**Module 23 Notes**

Video 1 –

1. I’ll learn few basic things or format or rule that should be maintained during coding. These will also be helpful for interview because these rules are common questions.
2. Creating a new repository named “advanced-javascript” to store files that I’ll create in this module and linking the local folder.

Video 2 –

1. Code –

const age = 4;

if(age){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

Condition is true

This loop will show any negative or positive number as “condition is true” but if the number is 0 than it’ll show “condition is false”.

1. To check if there any string as value –

const name = "Solaiman";

if(name.length > 0){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

Condition is true

I can also do the same thing with just writing “name.length” or “name” as condition inside the for loop.

1. **If I do not define anything as value of the variable then the condition will become false –**

let name;

console.log(name);

if(name){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

undefined

Condition is false

1. null is also a false value –

let name = null;

console.log(name);

if(name){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

null

Condition is false

1. NaN or not a number is also a false value.
2. Empty array is a truthy value –

let name = [];

console.log(name);

if(name){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

Condition is true

1. Empty object is also a true value –

let name = {};

1. Value “false” is a false value –

let name = false;

console.log(name);

if(name){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

false

Condition is false

1. But if I write “false” meaning write false with double or single quotation then it is true because then it becomes a string.
2. Zero (0) is a negative number. This can be confusing sometimes.
3. **This true and false values are important when writing conditions –**

let name = 0;

if(name || name == 0){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

Condition is true

**And if it is a number –**

let name = 1;

if(name || name == 0){

console.log("Condition is true");

}

else{

console.log("Condition is false");

}

Output –

Condition is true

Video 3 –

1. If there is no value of the variable then it’ll show undefined –

let name;

console.log(name);

Output –

undefined

1. If I do not return the value of a function then it’ll show undefined.

function add(num1, num2){

sum = num1 + num2;

console.log(sum);

}

const result = add(13, 82);

console.log(result);

Output –

95

undefined

1. Even if I write only “return” and do not specify what to return, it’ll show undefined.
2. If I do not pass the value of a parameter then it’ll show undefined –

function add(num1, num2){

console.log(num1, num2);

}

const result = add(13);

console.log(result);

**Output –**

13 undefined

undefined

In this case we can set a default value.

1. If I try to access a property of an object which does not exist then it’ll show undefined.
2. If I set the value “undefined” then it’ll show undefined –

let fun = undefined;

console.log(fun);

Output –

undefined

1. Null means there is no value at all and it can be set as a value. Undefined may occur if I do not follow the rule or format of coding.

Video 4 –

1. When I compare 2 elements with double equal then it won’t check what type of value it is. But when I compare it with triple equal then it’ll check whether it is numeric, string or object etc.
2. Double equal (==) – checks value

Triple equal (===) – checks value and value type

1. In a more advanced way to explain it – double equal first catches the value then checks if the values are same type or not, if not same then it converts one to make them same type and then compares them.
2. JavaScript considers –

0 = false

1 = true

1. So if I compare true with 1 using double equal then it’ll show as “condition is true”. But it’ll show “condition is false” if I use triple equal.

const first = 1;

const second = true;

if(first == second){

console.log('Condition is true');

}

else{

console.log('Condition is false');

}

Video 5 –

1. I can either write the function inside the map or outside –

**Outside the map –**

function square (element){

return element \* element;

}

numbers.map(square)

**Inside the map-**

numbers.map(function(element){

return element \* element;

})

1. I can pass 3 parameters inside the function which is inside the map. –

const numbers = [3, 4, 5, 6, 7, 8];

numbers.map(function(element, index, array){

console.log(element, index, array);

})

Output –

3 0 [ 3, 4, 5, 6, 7, 8 ]

4 1 [ 3, 4, 5, 6, 7, 8 ]

5 2 [ 3, 4, 5, 6, 7, 8 ]

6 3 [ 3, 4, 5, 6, 7, 8 ]

7 4 [ 3, 4, 5, 6, 7, 8 ]

8 5 [ 3, 4, 5, 6, 7, 8 ]

3 parameters are – element, index number and the whole array.

1. I can do the square math calculation with a for loop or with map –

**For loop –**

const numbers = [3, 4, 5, 6, 7, 8];

const output = [];

for(let i = 0; i < numbers.length; i++){

const element = numbers[i];

const result = element \* element;

output.push(result);

}

console.log(output);

Output –

[ 9, 16, 25, 36, 49, 64 ]

**With map –**

const numbers = [3, 4, 5, 6, 7, 8];

const result = numbers.map(function(element, index, array){

return element \* element;

})

console.log(result);

Output –

[ 9, 16, 25, 36, 49, 64 ]

1. So map method can be applied on an array.
2. More easier way to do this square calculation with arrow function –

const square = element => element \* element;

const result = square(5);

console.log(result);

Output –

25

More easier –

const square = x => x \* x;

const result = square(5);

console.log(result);

Output –

25

1. Arrow function inside map –

const numbers = [3, 4, 5, 6, 7, 8];

const result = numbers.map(x => x \* x);

console.log(result);

Output –

[ 9, 16, 25, 36, 49, 64 ]

1. I can filter elements according to a condition of an array with filter method –

const numbers = [3, 4, 5, 6, 7, 8];

const bigger = numbers.filter(x => x > 5);

console.log(bigger);

Output –

[ 6, 7, 8 ]

This will show all the elements which are bigger than 5 in the numbers array.

Filter will always give an array. If there is no element which fulfils the condition it will still give an empty array.

1. More things to learn about – for each, reduce (advance level).
2. I can also use find method to find any element with a specific condition –

const numbers = [3, 4, 5, 6, 7, 8];

const isThere = numbers.find(x => x > 5);

console.log(isThere);

Output –

6

Important thing to note here is find method will only give one element which is the most closest. Even if there is more element which fulfils the condition it’ll still give one element.

1. Map loops through the whole array and does the calculation or what it is supposed to do and then displays the result as an array.
2. Filter will also give an array but find will only give an element.

Video 6 –

1. Finding the value of a property from an array of objects –

const students = [

{id: 21, name: 'Omar Sunny'},

{id: 31, name: 'Manna'},

{id: 41, name: 'Leonardo DiCaprio'},

{id: 71, name: 'Tom Hanks'}

];

let studentsName = [];

for(let i = 0; i < students.length; i++){

let name = students[i].name;

studentsName.push(name);

}

console.log(studentsName);

Output –

[ 'Omar Sunny', 'Manna', 'Leonardo DiCaprio', 'Tom Hanks' ]

1. **Same things but with arrow function which makes it much more easier –**

const students = [

{id: 21, name: 'Omar Sunny'},

{id: 31, name: 'Manna'},

{id: 41, name: 'Leonardo DiCaprio'},

{id: 71, name: 'Tom Hanks'}

];

const names = students.map(s => s.name);

console.log(names);

Output –

[ 'Omar Sunny', 'Manna', 'Leonardo DiCaprio', 'Tom Hanks' ]