Operators
Session-3

CLARUSWAY
WAY TO REINVENT YOURSELF

Table of Contents

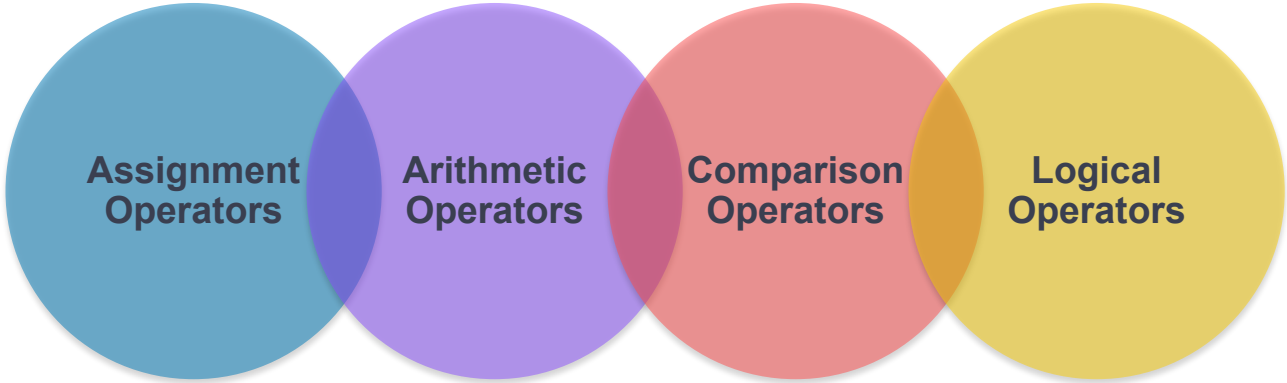
Did you finish Javascript Core pre-class material?

Play Kahoot!

1 Operators

Operators

- Let's take a simple 3 + 2 phrase equals 5. Number 3 and 2 are operands and '+' is the operator.
- Expressions rely on operators to create a single value from one or more values
- JavaScript supports the operators of the following types



2

Assignment Operators

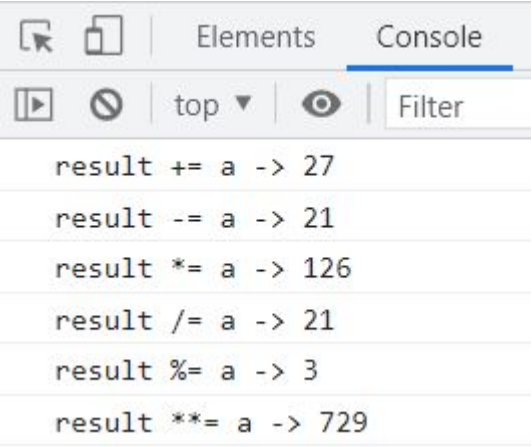
Assignment Operators

Assignment operators assign values to JavaScript variables

OPERATOR	EXAMPLE	MEANING
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

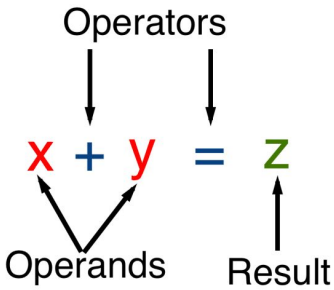
Assignment Operators

```
<script>
  let a = 6, result = 21;
  console.log(`result += a -> ${result += a}`);
  console.log(`result -= a -> ${result -= a}`);
  console.log(`result *= a -> ${result *= a}`);
  console.log(`result /= a -> ${result /= a}`);
  console.log(`result %= a -> ${result %= a}`);
  console.log(`result **= a -> ${result **= a}`);
</script>
```



3

Arithmetic Operators



Arithmetic Operators

Arithmetic operators execute arithmetic functions on numbers (literals or variables)

NAME	OPERATOR	EXAMPLE	RESULT
ADDITION	+	10 + 5	15
SUBTRACTION	-	10 - 5	5
DIVISION	/	10 / 5	2
MULTIPLICATION	*	10 * 5	50
EXPONENTIATION	**	10 ** 3	1000
MODULUS	%	10 % 4	2

Postfix Prefix Operators

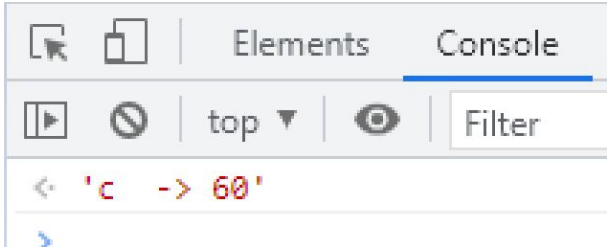
Postfix prefix operators add or subtract one from their operand

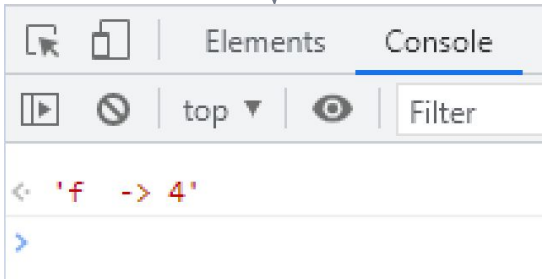
POSTFIX			
NAME	OPERATOR	EXAMPLE	RESULT
INCREMENT	++	`let a = 10;` `let b = a++;`	`a = 11` `b = 10`
DECREMENT	--	`let a = 10;` `let b = a--;`	`a = 9` `b = 10`
PREFIX			
NAME	OPERATOR	EXAMPLE	RESULT
INCREMENT	++	`let a = 10;` `let b = ++a;`	`a = 11` `b = 11`
DECREMENT	--	`let a = 10;` `let b = --a;`	`a = 9` `b = 9`

Arithmetic Operators

```
<script>
  let a = 20, b = 3, c = a * b;
  console.log(`c  -> ${c}`);
</script>
```

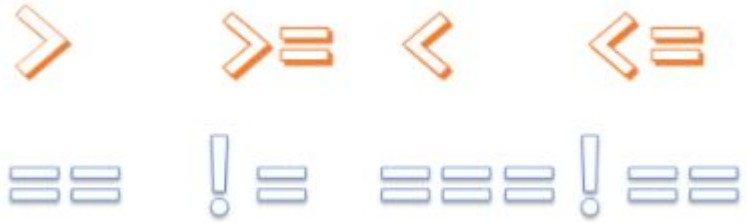
```
<script>
  let d = 25, e = 7, f = d % e;
  console.log(`f  -> ${f}`);
</script>
```





4

Comparison Operators



Comparison Operators

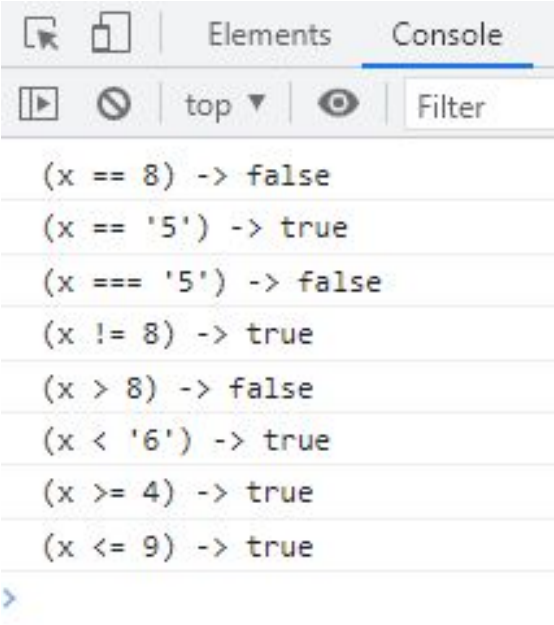
- Comparison operators are used to determine equality or difference between variables or values in logical statements
- All comparison operators return Boolean (true or false)

OPERATOR	DESCRIPTION	EXAMPLE
`==`	Equality	`3 == '3' // true`
`!=`	Inequality	`3 != '3' // false`
`===`	Strict equality (equal and of same type)	`3 === '3' // false`
`!==`	Strict inequality	`3 !== '3' // true`
`>`	Greater than	`3 > 2 // true`
`>=`	Greater than or equal	`3 >= 2 // true`
`<`	Less than	`3 < 2 // false`
`<=`	Less than or equal	`3 <= 2 // false`

The most notable difference between this operator and the equality (`==`) operator is that if the operands are of different types, the `==` operator attempts to convert them to the same type before comparing.

Comparison Operators

```
<script>
  let x = 5;
  console.log(`(x == 8) -> ${x == 8}`);
  console.log(`(x == '5') -> ${x == '5'}`);
  console.log(`(x === '5') -> ${x === '5'}`);
  console.log(`(x != 8) -> ${x != 8}`);
  console.log(`(x > 8) -> ${x > 8}`);
  console.log(`(x < '6') -> ${x < '6'}`);
  console.log(`(x >= 4) -> ${x >= 4}`);
  console.log(`(x <= 9) -> ${x <= 9}`);
</script>
```



5

Logical Operators

NOT (!)

AND (&&)

OR (||)

Logical Operators

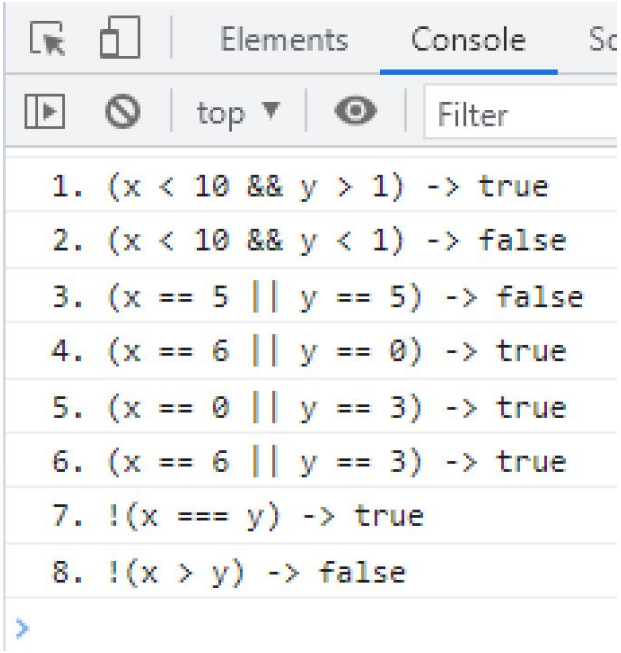
Logical operators, also known as Boolean Operators, are used to determine the logic between variables or values and return true or false.

Seeing as x = 3 and y = 2, logical operators are explained in the table below:

NAME	OPERATOR	DESCRIPTION	EXAMPLE
And	&&	Returns true, if both operands are true	(x < 5 && y > 3) // false
Or	||	Returns true, if either of operands are true	(x == 3 || y == 3) // true
Not	!	Simply toggles the operand value true/false	!(x == y) // true

Logical Operators

```
<script>
  let x = 6, y =3;
  console.log(`1. (x < 10 && y > 1) -> ${x < 10 && y > 1}`);
  console.log(`2. (x < 10 && y < 1) -> ${x < 10 && y < 1}`);
  console.log(`3. (x == 5 || y == 5) -> ${x == 5 || y == 5}`);
  console.log(`4. (x == 6 || y == 0) -> ${x == 6 || y == 0}`);
  console.log(`5. (x == 0 || y == 3) -> ${x == 0 || y == 3}`);
  console.log(`6. (x == 6 || y == 3) -> ${x == 6 || y == 3}`);
  console.log(`7. !(x === y) -> ${!(x === y)}`);
  console.log(`8. !(x > y) -> ${!(x > y)}`);
</script>
```



6

Nullish Coalescing Operator

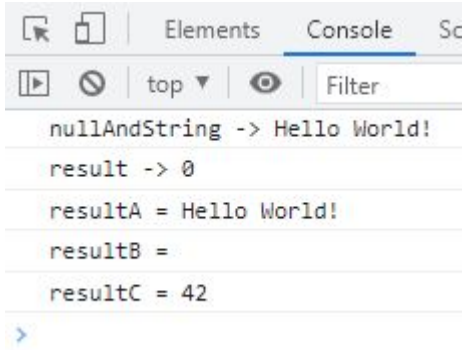
??

Nullish Coalescing Operator

The nullish coalescing operator (??) is a logical operator that returns its right-hand side operand when its left-hand side operand is null or undefined, and otherwise returns its left-hand side operand.

Contrary to the logical OR (||) operator, the left operand is returned if it is a falsy value that is not null or **undefined**.

```
<script>
  const nullAndString = null ?? "Hello World!";
  console.log(`nullAndString -> ${nullAndString}`);

  const result = 0 ?? 42;
  console.log(`result -> ${result}`);

  const nullValue = null;
  const emptyText = ""; //falsy
  const someNumber = 42;

  const resultA = nullValue ?? "Hello World!";
  const resultB = emptyText ?? "Hello World!"
  const resultC = someNumber ?? 0;

  console.log(`resultA = ${resultA}`);
  console.log(`resultB = ${resultB}`);
  console.log(`resultC = ${resultC}`);

</script>
```



7

Other Operators

Other Operators

- typeof operator is used to determine the type of given variable's value.
- instanceof operator used to determine object type of given object, such as arrays, maps etc.

```
<script>

  const arr = [1, 2, 3];
  console.log(typeof arr); //object
  console.log(arr instanceof Array); //true

</script>
```

THANKS!

Any questions?