Answers 1 to 18 – Akash Verma

Ans-1- d. frameset

Ans-2- a. aside

Ans-3- Option not given :: correct answer -> crossOrigin

Ans-4- b. It defers script execution until the page has been rendered.

Ans-5- c. pseudo-class

Ans-6- b. flex-flow

Ans-7- b. parent to child

Ans-8- c. lexical

Ans-9- a. setTimeout()

Ans-10- d. navigator

Ans-11- The return statement in the 2-function will just return undefined as the object is in the next line which will just be ignored whereas 1-function will return the object.

Ans-12- output : 0 2 2

This happens because of the property of the Increment operators :

- number++ -> means use then increment.
- ++number -> means increment then use.

Ans-13- output : 1 2 4

As the **if** statement will continue the loop and it will never go to the **console.log** command for the value of 3 and hence all the other values are logged which are in the range of 1 to 5.

Ans-14- output : 'string'

As the first **typeof statement** will return the type **Number** as a string and then the second **typeof statement** will return the type of the Number string Which will come out as **String**.

Ans-15- A closure is a feature in JavaScript where an inner function has access to the outer (enclosing) function's variables i.e. a scope chain.
The closure has three scope chains:

- it has access to its own scope.
- it has access to the outer function's variables.
- it has access to the global variables

example :

```
function outer() {
        var b =  10;
        function inner() {
                var a = 20;
                console.log(a + b);
        };
        return inner;
};
```

Here, The scope of variable b is to both the functions whereas the scope of variable a is limited to the inner function.

Ans-16- code snippet after 1-pass:

```
var first_name, last_name, result;
function concat(x, y) {
        return x + y;
}
result =  concat(first_name, last_name);
first_name = “John”;
last_name = “Doe”;
```

Ans-17- Anonymous functions are used where we don’t intend to use them again so they don’t get an identifier and just defined without one.
Generally Anonymous functions are given to functions that takes a function as an argument.
Example :

```
Array.forEach( (element) => {     // anonymous
        Console.log(element);
});
```

Callback functions are used where we want to pass the whole function definition as an argument which can be invoked anywhere in the given function, it is passed to.

Callback function is different as the argument that receives the callback function becomes its identifier and then the function can be invoked again and again using that identifier.

Example :

```
function do(x, callback) {   // callback
        Callback(x);
};
do('Akash', (x) => {console.log(x);} );
```

Ans-18- Event Bubbling : When an event happens on an element, it first runs the handlers on it, then on its parent, then all the way up on other ancestors.

Let's say we have 3 nested elements form > div > p with a handler on each of them:

Then,

A click on the inner <p> first runs 'onclick':

- On that <p>.
- Then on the outer .
- Then on the outer<form>.
- And so on upwards till the 'document' object.

Stopping Condition :

- To stop event bubbling all we have to do is to target the event and call the method 'e.stopPropagation()' inside the event listener's function.