**FSD - Assignment – 2**

Q1. **[1 Point]** Explain the difference between null and undefined in JavaScript.

**Undefined** means that a variable has been declared but has not been given a value yet.

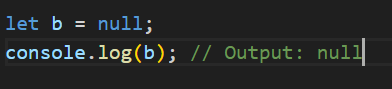
Example:



Here, a is declared, but we didn't give it a value. So, JavaScript automatically assigns undefined to a.

**Null** is a value that you can assign to a variable to show that it should be empty or have no value.

Example:



Here, null is assigned to b, which means we've chosen to give it no value.

Q2. **[1 Point]** What will be the output of the following code snippet, and why? 

**console.log('10'+5) // output: 105**

* Operator when used with a string undergoes concatenation. Therefore converts number 5 to string and joins with 10-> 105

**console.log('10'-5) //output: 5**

* Operator is a mathematical operator and tries to convert string 10 to number and performs subtraction for 10 and 5-> 5

**console.log(true+2) //output: 3**

Here, true is a boolean value which results to number 1 and adds (+) with 2-> 1+2=3

**console.log(false+undefined); //output: NaN**

Here, false is boolean which is treated as 0 mathematically, and undefined is given as a number but it’s not a number. Therefore gives NaN i.e. not a number.

Q3. **[1 Point]** What is the difference between == and === in JavaScript? Provide examples.

We use “==” when we want to compare two values for equality with type conversion.



We use “===” when we want to compare two values for equality, ensuring that both value and type match.



Q4. **[1 Point]** Predict the output of the following expressions and explain your reasoning: 

**console.log(0==false)**

Here, false is treated as 0, hence 0==0 is true

**console.log(0===false)**

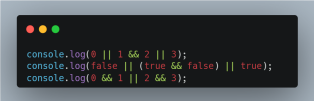
Here , o is number and false is boolean, hence two types are different. Therefore, false

**console.log(''==0)**

Here, ‘ ’ is an empty string and is treated as 0, hence 0==0 which results to true.

**console.log(''===0)**

Here, ‘ ‘ is an empty string and 0 is a number, therefore the data types are different and don't match. Hence, false

Q5. **[1 Point]** Given the following code, what will be the output and why? 

X Y X&&Y X||Y

0 0 0 0

0 1 0 1

1 0 0 1

1 1 1 1

Note: According to associative and precedence, && ha higher precedence than ||

**console.log(0 || 1 && 2 || 3)**

-> 1 && 2 is true -> 2 (because it’ll return the recently visited number)

->0||2||3 -> 0||2 is true-> 2 and it’ll not check with 3 because true || true is true

Therefore, it’ll return 2

**console.log(false || (true && false) || true)**

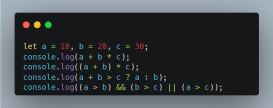
-> true && false-> false

false||false||true-> false||false is false, false||true is true

**console.log(0 && 1 || 2 && 3)**

-> 0 && 1 is 0 because 0 && any number is 0, 2 && 3 is 3

-> 0 || 3-> 3 (true)

Q6. **[1 Point]** Predict the output of the following expressions and explain your reasoning:

Follows BODMAS rule

**let a=10, b=20, c=30**

**console.log(a+b\*c) //610**

->b\*c= 20\*30=600

->a+600=10+600=610

**console.log((a+b)\*c) //900**

->a+b=10+20=30

->30\*c=30\*30=900

**console.log(a+b>c?a:b) //20**

->a+b=30, c=30; 30>30 is false

Therefore, value of b is returned = 20

**console.log((a>b) && (b>c) || (a>c)) //false**

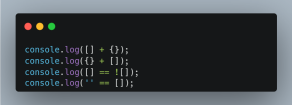
&& has higher precedence than ||

->a>b=10>20 is false, b>c=20>30 is false

-> false && false is false

->a>c= 10>30 is false

Therefore, false || false is false

Q7. **[2 Points]** Analyze and explain the output of the following code snippets : 

**console.log([] + {}) // output: [object Object]**

-> [ ] is empty array which is converted to an empty string and { } is an empty object which also converts to a string but results to [object Object]

**console.log({} + []) // output: [object Object]**

[] is converted to an empty string "".

{} is converted to the string "[object Object]".

**console.log([]== ![]) // true**

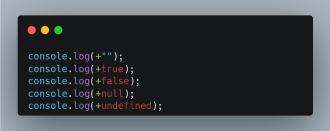
-> [ ] this empty array is converted to an empty string “ “

-> [ ] is a true value and ![ ] will be false. This false is also treated as “ “ when compared with a string.

Therefore, “ “ == “ “ results as true

**console.log('' == []) //true**

-> empty array [ ] is converted to an empty string “ “ and compared with another empty string which results as true

Q8. **[2 Points]** What will be the output of the following code, and why?

**console.log(+ "") // output: 0**

+” “ clearly is being treated as a number as a unary operator is used, " " is treated as 0. +0 is nothing but 0 itself.

**console.log(+true) // output: 1**

Here true is converted to 1as unary operator is used, +1 is nothing but 1

**console.log(+false) // output: 0**

Here false is treated as 0 as it is being represented as a number as a unary operator is used, hence +0 is 0.

**console.log(+null) //output: 0**

Here null is treated as 0, hence +0 is 0

**console.log(+undefined) // output: NaN**

Here undefined cannot be converted to a number hence it results as Nan.