JavaScript Data Type CheatSheet

Data Type

- Data type means which type of data.
- Defines the type of variable.
- It can be number, string, boolean, etc.
- There are 2 types of data types in JS.
- Primitive data type
- Non-primitive data type

Primitive

- Primitive values are immutable data types.
- That means it can’t be changed/altered.
- Like, string, number, bigint, boolean,null and undefined.

String

- A string is a series of characters.
- It is a set of characters.
- It represents textual data.
- JavaScript strings are immutable.
- JavaScript strings are case-sensitive.
- That means we can’t change the string.
- Must be surrounded by double or single quotes or backticks.

number

- Represents the interger/floating numbers.
- There are also 3 special number in JS.
- +infinity, -infinity, NaN
- + & - infinity simple as maths.
- -infinity 0 +infinity
- NaN stands for not a number.
- 23 * ”Ajay” is NaN

Undefined

- Which variable has no value.
- It can be a data type & a value.
- A variable without value.
- An empty variable is undefined.
- That means not initialized.

null

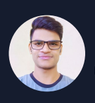
- null means nothing.
- null value means there is no value.
- We can forcefully assign a "null" value to a variable for creating an empty variable.
- null = 0 = “ “ = nothing
- It displayed object as the data type of the "null" which is a ***bug*** in JavaScript.

boolean

- It represents only two values true and false.
- Every non-zero values are true.
- All values are true except falsy values.
- Falsy values are 0, “ “, null, undefined, NaN & false as well.
- true = 1 = on
- false = 0 = off

bigInt

- There are 2 "number" data types in javascript.
- number and bigInt data type.
- It represents the big interger value.
- Like 21342342343324234234 24n


Ajay Yadav
@ATechAjay

JavaScript Variable & Operator

Keywords

- let is a reserved keyword for declaring a variable.
- We can’t use keywords as a name of the variable.

Expression

- It is a combination of operators and operands.
- Returns the single value to the variable.

Operand

- It is actual value.
- which is being manipulated using an operator is called an operand.

Variable Name

- x is the name of the variable.
- The The result of the expression is stored in the variable.
- Value is stored in a variable.

Operator

- It perform actions on values or variables.
- It is used to manipulate values or variables


Ajay Yadav
@ATechAjay

JavaScript Loops CheatSheet

for-in loop

🌱 The for-in loop allows us to access the property of an object without knowing the specific name of that object's property.

🌱 A different item is assigned to a variable on each iteration.

```
for in loop

for (const key in object) {
        // Line of code
}
```

🌱 key - It's an individual key of the object that will be a variable.

🌱 object - The object that is going to be iterated.

```
example of "for in" loop

let objectName = {
        firstName: "Ajay",
        lastName: "Yadav"
}

for (const propertyName in objectName) {
        console.log(propertyName);
        //firstName lastName
        console.log(objectName[propertyName]);
        //Ajay Yadav

}
```

🌱 We can also iterate an array using for-in loop.

```
example of "for in" loop with an array

let arr = [10, 20, 30, 40];
for (const index in arr) {
        console.log(index);
        //0 1 2 3
        console.log(arr[index]);
        // 10 20 30 40
}
```

for-of loop

🌱 The for-of statement creates a loop over iterable elements like strings, arrays, or array-like elements that are NodeList, arguments, etc.

```
"for of" loop

for (const item of arr) {
        //Line of code
}
```

🌱 item - It's an individual element of the iterable object.

🌱 arr - It's an iterable object (array, string, NodeList, etc.) whose elements are iterated.

```
Example of "for of" loop

let arr = [10, 20, 30, 40, 50];
for (const item of arr) {
        console.log(item);
        //10 20 30 40 50
}
```

while loop

🌱 The while loop iterates a block of code until the condition of the while loop is true.

```
while loop

while (condition) {
        //Line of code
        incre / decre
}
```

🌱 If we omit the increase or decrease variable in the loop, then it will iterate infinitly.

```
Infinite while loop

let a = 0;
while (a < 5) {
        console.log("Hello");
        //infinite
}
```

🌱 That's why we should use an increment or decrement variable.

```
Example of while loop

let a = 0;
while (a < 5) {
        console.log("Hello");
        //Hello Hello Hello Hello Hello
        a++;
}
```

forEach loop

🌱 forEach( ) method iterates all items of an array automatically.

🌱 forEach( ) method calls a function for each element in an array.

🌱 This is not executed for the empty array element.

```
forEach( )

arr.forEach(function (value, index) {
        //Lines of code
});
```

🌱 Array elements or NodeList can be iterated using forEach( ) method.

```
Examples of forEach( )

let arr = [10, 20, 30, 40, 50];

arr.forEach(item => console.log(item));
// 10 20 30 40 50

arr.forEach(function (value, index) {
        console.log(value);
        //10 20 30 40 50
        console.log(index);
        //0 1 2 3 4
});
```

for loop

🌱 There are 3 parts of the for loop.

```
for loop

for (initialization, condition, incre/decre) {
        //Lines of code
}
```

🌱 initialization - initialization expression is executed at least one time.

🌱 condition - It's a condition of the block of code of the for loop.

🌱 incre/decre - This expression is executed after the code of the block has been executed.

🌱 If we omit these expressions, then the block of code of the for loop will be executed infinitely.

```
Infinite loop

for ( ; ; ) {
        console.log("Hello");
        //infinite
}
```

```
for loop

for (let a = 0; a < 5; a++) {
        console.log(a);
        //0 1 2 3 4
}
```

do while loop

🌱 The do while loop is executed block of code at least once.

🌱 Don't forget to add an increase or decrease variable in the loop, otherwise the loop will be iterated infinitely.

```
do while loop

do {
        //Line of code
        incre / decre;
} while (condition);
```

```
Example of do while loop

let a = 0;
do {
        console.log(a);
        //0 1 2 3 4
        a++;

} while (a < 5);
```

🌱 The do while loop is executed at least once if the condition will be false.

```
Example of do while loop

let a = 2;
do {
        console.log(a);
        //2
        a++;

} while (a < 1);
```

NOTE

- A loop is used to iterate a block of code repeatedly until the condition is true.
- If the condition is false, then the loop will be stopped.

Ajay Yadav
@ATechAjay

Map Object CheatSheet

🌱 The Map( ) object is also used to store key-value pairs but we can specify any type of primitive key-value pairs.

🌱 Map( ) object is a collection of unique key-value pairs.

🌱 We can create an empty Map( ) object using new and Map( ) constructor.

Empty Map( ) Object

```
//Create an empty Map object
let myMap = new Map();
```

🌱 Now we can add key-value pairs to the Map( ) object using 2 ways.

1. Using set( ) method.

Adding elements to the Map( ) object

```
//Adding the value to the Map object
myMap.set("firstName", "Ajay");
myMap.set(10, "num value");
myMap.set(true, "boolean value");
```

2. Using square brackets.

Adding elements to the Map( ) object

```
//Adding the value to the Map object
//using square brackets.
let myMap = new Map(
          [
                    ["firstName", "Ajay"],
                    [10, "num value"],
                    [true, "boolean value"],
          ]
);
```

🌱 Once, we set the key-value pairs to the Map( ) object, provides methods that are used to manipulate Map( ) object.

💎 get( ) - Returns the value from the Map( ) object.

Getting values

```
//Getting the value from the Map object
console.log(myMap.get("firstName")); //Ajay
console.log(myMap.get(10));   //num value
console.log(myMap.get(true)); //boolean value
```

💎 size - Returns the size of the Map( ) object.

size of the Map( ) object

```
// Getting the size of the Map object
console.log(myMap.size); //3
```

💎 keys( ) - Returns the keys from the Map( ) object.

Getting keys

```
// Getting the keys from the Map object
console.log(myMap.keys());
//[Map Iterator] { 'firstName', 10, true }
```

💎 values( ) - Returns the values from the Map( ) object.

Getting values

```
// Getting the values from the Map object
console.log(myMap.values());
```

💎 delete( ) - Remove the key-value pairs from the Map( ) object.

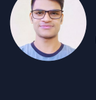
Delete key-value pair

```
//Deleting the key-pairs from the Map object
myMap.delete(10); //Remove 10 and its value
```

💎 has( ) - Check the availablibilty of the key-value pairs in the Map( ) object.

Check the availability

```
//Check the availability of the key-value
//pairs in the Map() object
console.log(myMap.has(10)); //false
```



Ajay Yadav
@ATechAjay

String Methods CheatSheet

str.slice( )

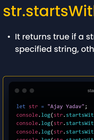
- It returns nothing(empty) if the start index is bigger index.
- It treats the NaN parameter as a 0.
- When we give a -ve parameter to it then it counts backward from the end of the string to find the index.

str.substring( )

- If the start parameter is greater than the end parameter then it swaps that parameters.
- It treats the negative parameter as 0.

str.trim( )

- The trim( ) method is used to remove whitespace from both ends of the string.
- To remove whitespace from the beginning and from the end of the string then we can use trimStart( ) & trimEnd( ) methods respectively.



str.concat( )

- The concat( ) method is used to merge two or more strings and returns a new string.
- We can join multiple string like str.concat(str1, str2, str3, ... strN);

str.toLowerCase( )

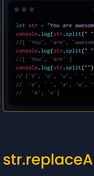
- This method is used to convert all string characters into lowercase letters.
- If the string is undefined or null then we'll get an error.

str.toUpperCase( )

- This method is used to convert all string characters into uppercase letters.
- If the string is undefined or null then we'll get an error.

str.startsWith( )

- It returns true if a string starts with a specified string, otherwise returns false.



str.endsWith( )

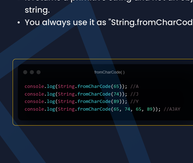
- It returns true if a string ends with a specified string, otherwise returns false.

str.repeat( )

- It returns the copied versions of the original string.
- We can pass an integer number between 0 and +infinity times to repeat the string.
- If we pass the -ve number, then we will get 'RangeError' as an output.
- Also, if we pass a fractional number, then it will round a number down to the nearest integer.

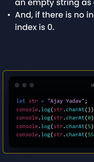
str.valueOf( )

- It is used to return the value of the given primitive string or a string object.
- This method doesn't accept any parameters.



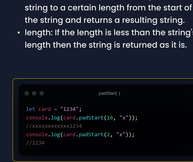
str.split( )

- It is used to divide a string into an array by a separator.
- If we omit the separator then it returns the entire string.
- Spaces has been removed from the string after using the split( ) method.



str.includes( )

- The includes( ) method is used to check if a string contains another string or not.
- It returns true if the desired string is found in the original string otherwise returns false.



str.replaceAll( )

- This method is recently added ES2021 version of JavaScript.
- It replaces all occurrences of a substring with a new string.
- Substring is a part of the original string

str.lastIndexOf( )

- It is used to find the last occurrence of a substring within a string.
- It also returns (-1) if the substring doesn't exist within a string.

str.indexOf( )

- It is used to find the index of a substring within a string.
- Keep in mind that it returns the index of the first occurrence of a substring within a string.
- And returns (-1) if the argument doesn't exist in the string.

str.fromCharCode( )

- This method converts Unicode values to characters.
- It returns a primitive string and not an object string.
- You always use it as 'String.fromCharCode( )'.

str.charCodeAt( )

- It returns the Unicode(ASCII) of the character in a string.

str.padEnd( )

- It is used to insert(pad) a string with another string to a certain length to the end of the string and returns a resulting string.

str.charAt( )

- It is used to return the characters from the string at the specified index.
- If the index is out of range, then it will return an empty string as an output.
- And, if there is no index, then the default index is 0.

str.padStart( )

- It is used to insert(pad) a string with another string to a certain length from the start of the string and returns a resulting string.
- length: If the length is less than the string's length then the string is returned as it is.

NOTE

- Searching values that will be passed to string methods are case-sensitive.
- String methods don't change the original string because the string is the immutable data type.

Ajay Yadav
@ATechAjay