sourcemind

JAVASCRIPT BASICS

VARIABLES
OPERATORS
TYPES

JavaScript often abbreviated **JS**, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries.

All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript was **invented by Brendan Eich in 1995**. It was developed for Netscape 2, and became the ECMA-262 standard in 1997.

Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called the JavaScript engine.

sourcemind

```html
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script>
    alert( 'Hello, world!' );
  </script>

  <p>...After the script.</p>

</body>

</html>
```

```html
<script src="/path/to/script.js"></script>
```

```html
<script src="file.js">
  alert(1); // the content is ignored, because src is set
</script>
```

Code Structure

atements are syntax constructs and commands that perform act

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

Usually, statements are written on separate lines to make the code readable:

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

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

```
alert('Hello')
alert('World')
```

There are cases when a newline does not mean a semicolon. F e:

```
alert(3 +
1
+ 2);
```

Comments

```
// This comment occupies a line of its own
alert('Hello');
```

```
/* An example with two messages.
This is a multiline comment.
*/
```

sourcemind

Variables

```
let message;
```

"Hello!"
message

Now, we can put some data into it by using the assignment operator `=`:

```
let message;
message = 'Hello!';

alert(message); // shows the variable content
```

We can also declare multiple variables in one line:

```
let user = 'John', age = 25, message = 'Hello';
```

The multiline variant is a bit longer, but easier to read:

```
let user = 'John';
let age = 25;
let message = 'Hello';
```

```
let user = 'John',
  age = 25,
  message = 'Hello';
```

sourcemind

Variables

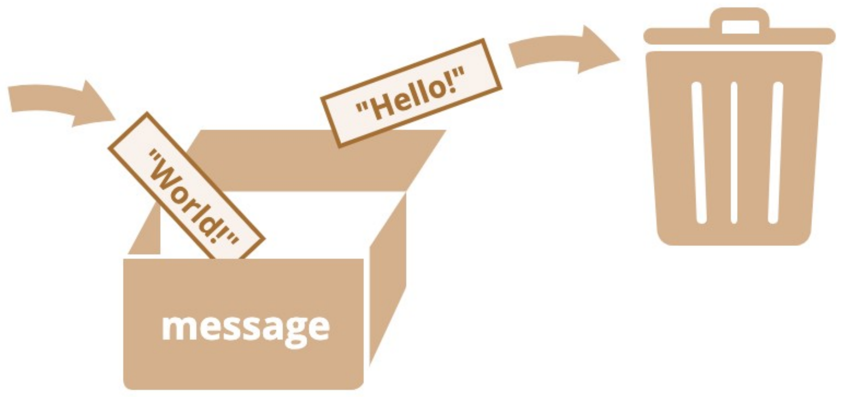
```
let message;

message = 'Hello!';

message = 'World!'; // value changed

alert(message);
```

When the value is changed, the old data is removed from the variable:



Declaring twice triggers an error

A variable should be declared only once.

A repeated declaration of the same variable is an error:

Variable naming

There are two limitations on variable names in JavaScript:

1. The name must contain only letters, digits, or the symbols `$` and `_`.
2. The first character must not be a digit.

Reserved names

For example: `let`, `class`, `return`, and `function` are reserved.

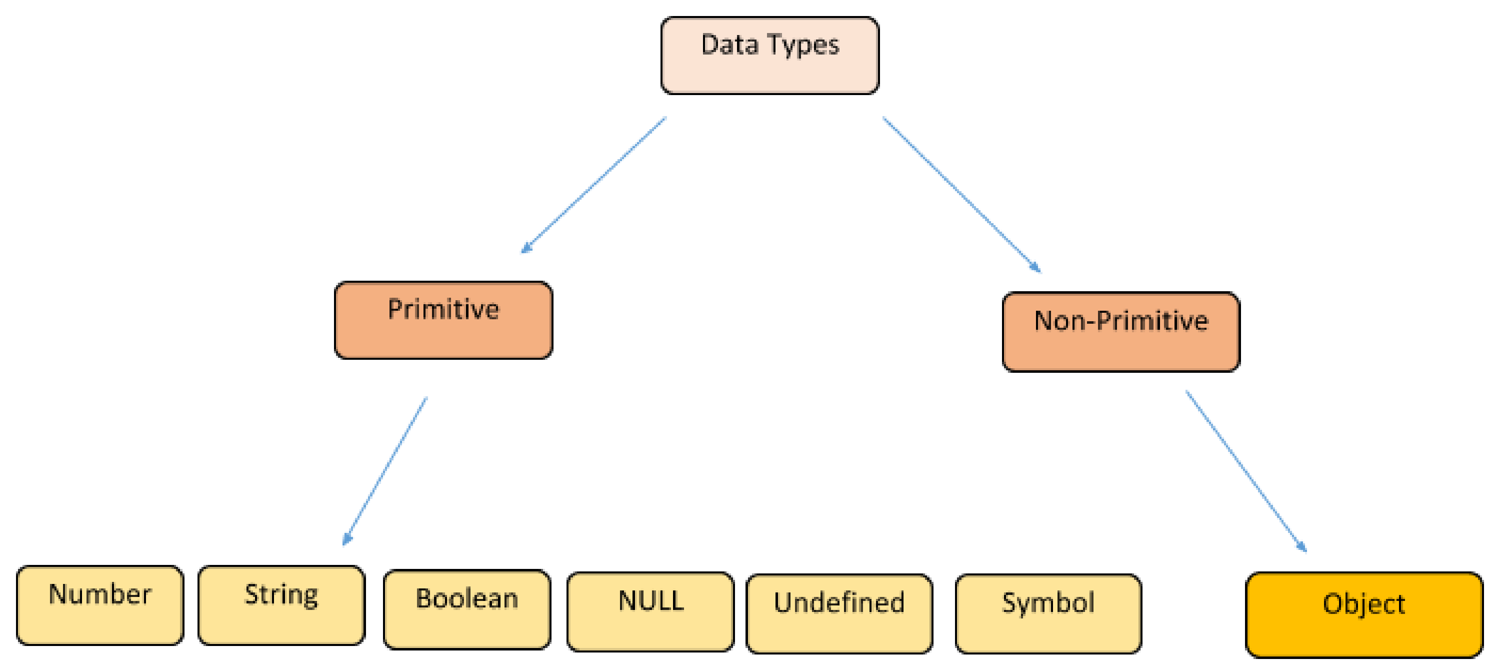
Constants

To declare a constant (unchanging) variable, use `const` instead of `let`:

```
const myBirthday = '18.04.1982';
```

sourcemind

Data Types


Data Types
Primitive
Non-Primitive
Number
String
Boolean
NULL
Undefined
Symbol
Object

Data Types

Number

```
let n = 123;
n = 12.345;
```

The *number* type represents both integer and floating point numbers.

Besides regular numbers, there are so-called “special numeric values” which also belong to this data type: `Infinity`, `-Infinity` and `NaN`.

```
alert( "not a number" / 2 ); // NaN, such division is erroneous
```

String

A string in JavaScript must be surrounded by quotes.

```
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed another ${str}`;
```

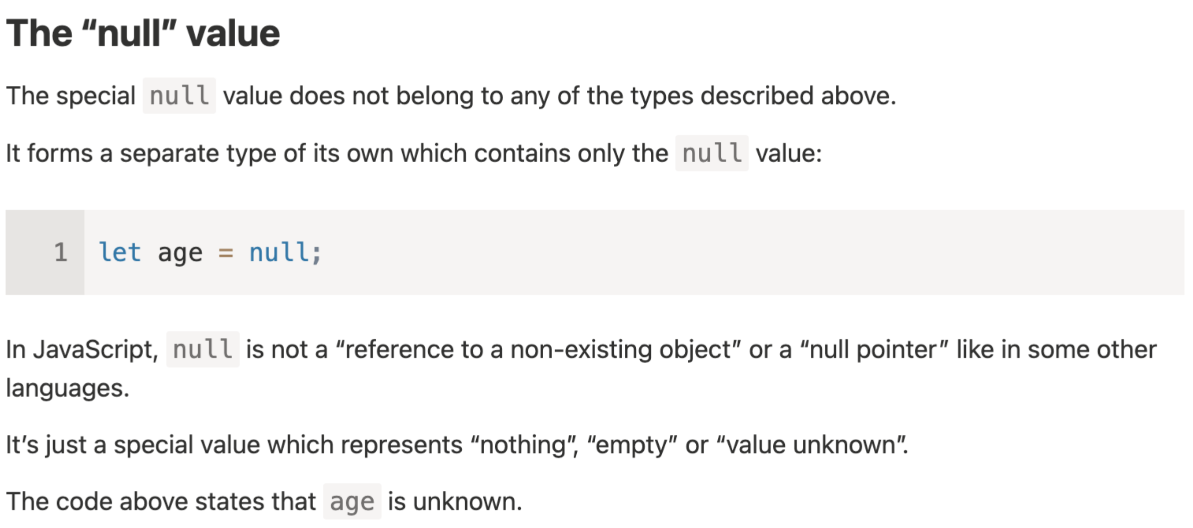
Boolean (logical type)

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

```
let nameFieldChecked = true; // yes, name field is checked
let ageFieldChecked = false; // no, age field is not checked
```

sourcemind

Data Types



The "null" value

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

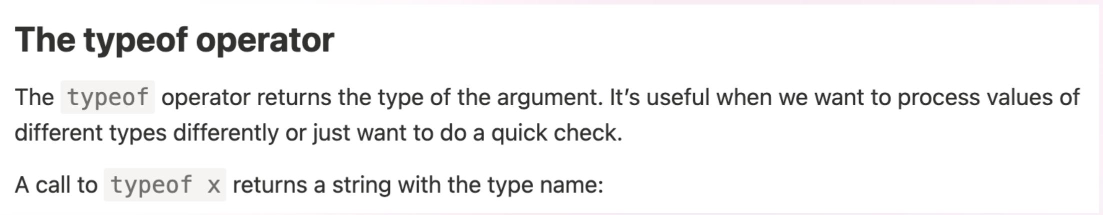
It forms a separate type of its own which contains only the `null` value:

```js
let age = null;
```

In JavaScript, `null` is not a "reference to a non-existing object" or a "null pointer" like in some other languages.

It's just a special value which represents "nothing", "empty" or "value unknown".

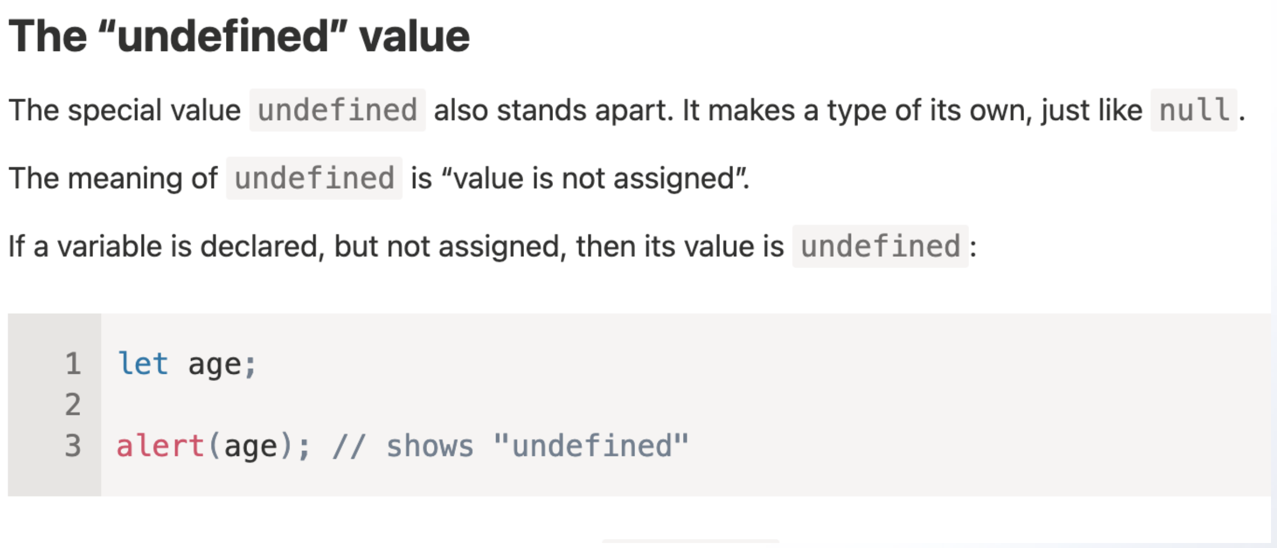
The code above states that `age` is unknown.



The typeof operator

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

A call to `typeof x` returns a string with the type name:



The "undefined" value

The special value `undefined` also stands apart. It makes a type of its own, just like `null`.

The meaning of `undefined` is "value is not assigned".

If a variable is declared, but not assigned, then its value is `undefined`:

```js
let age;

alert(age); // shows "undefined"
```

Type Conversion

Converting Strings to Numbers

The global method `Number()` can convert strings to numbers.

The **unary + operator** can be used to convert a variable to a number:

Example

```
let y = "5";       // y is a string
let x = + y;       // x is a number
```

```
Number("3.14")    // returns 3.14
Number(" ")       // returns 0
Number("")        // returns 0
Number("99 88")   // returns NaN
```

Number Methods

In the chapter Number Methods, you will find more methods that can be used to convert strings to numbers:

Method	Description
Number()	Returns a number, converted from its argument
parseFloat()	Parses a string and returns a floating point number
parseInt()	Parses a string and returns an integer

Converting Numbers to Strings

```
String(123)       // returns a string from a number literal 123
String(100 + 23)  // returns a string from a number from an expression
```

```
(123).toString()
(100 + 23).toString()
```

Tasks

1. Create 2 variables ***name,*** *age with values*
2. Reassign Values
3. Show changed values using *alert()* function

sourcemind

Interaction: alert, prompt, confirm

alert

This one we've seen already. It shows a message and waits for the user to press "OK".

For example:

```
alert("Hello");
```

prompt

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.

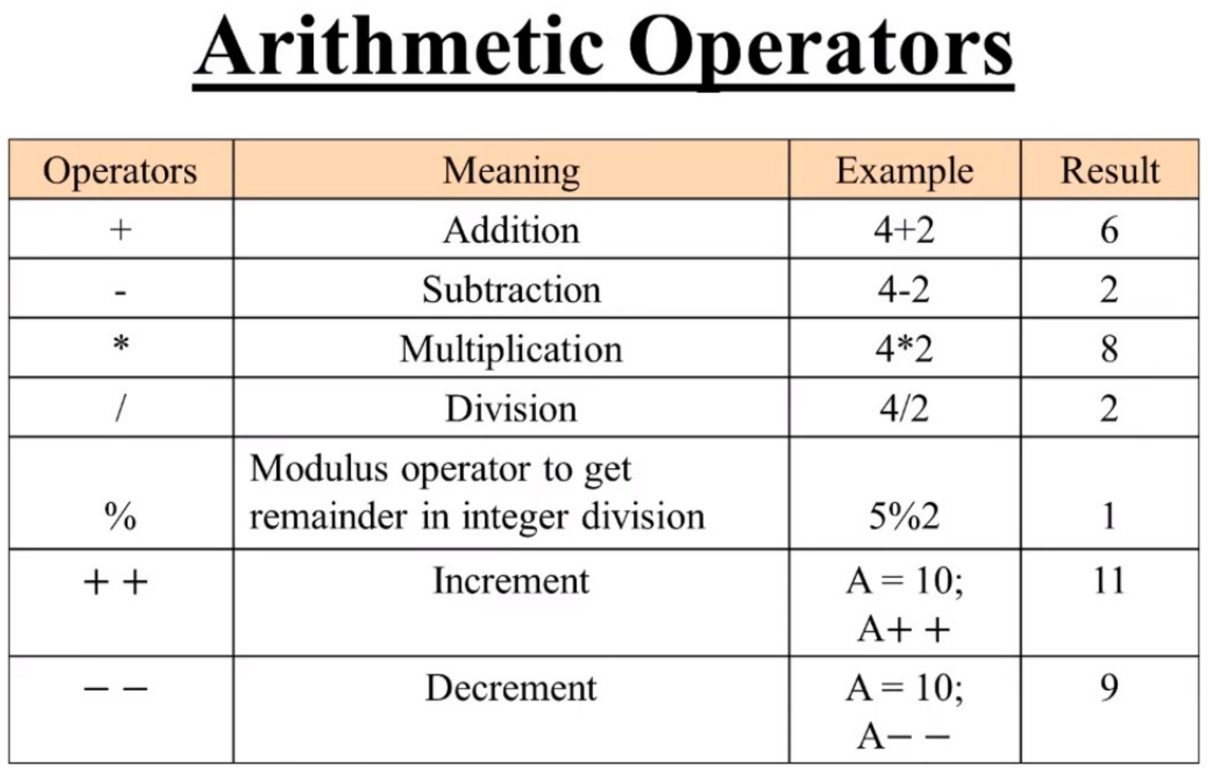
confirm

The syntax:

```
result = confirm(question);
```

sourcemind

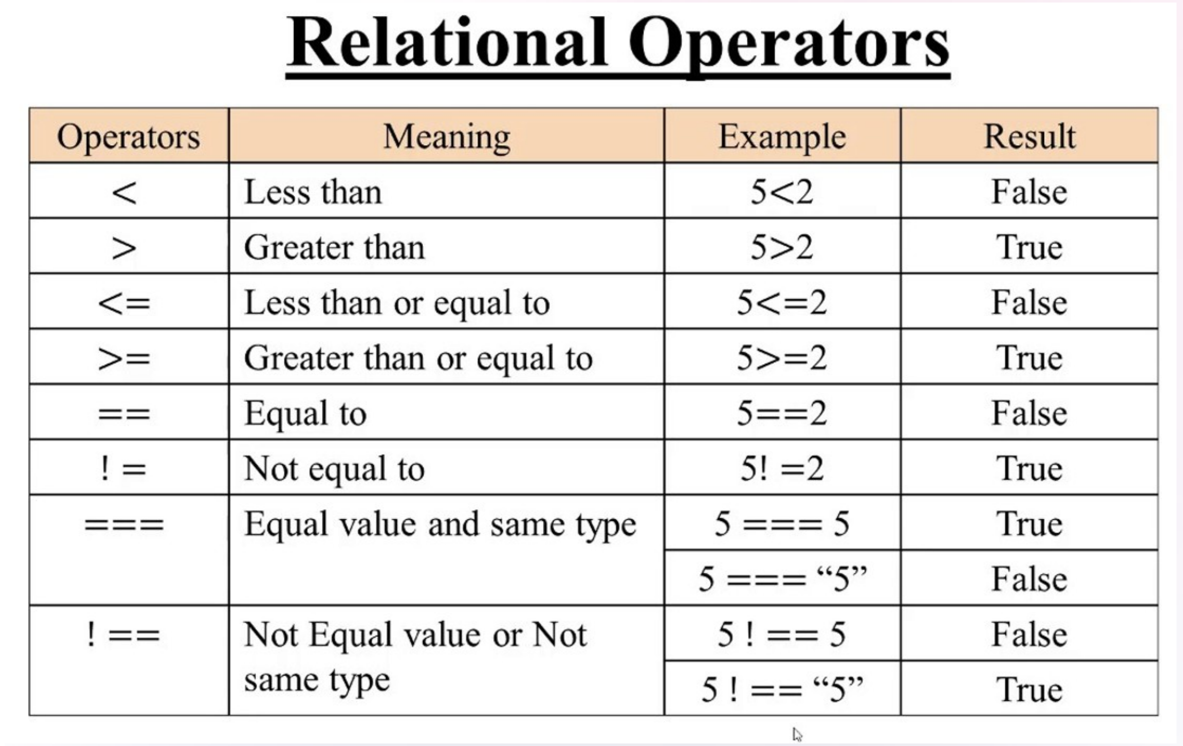
Basic operators



Arithmetic Operators

Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1
+ +	Increment	A = 10; A+ +	11
− −	Decrement	A = 10; A− −	9

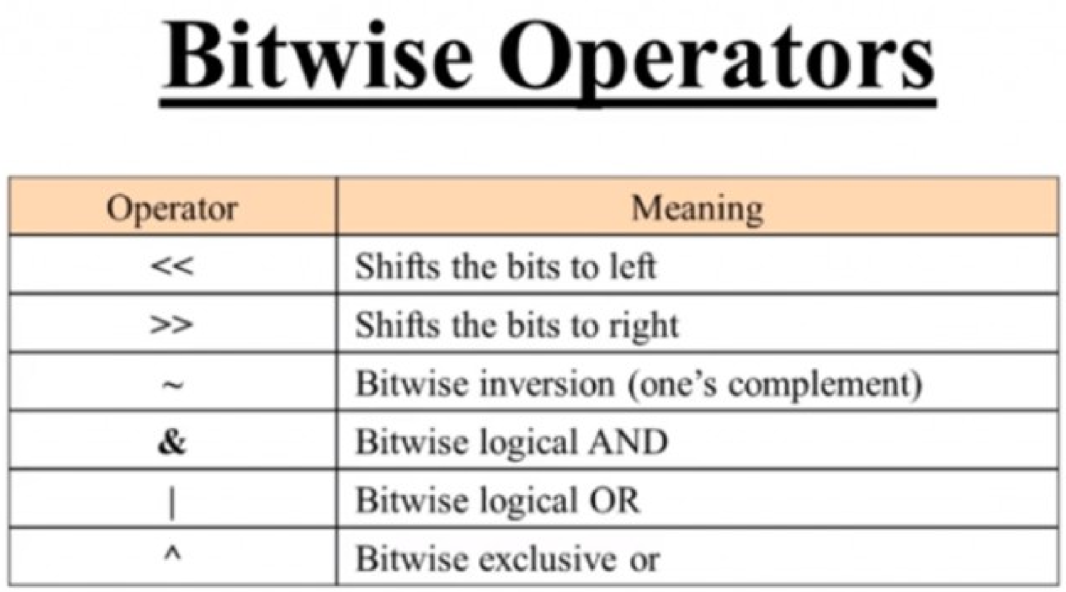
**	Exponentiation	2**3	8



Relational Operators

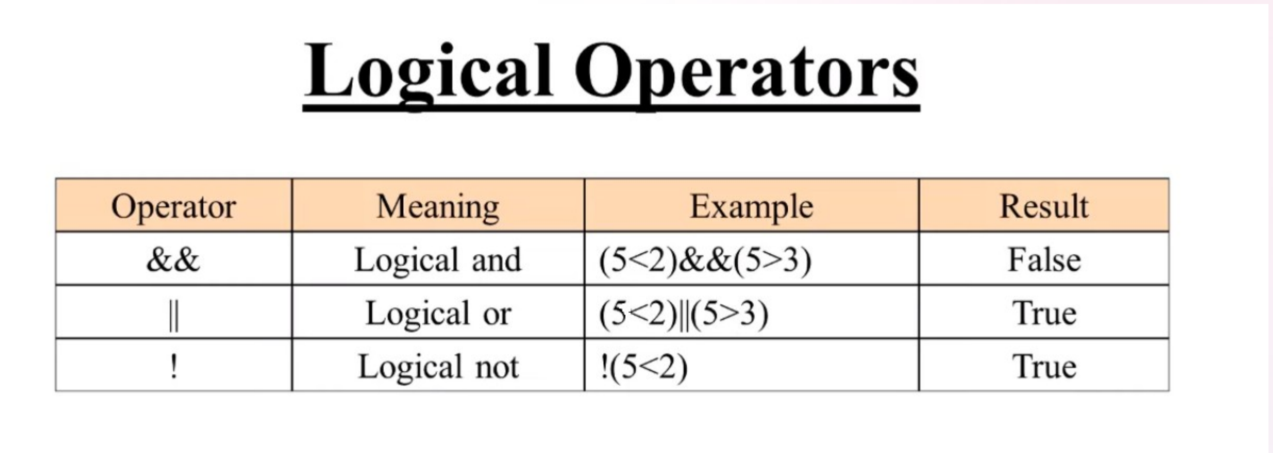
Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
! =	Not equal to	5! =2	True
===	Equal value and same type	5 === 5	True
		5 === "5"	False
! ==	Not Equal value or Not same type	5 ! == 5	False
		5 ! == "5"	True

sourcemind



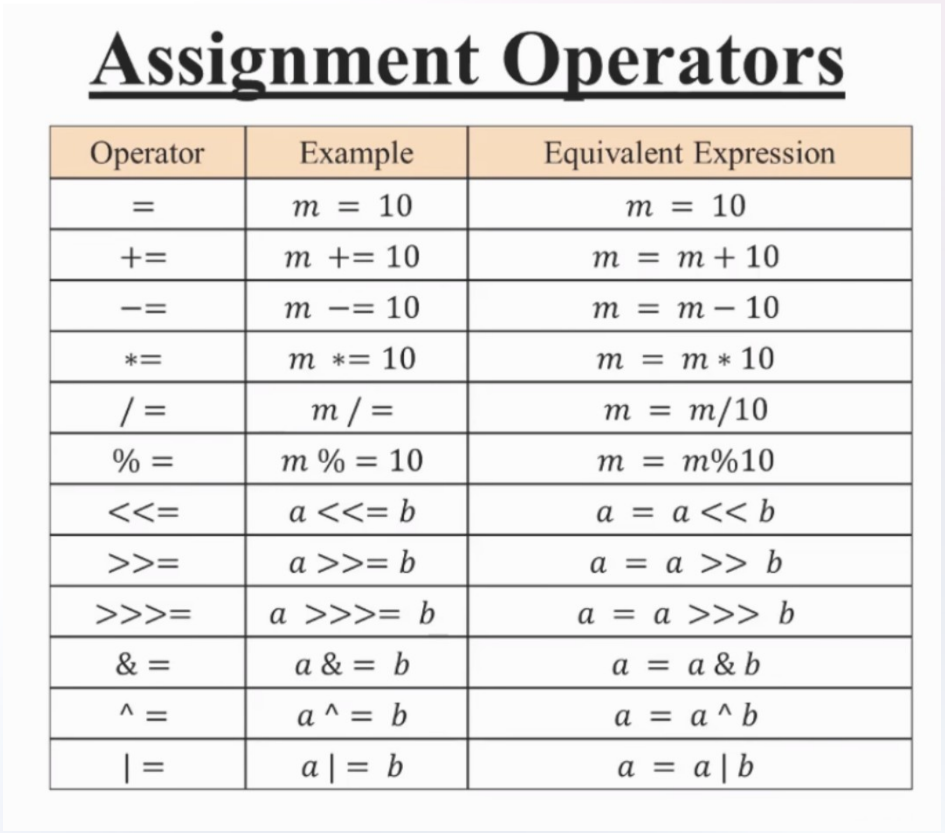
Bitwise Operators

Operator	Meaning
<<	Shifts the bits to left
>>	Shifts the bits to right
~	Bitwise inversion (one's complement)
&	Bitwise logical AND
|	Bitwise logical OR
^	Bitwise exclusive or



Logical Operators

Operator	Meaning	Example	Result
&&	Logical and	(5<2)&&(5>3)	False
||	Logical or	(5<2)||(5>3)	True
!	Logical not	!(5<2)	True



Assignment Operators

Operator	Example	Equivalent Expression
=	m = 10	m = 10
+=	m += 10	m = m + 10
−=	m −= 10	m = m − 10
*=	m *= 10	m = m * 10
/ =	m / =	m = m/10
% =	m % = 10	m = m%10
<<=	a <<= b	a = a << b
>>=	a >>= b	a = a >> b
>>>=	a >>>= b	a = a >>> b
& =	a & = b	a = a & b
^ =	a ^ = b	a = a ^ b
| =	a | = b	a = a | b

sourcemind

Tasks

1. Create prompt for getting user name and log it.
2. Create prompt for user age and if age is bigger than 18 log *true* else log *false*
3. Create x,y variables and use all operators (+,-, *, **.....)
4. Create x,y variables and swap them without using 3th variable.
5. Calculate and log areas of square, triangle, rectangle ,circle

sourcemind