following code:

```
var x = [1, 2, 3];
var y = […x];
console.log(y);
```

a) [1, 2, 3]

b) [3, 2, 1]

c) [1, 2]

d) Error

Answer: a) [1, 2, 3]

Explanation: The spread operator (…) in JavaScript allows you to copy the elements of an array into a new array. In this case, the variable “y” is assigned a new array that contains the same elements as “x”.

What is the output of the following code:
console.log(Math.random());

a) 0

b) 0.5

c) 1

d) A random number between 0 and 1

Answer: d) A random number between 0 and 1

Explanation: The Math.random() function in JavaScript generates a random number between 0 (inclusive) and 1 (exclusive) each time it is called.

What is the difference between “let” and “const” keywords in JavaScript?

a) They are interchangeable

b) “let” variables cannot be reassigned, while “const” variables can

c) “const” variables cannot be reassigned, while “let” variables can

d) “let” and “const” both refer to constant variables

Answer: c) “const” variables cannot be reassigned, while “let” variables can

Explanation: In JavaScript, “let” and “const” are used to declare variables. The difference is that “let” variables can be reassigned a new value, while “const” variables cannot be reassigned a new value after they are declared. Both “let” and “const” variables are block-scoped.

Which of the following is not a data type in JavaScript?

a) Boolean

b) String

c) Number

d) Character

Answer: d) Character

Explanation: In JavaScript, there is no data type called “Character”. Instead, characters are represented using the “String” data type.

What is the output of the following code:
console.log(“5” + 5);

a) “10”

b) 10

c) “55”

d) Error

Answer: c) “55”

Explanation: When you use the “+” operator with a string and a number, JavaScript converts the number to a string and concatenates it to the string.

What is the difference between “var” and “let” keywords in JavaScript?

a) They are interchangeable

b) “var” variables cannot be reassigned, while “let” variables can

c) “let” variables have block scope, while “var” variables have function scope

d) “var” and “let” both refer to constant variables

Answer: c) “let” variables have block scope, while “var” variables have function scope

Explanation: In JavaScript, “var” and “let” are used to declare variables. The difference is that “let” variables have block scope, which means they are only accessible within the block they are declared in, while “var” variables have function scope, which means they are accessible throughout the function they are declared in.

What is the output of the following code:

```
var x = 10;
if (true) {
var x = 5;
}
console.log(x);
```

a) 10

b) 5

c) undefined

d) Error

Answer: b) 5

Explanation: In JavaScript, “var” variables are function-scoped, not block-scoped. This means that the “x” variable inside the if statement is the same as the “x” variable outside of it, and changing its value inside the block also changes its value outside the block.

What is the output of the following code:

```
var x = [1, 2, 3];

console.log(x[3]);
```

a) 3

b) undefined

c) Error

d) NaN

Answer: b) undefined

Explanation: In JavaScript, arrays are zero-indexed, which means that the first element of an array has an index of 0. Since the “x” array has only three elements, attempting to access the fourth element using an index of 3 results in undefined.

Which of the following is not a valid way to declare a function in JavaScript?

a) function myFunction() {}

b) var myFunction = function() {}

c) () => {}

d) function = {}

Answer: d) function = {}

Explanation: The syntax for declaring a function in JavaScript is either “function functionName() {}” or “var functionName = function() {}”, or using an arrow function expression “() => {}”. The option d) is not a valid way to declare a function in JavaScript.

What is the difference between “==” and “===” operators in JavaScript?

a) They are interchangeable

b) “==” checks for value equality, while “===” checks for value and type equality

c) “===” checks for value equality, while “==” checks for value and type equality

d) They both perform the same operation

Answer: b) “==” checks for value equality, while “===” checks for value and type equality

Explanation: The “==” operator checks for value equality, and it performs type coercion if the types of the two values being compared are different. The “===” operator checks for value and type equality, and it does not perform type coercion.

What is the output of the following code:
console.log(typeof undefined);

a) “undefined”

b) “null”

c) “object”

d) “string”

Answer: a) “undefined”

Explanation: “undefined” is a primitive data type in JavaScript that represents the absence of a value. The typeof operator returns a string indicating the type of the operand.

What is the output of the following code:
console.log(1 + “2” + 3);

a) “123”

b) 6

c) “15”

d) Error

Answer: a) “123”

Explanation: When you use the “+” operator with a string and a number, JavaScript converts the number to a string and concatenates it to the string. So in this case, “1” is concatenated with “2” to create “12”, and then “3” is concatenated to create “123”.

Which of the following is not a loop in JavaScript?

a) for

b) while

c) do…while

d) next

Answer: d) next

Explanation: “next” is not a loop in JavaScript. The correct loop keywords are “for”, “while”, and “do…while”.

Which of the following is not a valid JavaScript array method?

a) push()

b) pop()

c) shift()

d) slice()

Answer: d) slice()

Explanation: “slice()” is a valid array method in JavaScript, but it is not a method for modifying the array. Instead, it returns a shallow copy of a portion of the array as a new array.

What is the output of the following code:
console.log(typeof []);

a) “array”

b) “object”

c) “array[]”

d) “undefined”

Answer: b) “object”

Explanation: In JavaScript, arrays are a type of object. The typeof operator returns “object” for any object type, including arrays.

What is the output of the following code:
```
var x = 10;

function foo() {
console.log(x);
var x = 5;
}
foo();
```

a) 10

b) 5

c) undefined

d) Error

Answer: c) undefined

Explanation: In JavaScript, variable declarations are hoisted to the top of the function, but their assignments are not. So the “x” variable inside the “foo” function is hoisted, but it is not assigned a value until after the console.log statement. Therefore, the output is “undefined”.

What is the output of the following code:

```
var x = 1;

function foo() {

var x = 2;

function bar() {
console.log(x);
}
return bar;
}
var baz = foo();

baz();
```

a) 1

b) 2

c) undefined

d) Error

Answer: b) 2

Explanation: In JavaScript, inner functions have access to the variables of their outer functions, even after the outer function has returned. In this case, the “baz” variable is assigned the “bar” function returned by “foo”, and when “baz” is called, it accesses the “x” variable of its parent function “foo”.

What is the output of the following code:

```
console.log(“5” == 5);
```

a) true

b) false

c) Error

d) NaN

Answer: a) true

Explanation: In JavaScript, the "==" operator performs type coercion before comparison. In this case, the string "5" is converted to the number 5 before comparison with the number 5, resulting in true.

Which of the following is not a valid way to declare a variable in JavaScript?

a) var x;

b) let y;

c) const z;

d) variable w;

Answer: d) variable w;

Explanation: "variable" is not a valid keyword for declaring variables in JavaScript. The correct keyword is "var", "let", or "const".

What is the output of the following code:
console.log(typeof null);

a) "null"

b) "object"

c) "undefined"

d) Error

Answer: b) "object"

Explanation: In JavaScript, the typeof operator returns "object" for the null value. This is considered to be a historical bug in the language.

What is the output of the following code:
console.log("Hello".charAt(1));

a) "H"

b) “e”

c) “l”

d) “o”

Answer: b) “e”

Explanation: In JavaScript, the “charAt” method is used to access a character in a string by its index. In this case, the character at index 1 is “e”.

What is the output of the following code:

```
var x = 5;

function foo() {

console.log(x);
}
function bar() {
var x = 10;
foo();
}

bar();
```

a) 5

b) 10

c) undefined

d) Error

Answer: a) 5

Explanation: In JavaScript, inner functions have access to the variables of their outer functions, even if the outer function has already returned. In this case, the “foo” function accesses the “x” variable in the global scope, which has a value of 5. The “x” variable declared in the “bar” function is a separate variable with a different scope.

What is the output of the following code:

```
console.log(2 + 3 + “4”);
```

a) “54”

b) “9”

c) “234”

d) Error

Answer: a) “54”

Explanation: In JavaScript, the “+” operator performs both addition and string concatenation. When the “+” operator is used with both numbers and strings, it performs addition first and then concatenates the result with the remaining string. In this case, 2 + 3 = 5, and then “4” is concatenated to create “54”.

Which of the following is not a valid way to create a JavaScript array?

a) var arr = [];

b) var arr = new Array();

c) var arr = Array();

d) var arr = {1, 2, 3};

Answer: d) var arr = {1, 2, 3};

Explanation: Curly braces are used to create JavaScript objects, not arrays. The correct way to create an array is to use square brackets or the Array constructor.

What is the output of the following code:
console.log(NaN == NaN);

a) true

b) false

c) undefined

d) Error

Answer: b) false

Explanation: In JavaScript, NaN is a special value that represents “Not a Number”. NaN is not equal to any other value, including itself.

What is the output of the following code:
var x = 5;

var y = x++;

console.log(y);

a) 4

b) 5

c) 6

d) Error

Answer: b) 5

Explanation: In JavaScript, the “++” operator is used to increment a variable by 1. When used after the variable (i.e. x++), the increment happens after the current value is used. In this case, the value of x is 5, so y is assigned the value of 5 before x is incremented to 6.

What is the output of the following code:
var x = [1, 2, 3];

var y = x.slice(1);

console.log(y);

a) [1, 2]

b) [2, 3]

c) [1, 3]

d) Error

Answer: b) [2, 3]

Explanation: In JavaScript, the “slice” method is used to create a new array that contains a portion of an existing array. In this case, the “y” variable is assigned a new array that contains all elements of “x” starting from index 1 (i.e. [2, 3]).

What is the output of the following code:

var x = [1, 2, 3];

console.log(x.length);

a) 1

b) 2

c) 3

d) 4

Answer: c) 3

Explanation: In JavaScript, the “length” property of an array returns the number of elements in the array. In this case, the “x” array has 3 elements, so the output is 3.

What is the output of the following code:
var x = “hello”;

console.log(x.charAt(0));

a) “h”

b) “e”

c) “l”

d) “o”

Answer: a) “h”

Explanation: In JavaScript, the “charAt” method is used to get the character at a specified index in a string. In this case, “x” is the string “hello”, and “charAt(0)” returns the first character, which is “h”.

What is the output of the following code:
var x = 5;

console.log(typeof x);

a) “number”

b) “string”

c) “boolean”

d) “undefined”

Answer: a) “number”

Explanation: In JavaScript, the “typeof” operator is used to get the type of a value or variable. In this case, “x” is assigned the value of 5, which is a number, so the output is “number”.

What is the output of the following code:
```
var x = “5”;

var y = 2;

console.log(x + y);
```

a) “7”

b) “52”

c) 7

d) 52

Answer: b) “52”

Explanation: In JavaScript, the “+” operator performs both addition and string concatenation. When the “+” operator is used with a string and a number, the number is converted to a string and then concatenated with the original string. In this case, “x” is the string “5”, and “y” is the number 2, so “x + y” concatenates the two values to create the string “52”.

What is the output of the following code:
```
var x = [1, 2, 3];

x.push(4);

console.log(x);
```

a) [1, 2, 3]

b) [2, 3, 4]

c) [1, 2, 3, 4]

d) Error

Answer: c) [1, 2, 3, 4]

Explanation: In JavaScript, the “push” method is used to add an element to the end of an array. In this case, the “x” array already contains the elements [1, 2, 3], and the “push(4)” statement adds the element 4 to the end of the array, resulting in the new array [1, 2, 3, 4].

What is the output of the following code:

```
var x = 10;

var y = “5”;

console.log(x – y);
```

a) 5

b) 10

c) 15

d) NaN

Answer: c) 15

Explanation: In JavaScript, when the “-” operator is used with a number and a string, the string is converted to a number and the subtraction is performed. In this case, “x” is the number 10, and “y” is the string “5”. However, because the “-” operator can also be used for string concatenation, the string “5” is first converted to the number 5, and then subtracted from 10 to get 15.

What is the output of the following code:

```
var x = “hello”;

console.log(x.toUpperCase());
```

a) “HELLO”

b) “Hello”

c) “hello”

d) Error

Answer: a) “HELLO”

Explanation: In JavaScript, the “toUpperCase” method is used to convert a string to all uppercase letters. In this case, “x” is the string “hello”, and “toUpperCase()” converts it to “HELLO”.

What is the output of the following code:

```
var x = true;

var y = false;

console.log(x || y);
```

a) true

b) false

c) Error

d) NaN

Answer: a) true

Explanation: In JavaScript, the “||” operator is the logical OR operator, and returns true if either operand is true. In this case, “x” is true, and “y” is false, so the expression “x || y” evaluates to true.

What is the output of the following code:

```
var x = [1, 2, 3];

console.log(x[0]);
```

a) 1

b) 2

c) 3

d) [1, 2, 3]

Answer: a) 1

Explanation: In JavaScript, arrays are indexed starting from 0. The square brackets notation is used to access an element of an array by its index. In this case, “x” is the array [1, 2, 3], and “x[0]” accesses the first element of the array, which is the number 1.

What is the output of the following code:
var x = 5;

var y = “10”;

console.log(x + y);

a) 15

b) “15”

c) “510”

d) 510

Answer: c) “510”

Explanation: In JavaScript, the “+” operator performs both addition and string concatenation. When the “+” operator is used with a number and a string, the number is converted to a string and then concatenated with the original string. In this case, “x” is the number 5, and “y” is the string “10”, so “x + y” concatenates the two values to create the string “510”.

What is the output of the following code:
var x = [1, 2, 3];

x.push(4);

console.log(x);

a) [1, 2, 3]

b) [1, 2, 3, 4]

c) [4, 3, 2, 1]

d) Error

Answer: b) [1, 2, 3, 4]

Explanation: In JavaScript, the “push” method is used to add an element to the end of an array. In this case, “x” is the array [1, 2, 3], and “x.push(4)” adds the number 4 to the end of the array to create the array [1, 2, 3, 4].

What is the output of the following code:
var x = “10”;

var y = “5”;

console.log(x + y);

a) 15

b) “15”

c) “105”

d) 105

Answer: c) “105”

Explanation: In JavaScript, the “+” operator performs both addition and string concatenation. When the “+” operator is used with two strings, the two strings are concatenated. In this case, “x” is the string “10”, and “y” is the string “5”, so “x + y” concatenates the two strings to create the string “105”.

What is the output of the following code:
var x = 10;

console.log(typeof x);

a) number

b) string

c) boolean

d) undefined

Answer: a) number

Explanation: In JavaScript, the “typeof” operator is used to determine the data type of a value. In this case, “x” is the number 10, so “typeof x” returns the string “number”.

What is the output of the following code:
var x = [1, 2, 3];

console.log(x.length);

a) 0

b) 1

c) 2

d) 3

Answer: d) 3

Explanation: In JavaScript, the “length” property is used to get the number of elements in an array. In this case, “x” is the array [1, 2, 3], and “x.length” returns the number 3.

What is the output of the following code:
var x = “10”;

var y = “5”;

console.log(x – y);

a) 5

b) “5”

c) “10”

d) NaN

Answer: a) 5

Explanation: In JavaScript, when the “-” operator is used with two strings, both strings are converted to numbers and the subtraction is performed. In this case, “x” is the string “10”, and “y” is the string “5”. Both strings are converted to numbers (10 and 5, respectively), and then 5 is subtracted from 10 to get 5.

What is the output of the following code:

var x = 5;

console.log(++x);

a) 5

b) 6

c) NaN

d) Error

Answer: b) 6

Explanation: In JavaScript, the “++” operator is used to increment a variable by 1. When the “++” operator is used before the variable (like in this example), the variable is first incremented and then used. So, “++x” increments the variable “x” from 5 to 6, and then returns the value 6.

What is the output of the following code:
var x = [1, 2, 3];

console.log(x[1]);

a) 1

b) 2

c) 3

d) Error

Answer: b) 2

Explanation: In JavaScript, arrays are zero-indexed, meaning the first element in the array has an index of 0, the second element has an index of 1, and so on. In this case, “x” is the array [1, 2, 3], and “x[1]” accesses the second element of the array (which has an index of 1), which is the number 2.

What is the output of the following code:
var x = 10;

if (x == “10”) {

console.log(“Equal”);

} else {

console.log(“Not equal”);

}

a) Equal

b) Not equal

c) Undefined

d) Error

Answer: a) Equal

Explanation: In JavaScript, the “==” operator performs a loose equality comparison, which means it compares the values without considering their data types. In this case, “x” is the number 10, and “x == ’10’” returns true because the string “10” is converted to the number 10 before the comparison is made. So, the “if” statement evaluates to true, and “Equal” is printed to the console.

What is the output of the following code:
var x = “hello”;

console.log(x.toUpperCase());

a) “hello”

b) “HELLO”

c) “Hello”

d) Error

Answer: b) “HELLO”

Explanation: In JavaScript, the “toUpperCase” method is used to convert a string to all uppercase letters. In this case, “x” is the string “hello”, and “x.toUpperCase()” converts the string to “HELLO” and prints it to the console.

What is the output of the following code:
var x = true;

var y = false;

console.log(x || y);

a) true

b) false

c) null

d) Error

Answer: a) true

Explanation: In JavaScript, the “||” operator performs a logical OR operation. The expression “x || y” evaluates to true if either “x” or “y” is true. In this case, “x” is true, so the expression “x || y” evaluates to true and is printed to the console.

What is the output of the following code:
var x = “5”;

console.log(typeof x);

a) “string”

b) “number”

c) “boolean”

d) “undefined”

Answer: a) “string”

Explanation: In JavaScript, the “typeof” operator is used to determine the data type of a variable. In this case, “x” is a string because it is enclosed in quotation marks.

What is the output of the following code:
var x = 3;

var y = “4”;

console.log(x + y);

a) “34”

b) “7”

c) 7

d) Error

Answer: a) “34”

Explanation: In JavaScript, the “+” operator is used to concatenate strings and add numbers. If one or both operands are strings, the “+” operator performs string concatenation. In this case, “x” is the number 3 and “y” is the string “4”, so the expression “x + y” converts the number 3 to a string and concatenates it with the string “4”, resulting in the string “34”.

What is the output of the following code:
```
var x = [1, 2, 3];

x.push(4);

console.log(x);
```

a) [1, 2, 3]

b) [1, 2, 3, 4]

c) [4, 3, 2, 1]

d) Error

Answer: b) [1, 2, 3, 4]

Explanation: In JavaScript, the “push” method is used to add elements to the end of an array. In this case, “x” is the array [1, 2, 3], and “x.push(4)” adds the number 4 to the end of the array, resulting in the array [1, 2, 3, 4].

What is the output of the following code:
```
var x = 5;

if (x === “5”) {

console.log(“Equal”);

} else {
```

console.log(“Not equal”);

}

a) Equal

b) Not equal

c) Undefined

d) Error

Answer: b) Not equal

Explanation: In JavaScript, the “===” operator performs a strict equality comparison, which means it compares the values and data types of the operands. In this case, “x” is the number 5, and “x === ‘5’” returns false because the string “5” is not the same data type as the number 5. So, the “if” statement evaluates to false, and “Not equal” is printed to the console.

What is the output of the following code:
var x = 10;

while (x > 0) {

console.log(x);

x- -
;}

a) 1 2 3 4 5 6 7 8 9 10

b) 10 9 8 7 6 5 4 3 2 1

c) 0

d) Error

Answer: b) 10 9 8 7 6 5 4 3 2 1

Explanation: In JavaScript, the “while” loop is used to repeatedly execute a block of code while a condition is true. In this case, the loop will execute as long as “x” is greater than 0. In each iteration of the loop, the value of “x” is printed to the console using the “console.log” function, and then “x” is decremented by 1.

What is the output of the following code:

```
var x = [1, 2, 3];

var y = x;

y.push(4);

console.log(x);
```

a) [1, 2, 3]

b) [1, 2, 3, 4]

c) [4, 3, 2, 1]

d) Error

Answer: b) [1, 2, 3, 4]

Explanation: In JavaScript, arrays and objects are reference types, which means that when a variable is assigned to an array or object, it holds a reference to the location in memory where the array or object is stored. In this case, “y” is assigned to the same array as “x” using the assignment operator, so both variables reference the same array. When “y.push(4)” is called, it modifies the array that both “x” and “y” reference, so the output of “console.log(x)” is the modified array [1, 2, 3, 4].

What is the output of the following code:

```
var x = 10;

var y = x++;

console.log(y);
```

a) 9

b) 10

c) 11

d) Error

Answer: b) 10

Explanation: In JavaScript, the “++” operator is used to increment a variable by 1. When “x++” is called, the value of “x” is first used as the operand of the expression, and then it is incremented by 1. The value of the expression “x++” is the original value of “x” before it was incremented, so “y” is assigned the value 10. The output of “console.log(y)” is therefore 10.

What is the output of the following code:
var x = 5;

var y = 3;

console.log(x *= y);

a) 8

b) 15

c) 12

d) 25

Answer: c) 15

Explanation: In JavaScript, the “*=” operator is used to perform multiplication and assignment in a single step. The expression “x *= y” is equivalent to “x = x * y”, which multiplies the value of “x” by the value of “y” and assigns the result back to “x”. The output of “console.log(x *= y)” is therefore the product of 5 and 3, which is 15.

What is the output of the following code:
var x = “10”;

var y = +”3″;

console.log(x + y);

a) “103”

b) “13”

c) 13

d) Error

Answer: b) “13”

Explanation: In JavaScript, the unary “+” operator can be used to convert a string to a number. When the expression “+ ‘3’” is evaluated, the string “3” is converted to the number 3. The expression “x + y” then concatenates the string “10” with the string “3”, resulting in the string “13”. The output of “console.log(x + y)” is therefore “13”.

What is the output of the following code:
var x = 0;

while (x < 5) {

console.log(x);

x += 2;

}

a) 0 2 4 6 8

b) 0 2 4

c) 2 4 6 8

d) Error

Answer: b) 0 2 4

What is the output of the following code:
console.log(“5” + 2);

a) “7”

b) 7

c) “52”

d) Error

Answer: c) “52”

Explanation: In JavaScript, the “+” operator is used for both addition and string concatenation. When one or both operands are strings, the “+” operator concatenates the strings. In this case, the string “5” is concatenated with the number 2, resulting in the string “52”. The output of “console.log(“5″ + 2)” is therefore “52”.

What is the output of the following code:
var x = [1, 2, 3];

console.log(x.slice(1));

a) [1, 2]

b) [2, 3]

c) [1, 2, 3]

d) Error

Answer: b) [2, 3]

Explanation: In JavaScript, the “slice” method is used to create a new array that contains a portion of an existing array. The “slice” method takes two arguments: the starting index (inclusive) and the ending index (exclusive) of the portion to be sliced. If the ending index is omitted, the “slice” method returns a new array that includes all elements from the starting index to the end of the original array. In this case, “x.slice(1)” creates a new array that includes all elements from index 1 (i.e., the second element) to the end of the “x” array. The output of “console.log(x.slice(1))” is therefore the array [2, 3].

What is the output of the following code:
var x = 5;

var y = 3;

if (x > y) {

console.log(“x is greater than y”);

} else {

console.log(“y is greater than x”);

}

a) “x is greater than y”

b) “y is greater than x”

c) 2

d) Error

Answer: a) “x is greater than y”

Explanation: In this code, an “if” statement is used to check whether the value of “x” is greater than the value of “y”. If the condition is true, the first block of code (i.e., the “console.log” statement) is executed. Otherwise, the second block of code is executed. In this case, the condition is true because 5 is greater than 3, so the output of “console.log” is “x is greater than y”.

What is the output of the following code:

```
var x = 10;

function foo() {

console.log(x);

var x = 5;

console.log(x);

}

foo();
```

a) 10 5

b) 5 10

c) Error

d) undefined 5

Answer: d) undefined 5

Explanation: In JavaScript, variable declarations are “hoisted” to the top of their scope, which means that they are processed before any other code in the same scope. However, variable assignments are not hoisted. In this case, the “foo” function includes a variable declaration “var x = 5” after the first “console.log” statement. This means that the “x” variable used in the first “console.log” statement refers to the local variable declared in the “foo” function, not the global variable with the same name. However, at the time of the first “console.log” statement, the local variable “x” has not yet been assigned a value, so its value is undefined.

What is the output of the following code:

console.log(typeof null);

a) “null”

b) “object”

c) “undefined”

d) Error

Answer: b) “object”

Explanation: In JavaScript, the “typeof” operator is used to determine the data type of a value or variable. When used with the “null” keyword, “typeof” returns “object”. This is a historical artifact of JavaScript, as “null” was originally intended to represent an empty object reference.

What is the output of the following code:
var x = [1, 2, 3];

var y = x;

y.push(4);

console.log(x);

a) [1, 2, 3]

b) [1, 2, 3, 4]

c) [4, 3, 2, 1]

d) Error

Answer: b) [1, 2, 3, 4]

Explanation: In JavaScript, arrays are objects, and objects are passed by reference rather than by value. This means that when “y” is assigned the value of “x”, it is actually assigned a reference to the same array in memory. Therefore, when “y.push(4)” is called, it modifies the same array that “x” refers to. The output of “console.log(x)” is therefore [1, 2, 3, 4].

What is the output of the following code:
console.log(5 == “5”);

a) true

b) false

c) Error

d) undefined

Answer: a) true

Explanation: In JavaScript, the “==” operator is used to compare two values for equality. When the values being compared have different data types, JavaScript will attempt to convert one or both of the values to a common data type before making the comparison. In this case, the string “5” is converted to the number 5 before the comparison is made. Therefore, the output of “console.log(5 == “5”)” is true.

What is the output of the following code:

```
var x = 10;

function foo() {

console.log(x);

}

foo();

var x = 5;
```

a) 10

b) 5

c) undefined

d) Error

Answer: a) 10

Explanation: In JavaScript, function declarations are processed before any other code in the same scope. Therefore, the “foo” function is defined before the global variable “x” is assigned the value 5. When “foo” is called, it outputs the current value of the global variable “x”, which is 10 at the time of the call. The output of “console.log(x)” is therefore 10.

What is the output of the following code:

var x = 5;

var y = 3;

var z = x++ – y–;

console.log(z);

a) 2

b) 3

c) 4

d) 5

Answer: a) 2

Explanation: In JavaScript, the “++” and “–” operators are used to increment and decrement a variable’s value by 1, respectively. The placement of these operators determines whether the variable’s value is incremented or decremented before or after an expression is evaluated. In this case, the expression “x++ – y–” is evaluated as follows:

The value of “x” (5) is used in the expression, and then incremented to 6.
The value of “y” (3) is used in the expression, and then decremented to 2.

What is the output of the following code:
console.log(“hello” + 3);

a) “hello3”

b) 3

c) “hello”

d) Error

Answer: a) “hello3”

Explanation: In JavaScript, the “+” operator is used for both addition and concatenation. When used with a string and a number, it performs concatenation rather than addition. Therefore, the output of “console.log(“hello” + 3)” is “hello3”.

What is the output of the following code:

console.log(true && “hello”);

a) true

b) “hello”

c) false

d) undefined

Answer: b) “hello”

Explanation: In JavaScript, the “&&” operator is used for logical AND. It returns the first falsy value it encounters, or the last truthy value if all values are truthy. In this case, both “true” and “hello” are truthy values, so the expression evaluates to “hello”.

What is the output of the following code:
var x = 5;

function foo() {

console.log(x);

var x = 10;

}

foo();

a) 5

b) 10

c) undefined

d) Error

Answer: a) 5

Explanation: In JavaScript, variable declarations are processed before any other code in the same scope, but their values are not assigned until they are reached in the code. Therefore, the “var x = 10;” statement in the “foo” function does not affect the value of “x” in the outer scope. When “foo” is called, it outputs the value of “x” in the outer scope, which is 5. The output of “console.log(x)” is therefore 5.

What is the output of the following code:
console.log(typeof NaN);

a) “number”

b) “string”

c) “undefined”

d) Error

Answer: a) “number”

Explanation: In JavaScript, NaN (Not a Number) is a special value of the number data type that represents an undefined or unrepresentable value resulting from a mathematical operation. Despite its name, NaN is still considered a number data type. Therefore, the output of “console.log(typeof NaN)” is “number”.

What is the output of the following code:
var x = [1, 2, 3];

var y = x.slice(0, 2);

console.log(y);

a) [1, 2]

b) [2, 3]

c) [1, 2, 3]

d) Error

Answer: a) [1, 2]

Explanation: In JavaScript, the “slice” method is used to extract a portion of an array into a new array. It takes two arguments: the starting index and the ending index (exclusive) of the slice. In this case, “x.slice(0, 2)” extracts the elements at indices 0 and 1 (i.e. [1, 2]) into a new array, which is assigned to “y”. Therefore, the output of “console.log(y)” is [1, 2].

What is the output of the following code:
console.log(“3” + 2);

a) “32”

b) “5”

c) 5

d) Error

Answer: a) “32”

Explanation: In JavaScript, the “+” operator is used for both addition and concatenation. When used with a string and a number, it performs concatenation rather than addition. Therefore, the output of “console.log(“3″ + 2)” is “32”.

What is the output of the following code:
console.log(null == undefined);

a) true

b) false

c) NaN

d) Error

Answer: a) true

Explanation: In JavaScript, null and undefined are equal to each other and to no other value. Therefore, the output of “console.log(null == undefined)” is true.

What is the output of the following code:
var x = 5;

function foo() {

console.log(x);

}

foo();

a) 5

b) undefined

c) null

d) Error

Answer: a) 5

Explanation: In this code, the variable “x” is declared in the global scope and assigned the value 5. The function “foo” is defined to log the value of “x”. When “foo” is called, it outputs the value of “x” in the outer scope, which is 5. Therefore, the output of “console.log(x)” is 5.

What is the output of the following code:
console.log(0.1 + 0.2 == 0.3);

a) true

b) false

c) NaN

d) Error

Answer: b) false

Explanation: In JavaScript, floating-point numbers are represented using binary fractions, which can sometimes result in rounding errors. Therefore, the expression “0.1 + 0.2” does not equal exactly 0.3, but instead equals a slightly larger number (e.g. 0.30000000000000004). Therefore, the output of “console.log(0.1 + 0.2 == 0.3)” is false.

What is the output of the following code:
var x = {name: “John”, age: 30};

console.log(Object.keys(x));

a) {name: “John”, age: 30}

b) [“name”, “age”]

c) “John,30”

d) Error

Answer: b) [“name”, “age”]

Explanation: In JavaScript, the “Object.keys” method is used to retrieve an array of an object’s own enumerable property names. In this case, “Object.keys(x)” returns an array containing the property names “name” and “age”. Therefore, the output of “console.log(Object.keys(x))” is [“name”, “age”].

What is the output of the following code:
console.log(typeof NaN);

a) “number”

b) “string”

c) “NaN”

d) “undefined”

Answer: a) “number”

Explanation: In JavaScript, NaN (Not-a-Number) is a special value of the number type. Therefore, the output of “console.log(typeof NaN)” is “number”.

What is the output of the following code:
console.log(“foo”.indexOf(“o”));

a) 0

b) 1

c) 2

d) -1

Answer: b) 1

Explanation: In JavaScript, the “indexOf” method is used to find the index of the first occurrence of a substring within a string. In this case, “foo” contains two occurrences of “o”, but “indexOf” returns the index of the first occurrence, which is 1. Therefore, the output of “console.log(“foo”.indexOf(“o”))” is 1.

What is the output of the following code:
var x = 5;

var y = x++;

console.log(x, y);

a) 5, 5

b) 6, 5

c) 6, 6

d) Error

Answer: b) 6, 5

Explanation: In this code, the variable “x” is assigned the value 5 and then incremented using the “++” operator. The variable “y” is assigned the original value of “x” (i.e. 5) before the incrementation. Therefore, the output of “console.log(x, y)” is 6, 5.

What is the output of the following code:
var x = [“a”, “b”, “c”];

x.splice(1, 1);

console.log(x);

a) [“a”, “b”, “c”]

b) [“a”, “c”]

c) [“b”, “c”]

d) Error

Answer: b) [“a”, “c”]

Explanation: In JavaScript, the “splice” method is used to add or remove elements from an array. In this case, “x.splice(1, 1)” removes one element starting from the index 1 (i.e. “b”) from the array “x”. Therefore, the output of “console.log(x)” is [“a”, “c”].

What is the output of the following code:
console.log(10 < 9 < 8);

a) true

b) false

c) NaN

d) Error

Answer: a) true

Explanation: In JavaScript, the “<” operator performs a numerical comparison between two values. Therefore, the expression “10 < 9 < 8” is evaluated as “(10 < 9) < 8”, which first evaluates “10 < 9” to false and then “false < 8” to true. Therefore, the output of “console.log(10 < 9 < 8)” is true.

What is the output of the following code:
```
var x = [1, 2, 3];

var y = x.map(function(n) { return n * 2; });

console.log(y);
```

a) [1, 2, 3]

b) [2, 4, 6]

c) [1, 4, 9]

d) Error

Answer: b) [2, 4, 6]

Explanation: In JavaScript, the “map” method is used to create a new array by applying a function to each element of an existing array. In this case, the “map” method is used to create a new array “y” where each element is twice the value of the corresponding element in the original array “x”. Therefore, the output of “console.log(y)” is [2, 4, 6].

What is the output of the following code:
```
console.log(typeof null);
```

a) “object”

b) “null”

c) “undefined”

d) “string”

Answer: a) “object”

Explanation: In JavaScript, the “typeof” operator returns the data type of a value. However, there is a known bug where “typeof null” returns “object” instead of “null”. Therefore, the output of “console.log(typeof null)” is “object”.

What is the output of the following code:
var x = 5;

var y = “5”;

console.log(x == y);

a) true

b) false

c) NaN

d) Error

Answer: a) true

Explanation: In JavaScript, the “==” operator performs type coercion before comparison. In this case, the number 5 and the string “5” are coerced to the same type (i.e. number) before comparison. Therefore, the output of “console.log(x == y)” is true.

What is the output of the following code:
var x = 5;

var y = “5”;

console.log(x === y);

a) true

b) false

c) NaN

d) Error

Answer: b) false

Explanation: In JavaScript, the “===” operator performs strict comparison without type coercion. In this case, the number 5 and the string “5” are of different types, so strict comparison will always return false. Therefore, the output of “console.log(x === y)” is false.

What is the output of the following code:
var x = [1, 2, 3];

console.log(x.toString());

a) “1,2,3”

b) [1, 2, 3]

c) [“1”, “2”, “3”]

d) Error

Answer: a) “1,2,3”

Explanation: In JavaScript, the “toString” method is used to convert an array to a string by concatenating its elements with commas. Therefore, the output of “console.log(x.toString())” is “1,2,3”.

What is the output of the following code:
var x = 5;

var y = 3;

console.log(x %= y);

a) 1

b) 2

c) 3

d) 4

Answer: b) 2

Explanation: In JavaScript, the %= operator is the modulus assignment operator, which computes the modulus of two operands and assigns the result to the left operand. Therefore, x %= y is equivalent to x = x % y, which evaluates to 2. The output of console.log(x %= y) is 2.

What is the output of the following code:
console.log(typeof []);

a) “array”

b) “object”

c) “null”

d) “undefined”

Answer: b) “object”

Explanation: In JavaScript, an array is a special type of object that can hold a collection of values. Therefore, the typeof operator returns “object” when applied to an array.

What is the output of the following code:
var x = [1, 2, 3];

x.push(4);

console.log(x);

a) [1, 2, 3]

b) [1, 2, 3, 4]

c) [4, 3, 2, 1]

d) Error

Answer: b) [1, 2, 3, 4]

Explanation: In JavaScript, the push() method adds one or more elements to the end of an array and returns the new length of the array. Therefore, after x.push(4), the array x contains the elements [1, 2, 3, 4]. The output of console.log(x) is [1, 2, 3, 4].

What is the output of the following code:
var x = “hello”;

var y = x.toUpperCase();

console.log(y);

a) “hello”

b) “HELLO”

c) “Hello”

d) Error

Answer: b) “HELLO”

Explanation: In JavaScript, the toUpperCase() method is used to convert a string to uppercase letters. Therefore, after y = x.toUpperCase(), the value of y is “HELLO”. The output of console.log(y) is “HELLO”.

What is the output of the following code:

```
var x = 5;

var y = “3”;

console.log(x + y);
```

a) 8

b) “53”

c) 53

d) Error

Answer: b) “53”

Explanation: In JavaScript, the + operator can be used for addition or string concatenation, depending on the type of the operands. If either operand is a string, the + operator performs string concatenation. In this case, the value of x is a number and the value of y is a string, so the + operator performs string concatenation and the output of console.log(x + y) is “53”.

What is the output of the following code:

```
var x = 10;

var y = 5;

console.log((x > y) && (y < 6));
```

a) true

b) false

c) TypeError

d) ReferenceError

Answer: b) false

Explanation: In JavaScript, the && operator returns true if both operands are true, and false otherwise. In this case, (x > y) evaluates to true (because 10 is greater than 5), but (y < 6) evaluates to false (because 5 is not less than 6). Therefore, the expression (x > y) && (y < 6) evaluates to false. The output of console.log((x > y) && (y < 6)) is false.

What is the output of the following code:
var x = “5”;

var y = “3”;

console.log(x + y);

a) 8

b) “53”

c) 53

d) Error

Answer: b) “53”

Explanation: In JavaScript, the + operator performs string concatenation when either operand is a string. In this case, both x and y are strings, so the + operator performs string concatenation and the output of console.log(x + y) is “53”.

What is the output of the following code:
var x = 10;

var y = “5”;

console.log(x + y);

a) 15

b) “105”

c) 105

d) Error

Answer: b) “105”

Explanation: In JavaScript, the + operator performs string concatenation when either operand is a string. In this case, y is a string and x is a number, so the + operator performs string concatenation and the output of console.log(x + y) is “105”.

What is the output of the following code:
```
var x = [1, 2, 3];

var y = [1, 2, 3];

console.log(x == y);
```

a) true

b) false

c) TypeError

d) ReferenceError

Answer: b) false

Explanation: In JavaScript, arrays are objects, and two arrays are considered equal only if they refer to the same object. In this case, x and y are two different arrays with the same elements, so the expression x == y evaluates to false. The output of console.log(x == y) is false.

What is the output of the following code:
```
var x = 10;

var y = 3;

console.log(x / y);
```

a) 3.33

b) 3.3333

c) 3.333333

d) 3

Answer: a) 3.33

Explanation: In JavaScript, the / operator performs division of two operands. When one or both operands are floating-point numbers, the result is also a floating-point number. Therefore, the expression x / y evaluates to 3.33333… (with repeating decimals), but the output of console.log(x / y) is rounded to two decimal places and displayed as 3.33.

Which of the following is not a primitive data type in JavaScript?

A) String

B) Number

C) Boolean

D) Array

Answer: D) Array

Explanation: In JavaScript, primitive data types are String, Number, Boolean, null, undefined, and symbol. Array is not a primitive data type; it is an object type.

Which of the following methods removes the last element from an array and returns that element?

A) push()

B) pop()

C) shift()

D) unshift()

Answer: B) pop()

Explanation: The pop() method removes the last element from an array and returns that element. The push() method adds one or more elements to the end of an array, while the shift() method removes the first element from an array and returns that element. The unshift() method adds one or more elements to the beginning of an array.

What is the output of the following code?

```
function foo() {

  var a = b = 3;

}

foo();

console.log("a defined? " + (typeof a !== 'undefined'));

console.log("b defined? " + (typeof b !== 'undefined'));
```

A) a defined? true, b defined? true

B) a defined? false, b defined? true

C) a defined? true, b defined? false

D) a defined? false, b defined? false

Answer: B) a defined? false, b defined? true

Explanation: The function foo() declares a variable a and assigns it the value of 3, but it also assigns the value of 3 to an undeclared variable b. This creates a global variable b with a value of 3, but a is only defined within the scope of the function. Therefore, typeof a !== 'undefined' will be true within the function, but false outside of it. typeof b !== 'undefined' will be true because it is a global variable.

_____*"The End"*_____

- "Remember me in your prayers" -

Regards:"Ismail Shah"