**Code structure:**

**Statements:**

Statements are syntax constructs and commands that perform actions.

For example:

alert('Hello'); alert('World');

**Semicolons:**

A semicolon may be omitted in most cases when a line break exists.

For ex:

alert('Hello')

alert('World')

**Comments:**

Single Line comment:

// This comment occupies a line of its own

alert('Hello');

alert('World'); // This comment follows the statement

multiline Comment:

/\* An example with two messages.

This is a multiline comment.

\*/

**Use strict Mode:**

"use strict"; Defines that JavaScript code should be executed in "strict mode".

It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.

The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

With strict mode, you can not, for example, use undeclared variables.

You can use strict mode in all your programs. It helps you to write cleaner code, like preventing you from using undeclared variables.

"use strict" is just a string, so IE 9 will not throw an error even if it does not understand it.

**Variable:**

A variable is a “named storage” for data. We can use variables to store goodies, visitors, and other data.

Create variable using: Let

For example

Ex. let x;//declare variable

X=10//store value or assign value

Multiple variable in single line:

Ex: let user = “Megha” , subject=”js”;

Create variable using: var=> The var keyword is almost the same as let. It also declares a variable

Ex : Var message = “hello world”

Difference between let and var We can re-declare variable while use var but not doing same while use let:

We cannot re-declare variable in let

Ex let x=10;

Let x=20;//show error

**[Constants](https://javascript.info/variables" \l "constants):** Variables declared using const are called “constants”. They cannot be reassigned ex: const pi = 3.14;

# **Data types:**

A value in JavaScript is always of a certain type. For example, a string or a number.Programming languages that allow such things, such as JavaScript, are called “dynamically typed”, meaning that there exist data types, but variables are not bound to any of them.

Ex: let x = 10; x = “hello”

**1.Number=>**

The number type represents both integer and floating point numbers.

There are many operations for numbers, e.g. multiplication \*, division /, addition +, subtraction -, and so on.Besides regular numbers, there are so-called “special numeric values” which also belong to this data type: Infinity, -Infinity and NaN.

Infinity represents the mathematical Infinity ∞. It is a special value that’s greater than any number.

Ex: alert(1/0) ;

NaN represents a computational error. It is a result of an incorrect or an undefined mathematical operation, for instance:

alert( "not a number" / 2 );

**2.BigInt=>**

In JavaScript, the “number” type cannot safely represent integer values larger than (253-1) (that’s 9007199254740991), or less than -(253-1) for negatives.

Example const data= 787798787987n;

**3.String=>**

A string in JavaScript must be surrounded by quotes.

Ex: let name= “megha”;

**3.Boolen=>**

The boolean type has only two values: true and false.

This type is commonly used to store yes/no values: true means “yes, correct”, and false means “no, incorrect”.

Ex: let isGreter = 4>1;

Alert(isGreter);

**4.The null value=>**

The special null value does not belong to any of the types described above.

Let marks= null;

**5.The undefined value=>**

Ex:let marks;

Alert(marks);

**6.Objects and Symbols=>**

The object type is special.

All other types are called “primitive” because their values can contain only a single thing (be it a string or a number or whatever). In contrast, objects are used to store collections of data and more complex entities. Being that important, objects deserve a special treatment. We’ll deal with them later in the chapter Objects, after we learn more about primitives. The symbol type is used to create unique identifiers for objects. We have to mention it here for the sake of completeness, but also postpone the details till we know objects.

**6.The type of operator=>**

The typeof operator returns the type of the operand. It’s useful when we want to process values of different types differently or just want to do a quick check.

typeof undefined // "undefined"

typeof 0 // "number"

typeof 10n // "bigint"

typeof true // "boolean"

typeof "foo" // "string"

typeof Symbol("id") // "symbol"

typeof Math // "object" (1)

typeof null // "object" (2)

typeof alert // "function" (3)

**Interaction: alert, prompt, confirm**

**1.alert**

shows a message.

Ex:alert(“hello world”);

**2.Prompt:**

shows a message asking the user to input text. It returns the text or, if Cancel button or Esc is clicked, null.

The function prompt accepts two arguments:

result = prompt(title, [default]);

It shows a modal window with a text message, an input field for the visitor, and the buttons OK/Cancel.

**title**

The text to show the visitor.

**default**

An optional second parameter, the initial value for the input field.

**2.Prompt:**

shows a message and waits for the user to press “OK” or “Cancel”. It returns true for OK and false for Cancel/Esc.

result = confirm(question);

The function confirm shows a modal window with a question and two buttons: OK and Cancel.

The result is true if OK is pressed and false otherwise.

# **Type Conversions**

**1. String Conversion:**

String conversion happens when we need the string form of a value.

For example, alert(value) does it to show the value.

We can also call the String(value) function to convert a value to a string:

Example:

let value = true;

alert(typeof value); // boolean

value = String(value); // now value is a string "true"

alert(typeof value); /

**2. Numeric Conversion:**

Numeric conversion in mathematical functions and expressions happens automatically.

For example, when division / is applied to non-numbers:

let str = "123";

alert(typeof str); // string

let num = Number(str); // becomes a number 123

alert(typeof num); // number

Example:

alert( Number(" 123 ") ); // 123

alert( Number("123z") ); // NaN (error reading a number at "z")

alert( Number(true) ); // 1

alert( Number(false) ); // 0

**2. Boolean Conversion:** Boolean conversion is the simplest one.

It happens in logical operations (later we’ll meet condition tests and other similar things) but can also be performed explicitly with a call to Boolean(value).

The conversion rule:

Values that are intuitively “empty”, like 0, an empty string, null, undefined, and NaN, become false.

Other values become true.

Example:

alert( Boolean(1) ); // true

alert( Boolean(0) ); // false

alert( Boolean("hello") ); // true

alert( Boolean("") ); // false

alert( Boolean("0") ); // true

alert( Boolean(" ") ); // spaces, also true (any non-empty string is true)

# **Basic operators, maths:**

**1.Math:**

The following math operations are supported:

Addition +,

Subtraction -,

Multiplication \*,

Division /,

Remainder %,

Exponentiation \*\*.

**Example:**

**Reminder:**

alert( 5 % 2 ); // 1, the remainder of 5 divided by 2

**Exponentiation :**

alert( 2 \*\* 2 ); // 2² = 4

**String concatenation with binary +**

let s = "my" + "string";

alert(s); // mystring

**Numeric conversion, unary +**

let x = 1;

alert( +x ); // 1

let y = -2;

alert( +y ); // -2

// Converts non-numbers

alert( +true ); // 1

alert( +"" ); // 0

**2. Assignment**

an assignment = is also an operator. It is listed in the precedence table with the very low priority of 2.

Example:

let x = 2 \* 2 + 1;

alert( x ); // 5

**Chaining assignments**

Example:

let a, b, c;

a = b = c = 2 + 2;

alert( a ); // 4

alert( b ); // 4

alert( c ); // 4

**Modify-in-place**

**Eample:**

let n = 2;

n += 5; // now n = 7 (same as n = n + 5)

n \*= 2; // now n = 14 (same as n = n \* 2)

alert( n ); // 14

**3.Increment/decrement**

**Increment:**

let counter = 2;

counter++; // works the same as counter = counter + 1, but is shorter

alert( counter ); // 3

**Decrement:**

let counter = 2;

counter--; // works the same as counter = counter - 1, but is shorter

alert( counter ); // 1

**4. Bitwise operators:**

The list of operators:

AND ( & )

OR ( | )

XOR ( ^ )

NOT ( ~ )

LEFT SHIFT ( << )

RIGHT SHIFT ( >> )

ZERO-FILL RIGHT SHIFT ( >>> )

**5. Comma**

The comma operator , is one of the rarest and most unusual operators. Sometimes, it’s used to write shorter code,

Example:

let a = (1 + 2, 3 + 4);

alert( a );

# **Conditional branching: if, '?'**

**The “if” statement**

The if(...) statement evaluates a condition in parentheses and, if the result is true, executes a block of code.

Subtopic AND,OR,NOT

**Nullish coalescing operator '??'**

Example

let user;

alert(user ?? "Anonymous"); // Anonymous (user is undefined)

**The "switch" statement**

switch(x) {

case 'value1': // if (x === 'value1')

...

[break]

case 'value2': // if (x === 'value2')

...

[break]

default:

...

[break]

}

**Loops: while and for:**

**1.while loop:**

while (condition) {

// code

// so-called "loop body"

}

**2.do while loop:**

do {

// loop body

} while (condition);

**3.for loop:**

for (begin; condition; step) {

// ... loop body ...

}