**Logical operator**

Logical operators are also known as “logical connectives” as they are used to **connect** one or more **conditions not values**. They accept Boolean operands, on evaluation they yield the result **true** or **false**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Logical operator** | **sign** | **function** |  |
| **And Operator** | **&&** | If both left and right operands are **true** then the result will be **true**, in all other cases the result will be **false.** | Accept 2 conditions (Binary operator) |
| **Or Operator** | **||** | If both left and right operands are **false** then the result will be **false**, in all other cases the result will be **true.** | Accept 2 conditions (Binary operator) |
| **Not / Negation Operator** | **!** | If the given operand is **false**, then the result will be **true**. **Vice versa** | Accept 1 condition (unary operator) |
| **Nullish coalescing operator** | **??** | If the first operand is “**null**” or “**undefined**” then the second one will be executed, otherwise the first one will be executed. | Accept 2 conditions (Binary operator) |

**Important steps:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **&&** | | | **||** | | | **!** | |
| **1st condition** | **2nd condition** | **Result** | **1st condition** | **2nd condition** | **Result** | **1st condition** | **2nd condition** |
| **true** | **true** | **true** | **true** | **true** | **true** | **true** | **false** |
| **true** | **false** | **false** | **true** | **false** | **true** | **false** | **true** |
| **false** | **true** | **false** | **false** | **true** | **true** |  | |
| **false** | **false** | **false** | **false** | **false** | **false** |  | |

|  |  |  |
| --- | --- | --- |
| **??** | | |
| **1st condition** | **2nd condition** | **Result** |
| **null / undefined** | **Any value (!null / !undefined)** | **Condition 2** |
| **!null / !undefined** | **null / undefined** | **Condition 1** |
| **null / undefined** | **null / undefined** | **Nothing** |
| **!null / !undefined** | **!null / !undefined** | **Condition1** |

**Summary:** ?? Operator executes the first “**defined”** value, while || operator execute the first “**truth**” value.

**Example:**

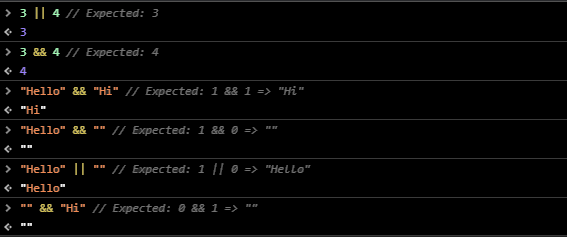
****

**Note:** if you use non Booleans values ***treat* *as*** 0 and 1 **Remember**: 0, NaN, undefined, null, false, “”empty string give **0** otherwise give **1**.

**Trick:**

* **&&**, **Second** value will return.
* **||**, **First** value will return.

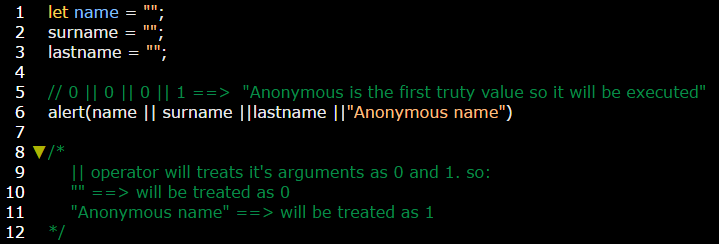
**Example:**

****

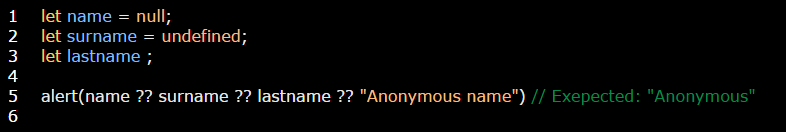
**Short-circuit evaluation and || operator:**

Short-circuit means that **||** operator will processes it’s operands until the first truthy value is reached.

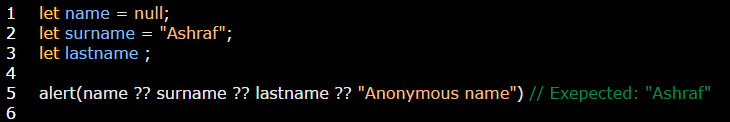
**Example:** on short-circuit



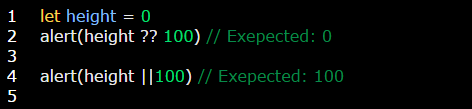
**Example:** on?? Operator

****

**Example 2:**

****

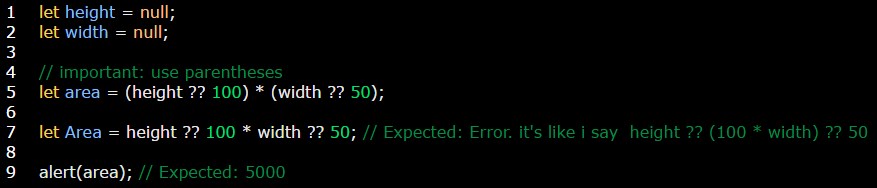
**Example 3:**

****

**Note:** The precedence of the??Operator is low; Where it evaluated before **=** and**?** Operators, but after most other operations, such as **+, \***.

So if we’d like to choose a value with**??** An expression with other operators, consider adding parentheses.

**Example 4:**

****

**Note**: JS forbids using**??** Together with **&&** and **||** operators, unless using**??** with parentheses.

**Example:**



**Example:** on using?? With || and &&.

