1. Variables

<script>

*// valid variable names*

const my\_firstName = "Mohit";

const \_myLastName = "Verma";

const \_123Age = 25;

const $country = "India";

*// invalid variable names*

const 123Age = 25;

const @my@country = "India";

</script>

2. Data types

<script>

*// data types*

*// =============*

*// string*

let myName = "Mohit";

console.log(typeof myName);

*// number*

let myAge = 25;

console.log(typeof myAge);

*// boolean*

let todayIsHoliday = true;

console.log(typeof todayIsHoliday);

*// undefined*

let subject;

console.log(typeof subject);

*// null*

let testNul = null;

console.log(testNul);

*// convert string to number*

let numTest = "100";

console.log(numTest);

console.log(typeof numTest); *// Output = string*

console.log(typeof +numTest); *// Output = number*

console.log(typeof Number(numTest)); *// Output = number*

console.log(typeof parseInt(numTest)); *// Output = number*

*// convert number to string*

let stringTest = 500;

console.log(stringTest);

console.log(typeof stringTest); *// Output = number*

console.log(typeof String(stringTest)); *// Output = string*

console.log(typeof (stringTest + "")); *// Output = string*

</script>

3. Truthy falsy

<script>

*// truthy falsy*

let today = "Sunday";

if (today == "Sunday") {

console.log("Today is Holiday");

} else {

console.log("Go to office");

}

</script>

4. parseInt VS parseFloat

parseInt(count) = converts the floating-point number 100.5 to an integer, so it returns 100.

It discards the decimal part and any characters after it because parseInt() stops parsing as soon as it encounters the decimal point.

parseFloat(count) = it retains the decimal portion of the number.

In your example, parseFloat(count) returns 100.5, as it correctly parses the string 100.5 into a floating-point number and retains the decimal part.

<script>

*// Differnce between parseInt & parseFloat*

let count = 100.5;

console.log(count);

console.log(typeof count); *//output = string*

console.log(parseInt(count)); *//output = 100*

console.log(parseFloat(count)); *//output = 100.5*

</script>

5. NaN

NaN is often the result of arithmetic operations involving non-numeric operands, or when parsing strings that cannot be converted into numbers.

It's important to note that NaN is not equal to any other value, including itself.

You can check for NaN using the isNaN() function:

<script>

*// NaN in js - not a number*

let isNumber = "Hello";

console.log(isNaN(isNumber)); *//output = true*

console.log(parseInt(isNumber)); *//output = NaN*

</script>