**Competency Unit -WPSI 100**

**Module-02**

**Batch id-ESAD-WDPF/TCLD-01M/R37/01**

**Exam Date- 02/06/2018**

**Total Marks-60**

**MCQ**

**1. Consider the following code snippet**

function printArray(a)

{

var len = a.length, i = 0;

if (len == 0)

console.log("Empty Array");

else

{

do

{

console.log(a[i]);

} while (++i < len);

}

}

**2. What does the above code result?**  
a. Prints the numbers in the array in order  
b. Prints the numbers in the array in the reverse order  
c. Prints 0 to the length of the array  
d. Prints “Empty Array”

**3. What are the three important manipulations done in a for loop on a loop variable?**  
a. Updation, Incrementation, Initialization  
b. Initialization,Testing, Updation  
c. Testing, Updation, Testing  
d. Initialization,Testing, Incrementation

**4. Consider the following code snippet**

**function tail(o)**

**{**

**for (; o.next; o = o.next) ;**

**return o;**

**}**

Will the above code snippet work? If not, what will be the error?  
a. No, this will throw an exception as only numerics can be used in a for loop  
b. No, this will not iterate  
c. Yes, this will work  
d. No, this will result in a runtime error with the message “Cannot use Linked List”

**5. Consider the following code snippet**

**for(var p in o)**

**console.log(o[p]);**

The above code is equivalent to which code?

a. for (var i = 0;i < a.length;i++)

console.log(a[i]);

b. for (int i = 0;i < a.length;i++)

console.log(a[i]);

c. for (var i = 0;i <= a.length;i++)

console.log(a[i]);

d. for (var i = 1;i < a.length;i++)

console.log(a[i]);

**6. One of the special feature of an interpreter in reference with the for loop is that**a. Before each iteration, the interpreter evaluates the variable expression and assigns the name of the property  
b. The iterations can be infinite when an interpreter is used  
c. The body of the loop is executed only once  
d. All of the mentioned  
**7. What will happen if the body of a for/in loop deletes a property that has not yet been enumerated?**  
a. The property will be stored in a cache  
b. The loop will not run  
c. That property will not be enumerated  
d. All of the mentioned  
**8. What will be the step of the interpreter in a jump statement when an exception is thrown?**  
a. The interpreter stops its work  
b. The interpreter throws another exception  
c. The interpreter jumps to the nearest enclosing exception handler  
d. None of the mentioned  
**9. Consider the following code snippet**

**while (a != 0)**

**{**

**if (a == 1)**

**continue;**

**else**

**a++;**

**}**

What will be the role of the **continue** keyword in the above code snippet?  
a. The continue keyword restarts the loop  
b. The continue keyword skips the next iteration  
c. The continue keyword skips the rest of the statements in that iteration  
d. None of the mentioned  
**10. Consider the following code snippet**

**function f(o)**

**{**

**if (o === undefined) debugger;**

**}**

What could be the task of the statement debugger?  
a. It does nothing but a simple breakpoint  
b. It debugs the error in that statement and restarts the statement’s execution  
c. It is used as a keyword that debugs the entire program at once  
d. All of the mentioned  
**11. Among the keywords below, which one is not a statement?**  
a. debugger  
b. with  
c. if  
d. use strict  
View Answer

This set of Javascript Multiple Choice Questions & Answers (MCQs) focuses on “Statements”.

**12. JavaScript is a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ language**  
a. Object-Oriented  
b. High-level  
c. Assembly-language  
d. Object-Based  
View Answer

**13. The output for the following code snippet would most appropriately be**

var a=5 , b=1

var obj = { a : 10 }

with(obj)

{

alert(b)

}

a. 10  
b. Error  
c. 1  
d. 5

**14. A conditional expression is also called a**  
a. Alternate to if-else  
b. Immediate if  
c. If-then-else statement  
d. None of the above

**15. Which is a more efficient code snippet ?**  
Code 1 :

for(var num=10;num>=1;num--)

{

document.writeln(num);

}

Code 2 :

var num=10;

while(num>=1)

{

document.writeln(num);

num++;

}

a. Code 1  
b. Code 2  
c. Both Code 1 and Code 2  
d. Cannot Compare  
**16. A statement block is a**  
a. conditional block  
b. block that contains a single statement  
c. Both a and b  
d. block that combines multiple statements into a single compound statement  
**17. When an empty statement is encountered, a JavaScript interpreter**  
a. Ignores the statement  
b. Prompts to complete the statement  
c. Throws an error  
d. Throws an exception

**18. The “var” and “function” are**  
a. Keywords  
b. Declaration statements  
c. Datatypes  
d. Prototypes

**19. Consider the following statements**

switch(expression)

{

statements

}

**In the above switch syntax, the expression is compared with the case labels using which of the following operator(s) ?**  
a. ==  
b. equals  
c. equal  
d. ===

**20. Consider the following statements**

**var count = 0;**

**while (count < 10)**

**{**

**console.log(count);**

**count++;**

**}**

**In the above code snippet, what happens?**  
a. The values of count is logged or stored in a particular location or storage.  
b. The value of count from 0 to 9 is displayed in the console.  
c. An error is displayed  
d. An exception is thrown

. **20. The enumeration order becomes implementation dependent and non-interoperable if :**  
a. If the object inherits enumerable properties  
b. The object does not have the properties present in the integer array indices  
c. The delete keyword is never used  
d. Object.defineProperty() is not used

**21. Consider the following code snippet :**

var grand\_Total=eval("10\*10+5");

The output for the above statement would be :  
a. 10\*10+5  
b. 105 as a string  
c. 105 as an integer value  
d. Exception is thrown  
**22. Do functions in JavaScript necessarily return a value ?**  
a. It is mandatory  
b. Not necessary  
c. Few functions return values by default  
d. All of the above  
**23. Consider the following code snippet :**

var tensquared = (function(x) {return x\*x;}(10));

Will the above code work ?  
a. Yes, perfectly  
b. Error  
c. Exception will be thrown  
d. Memory leak  
**24. Consider the following code snippet :**

var string2Num=parseInt("123xyz");

The result for the above code snippet would be :  
a. 123  
b. 123xyz  
c. Exception  
d. NaN  
**25. The one-liner code that concatenates all strings passed into a function is**

a. function concatenate()

{

return String.prototype.concat('', arguments);

}

b. function concatenate()

{

return String.prototype.apply('', arguments);

}

c. function concatenate()

{

return String.concat.apply('', arguments);

}

d. function concatenate()

{

return String.prototype.concat.apply('', arguments);

}

**26. If you have a function f and an object o, you can define a method named m of o with**  
a. o.m=m.f;  
b. o.m=f;  
c. o=f.m;  
d. o=f;  
**27. For the below mentioned code snippet:**

var o = new Object();

The equivalent statement is:

a. var o = Object();

b. var o;

c. var o= new Object;

d. Object o=new Object();

**28. What is the difference between the two lines given below ?**

**!!(obj1 && obj2);**

**(obj1 && obj2);**

a. Both the lines result in a boolean value “True”  
b. Both the lines result in a boolean value “False”  
c. Both the lines checks just for the existence of the object alone  
d. The first line results in a *real* boolean value whereas the second line merely checks for the existence of the objects  
**29. Consider the following code snippet :**

**var c = counter(), d = counter();**

c.count()

d.count()

c.reset()

c.count()

d.count()

The state stored in d is :  
a. 1  
b. 0  
c. Null  
d. Undefined  
**30. Consider the following code snippet :**

**function constfuncs()**

**{**

**var funcs = [];**

**for(var i = 0; i < 10; i++)**

**funcs[i] = function() { return i; };**

**return funcs;**

**}**

**var funcs = constfuncs();**

**funcs[5]()**

**What does the last statement return ?**  
a. 9  
b. 0  
c. 10  
d. None of the above

**31. What is the Output of this code?**

**J=5;**

**J++**

**++J**

**document.write(j)**

1. 5
2. 6
3. 8
4. 7

**32. What is the Output of this Code?**

**K=3**

K++

K--

--K

**document.write(K)**

1. 1
2. 3
3. 4
4. 2