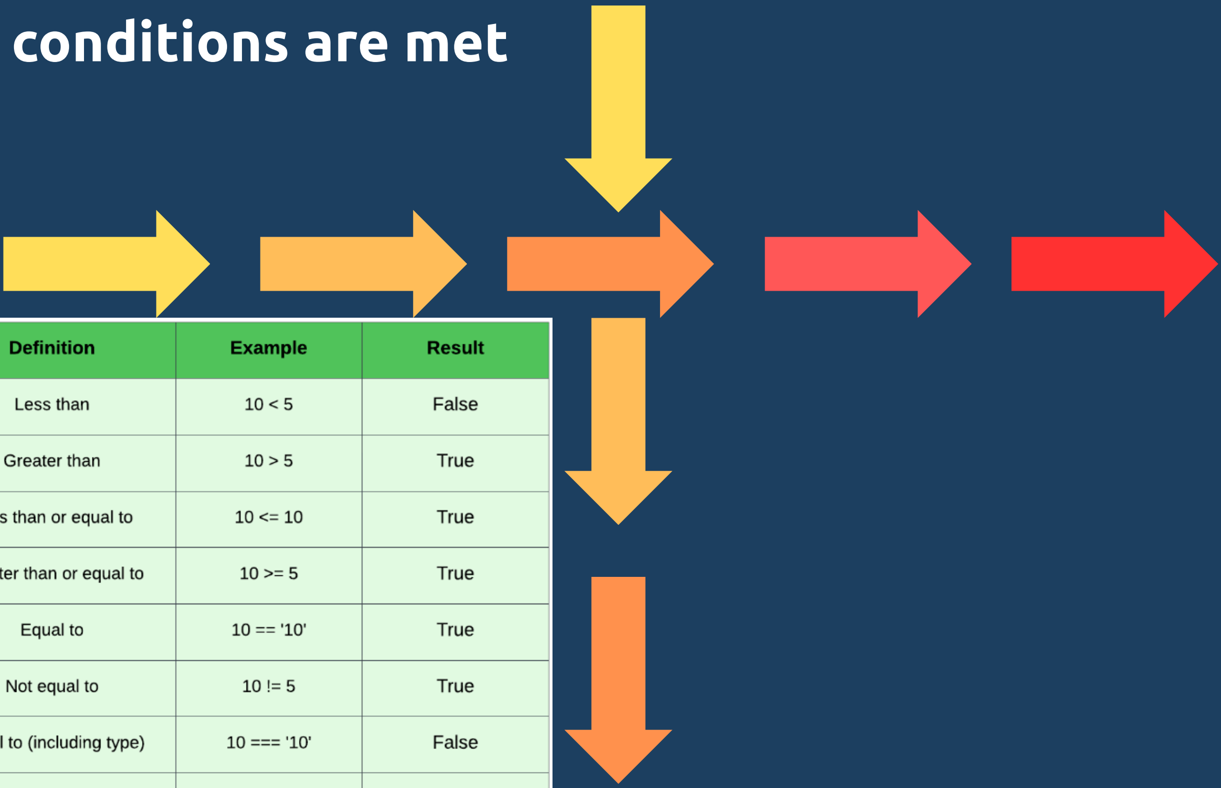
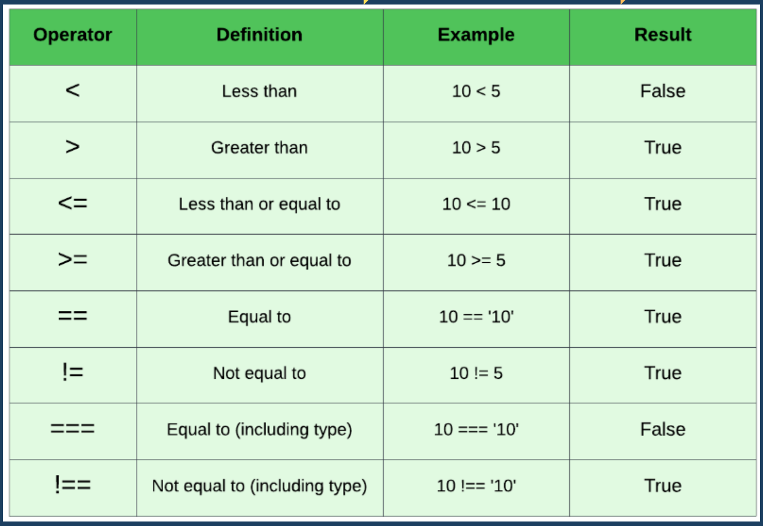
Breaking Javascript

Logic Control Flow

Logic Control Flow

The default control flow is for statements to be read and executed in order from left-to-right, top-to-bottom in a program file. Control structures such as conditionals (if statements and the like) alter control flow by only executing blocks of code if certain conditions are met

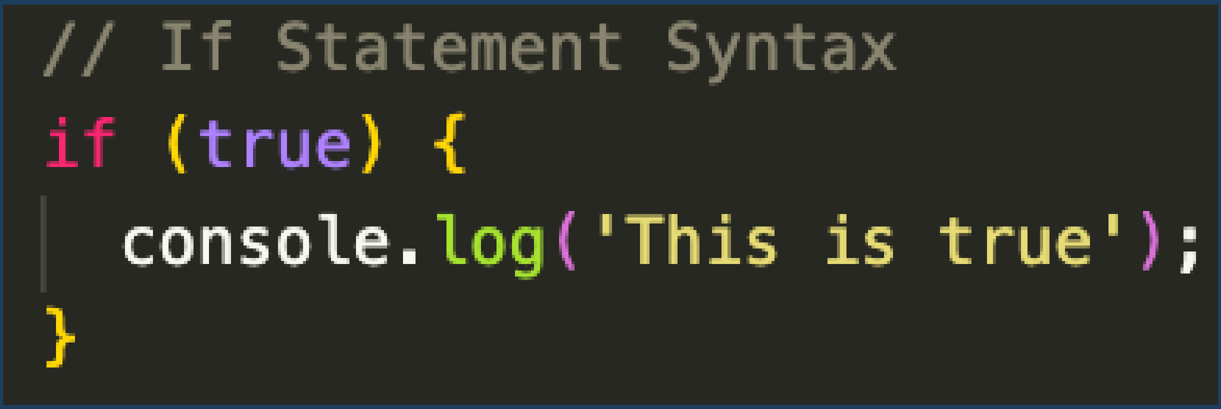




Operator	Definition	Example	Result
<	Less than	10 < 5	False
>	Greater than	10 > 5	True
<=	Less than or equal to	10 <= 10	True
>=	Greater than or equal to	10 >= 5	True
==	Equal to	10 == '10'	True
!=	Not equal to	10 != 5	True
===	Equal to (including type)	10 === '10'	False
!==	Not equal to (including type)	10 !== '10'	True

if statement

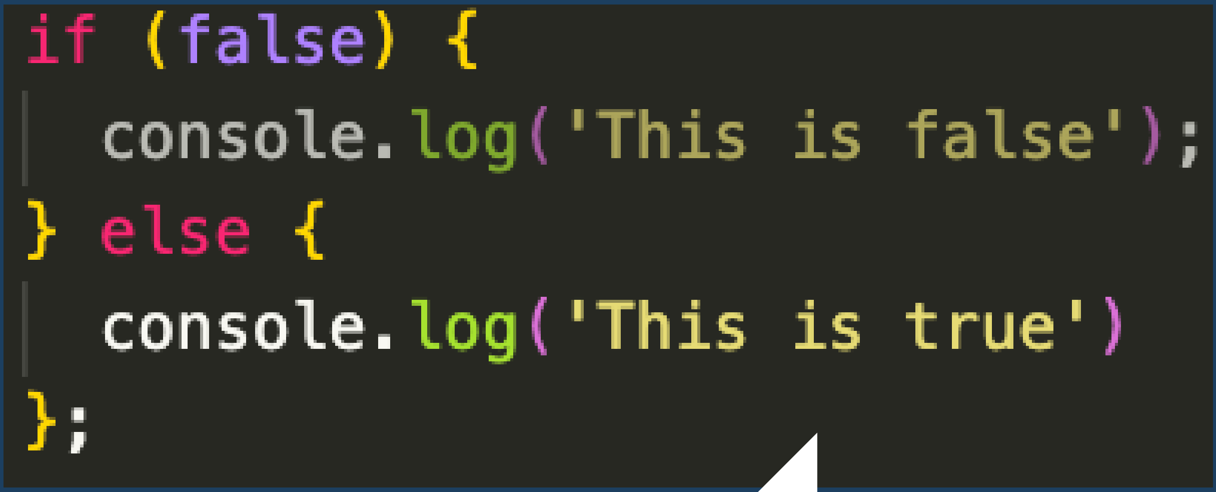
The if statement specifies a block of code to be executed if a condition is true:



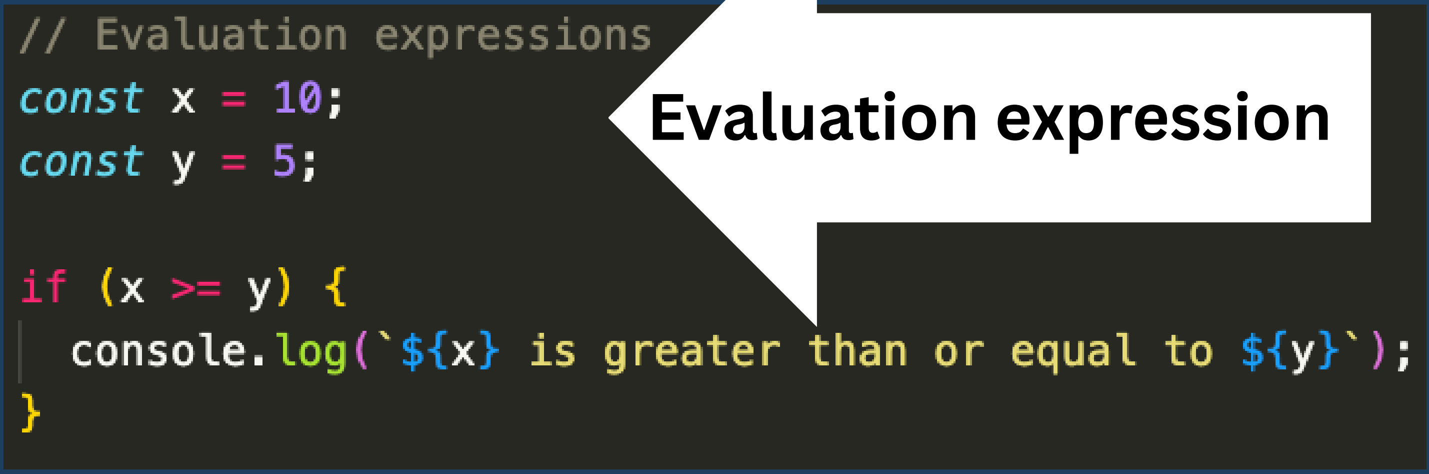
```
// If Statement Syntax
if (true) {
  console.log('This is true');
}
```

else statement.

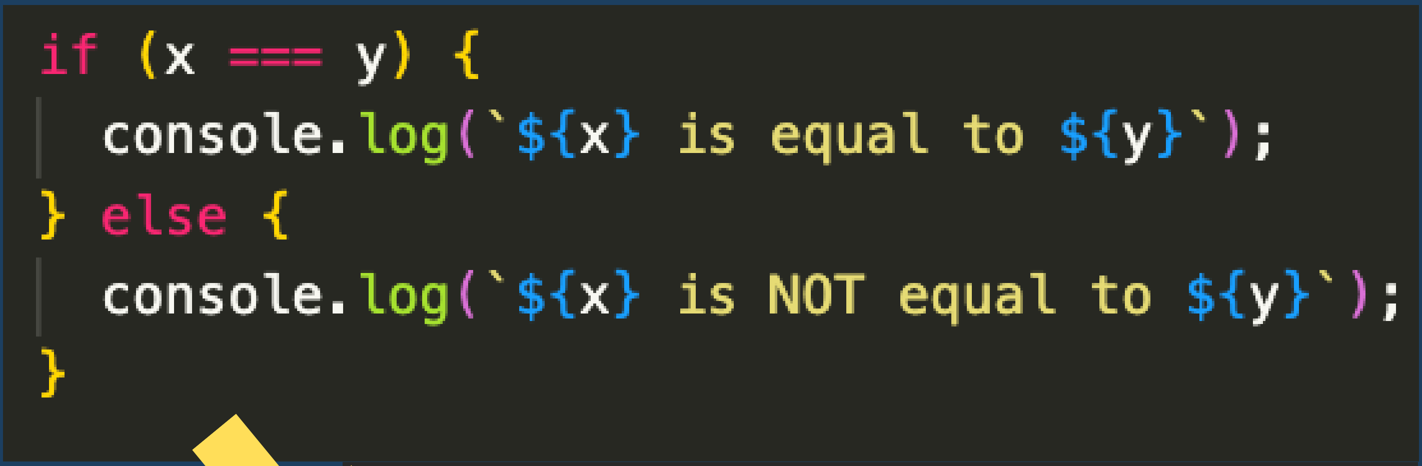
The else statement specifies a block of code to be executed if the condition is false:



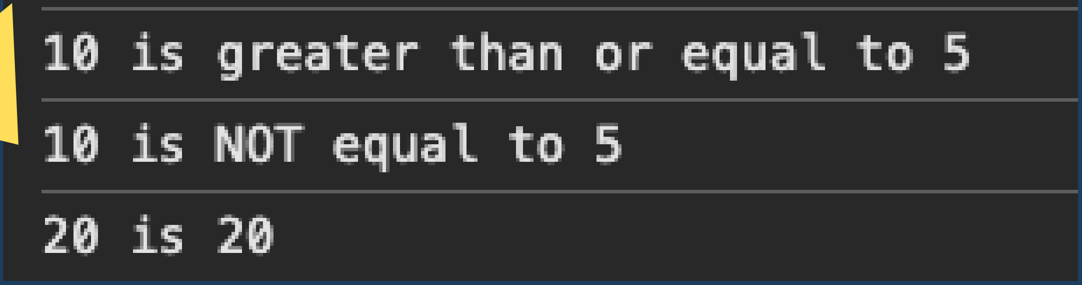
```javascript
if (false) {
  console.log('This is false');
} else {
  console.log('This is true')
};
```



```javascript
// Evaluation expressions
const x = 10;
const y = 5;

if (x >= y) {
  console.log(`${x} is greater than or equal to ${y}`);
}
```



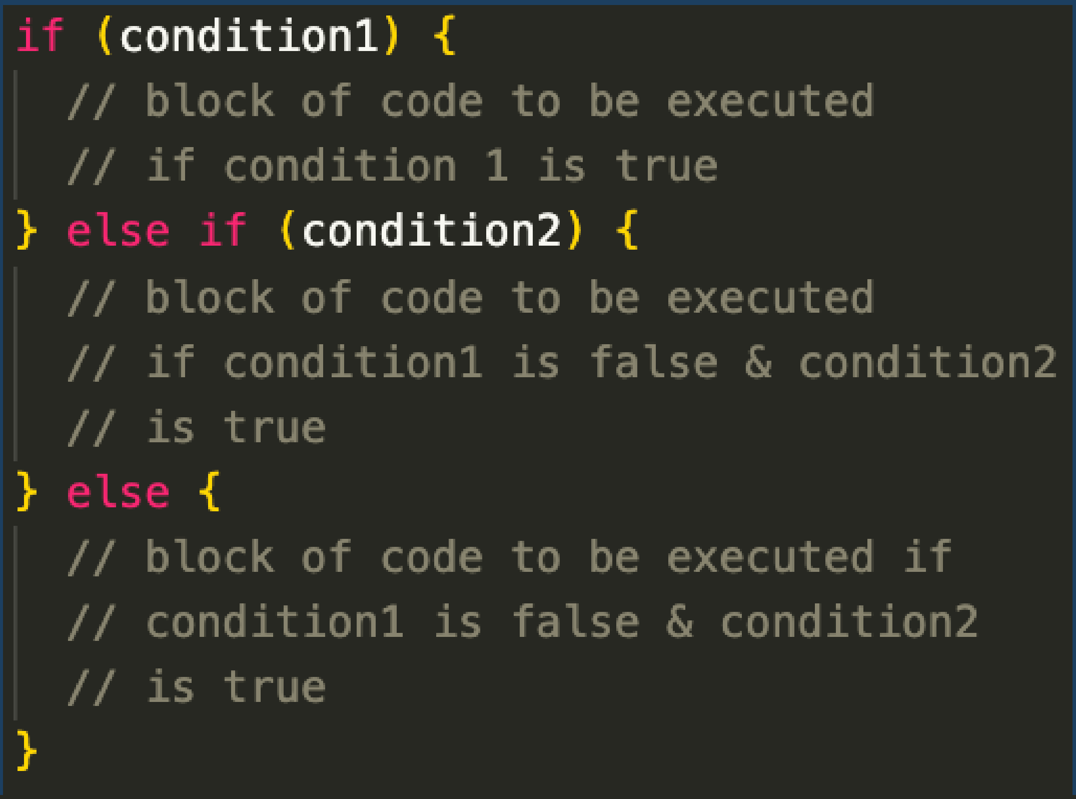
```javascript
if (x === y) {
  console.log(`${x} is equal to ${y}`);
} else {
  console.log(`${x} is NOT equal to ${y}`);
}
```



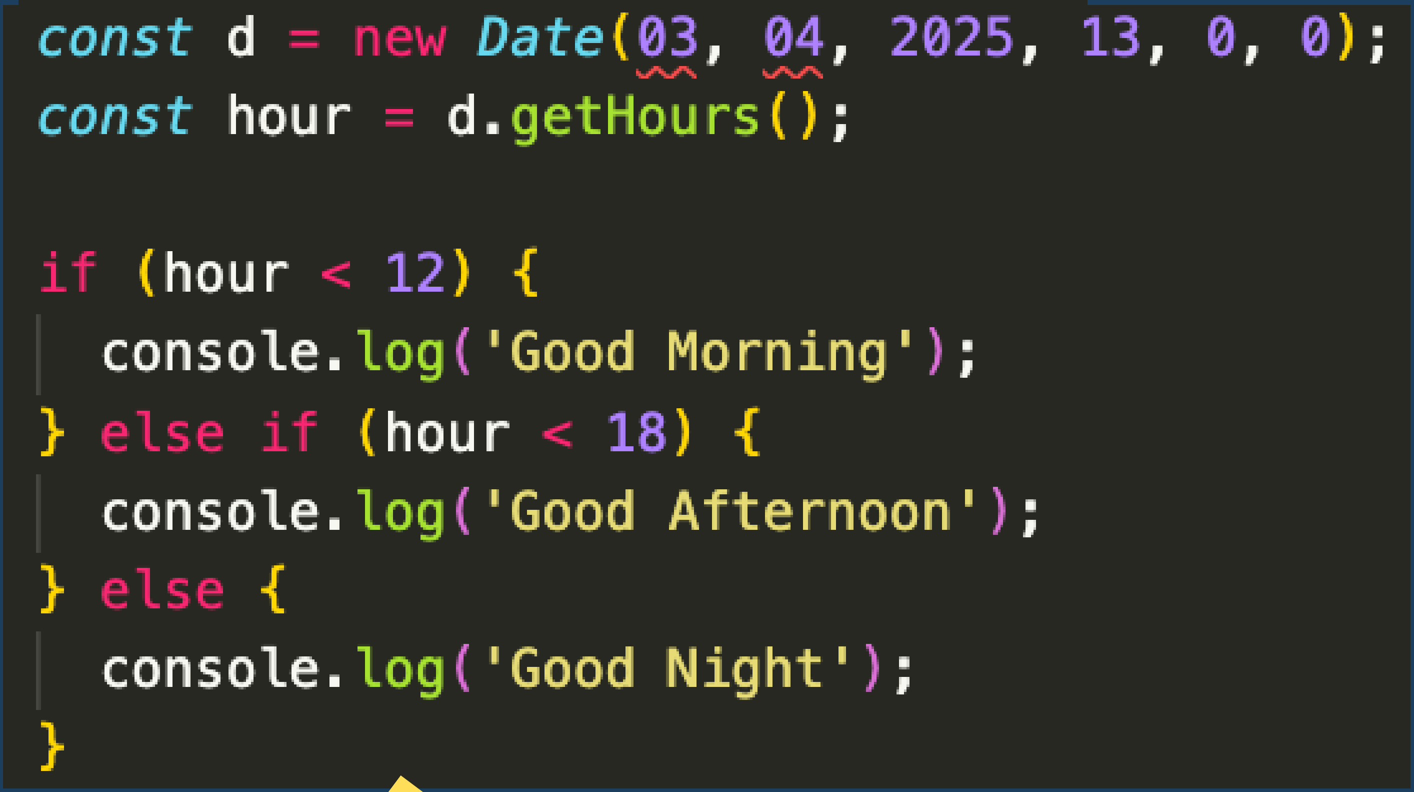
```
10 is greater than or equal to 5
10 is NOT equal to 5
20 is 20
```

else if statement.

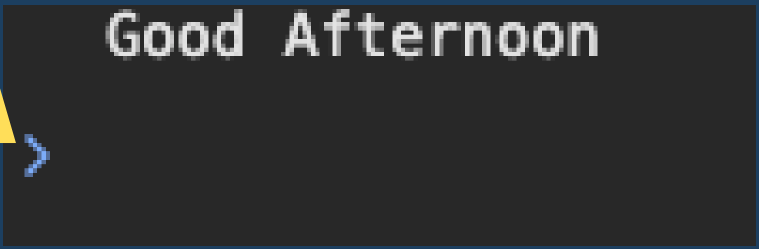
The else if statement specifies a new condition if the first condition is false:



```
if (condition1) {
  // block of code to be executed
  // if condition 1 is true
} else if (condition2) {
  // block of code to be executed
  // if condition1 is false & condition2
  // is true
} else {
  // block of code to be executed if
  // condition1 is false & condition2
  // is true
}
```



```
const d = new Date(03, 04, 2025, 13, 0, 0);
const hour = d.getHours();

if (hour < 12) {
  console.log('Good Morning');
} else if (hour < 18) {
  console.log('Good Afternoon');
} else {
  console.log('Good Night');
}
```



```
Good Afternoon
>
```

nested if

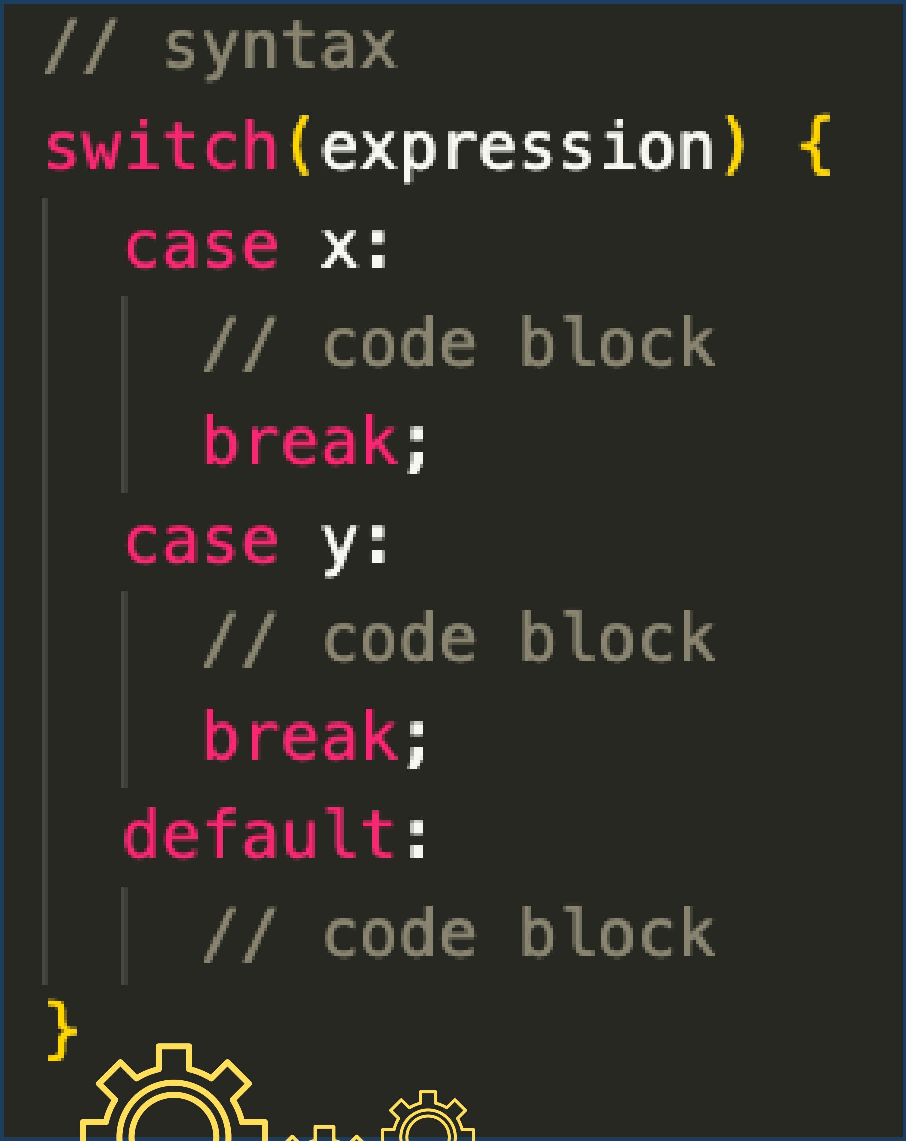
```javascript
const d = new Date(03, 04, 2025, 13, 0, 0);
const hour = d.getHours();
```

```javascript
if (hour < 12) {
  console.log('Good Morning');

  if (hour === 6) {
    console.log('Wake Up!');
  }
} else if (hour < 18) {
  console.log('Good Afternoon');
} else {
  console.log('Good Night');

  if (hour >= 20) {
    console.log('zzzzzzzz');
  }
}


if (hour >= 7 && hour < 15) {
  console.log('It is work time!');
}


if (hour === 6 || hour === 20) {
  console.log('Brush your teeth!')
}
```

```
Good Afternoon
It is work time!
```

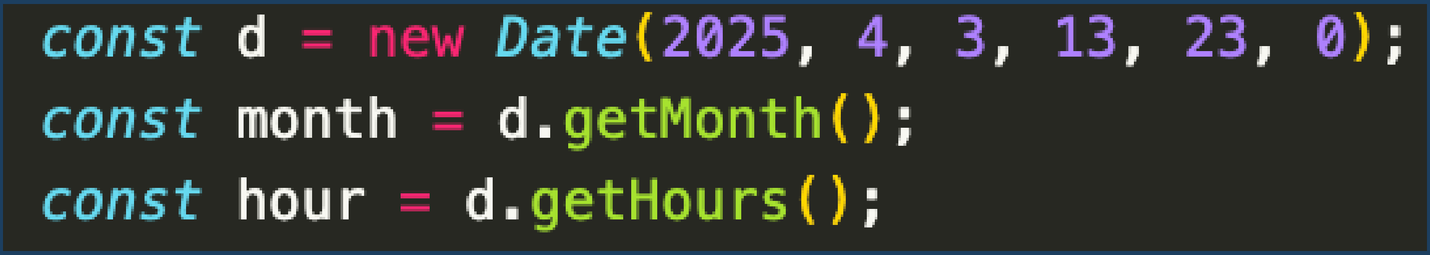
switch statement

The switch statement is used to perform different actions based on different conditions.

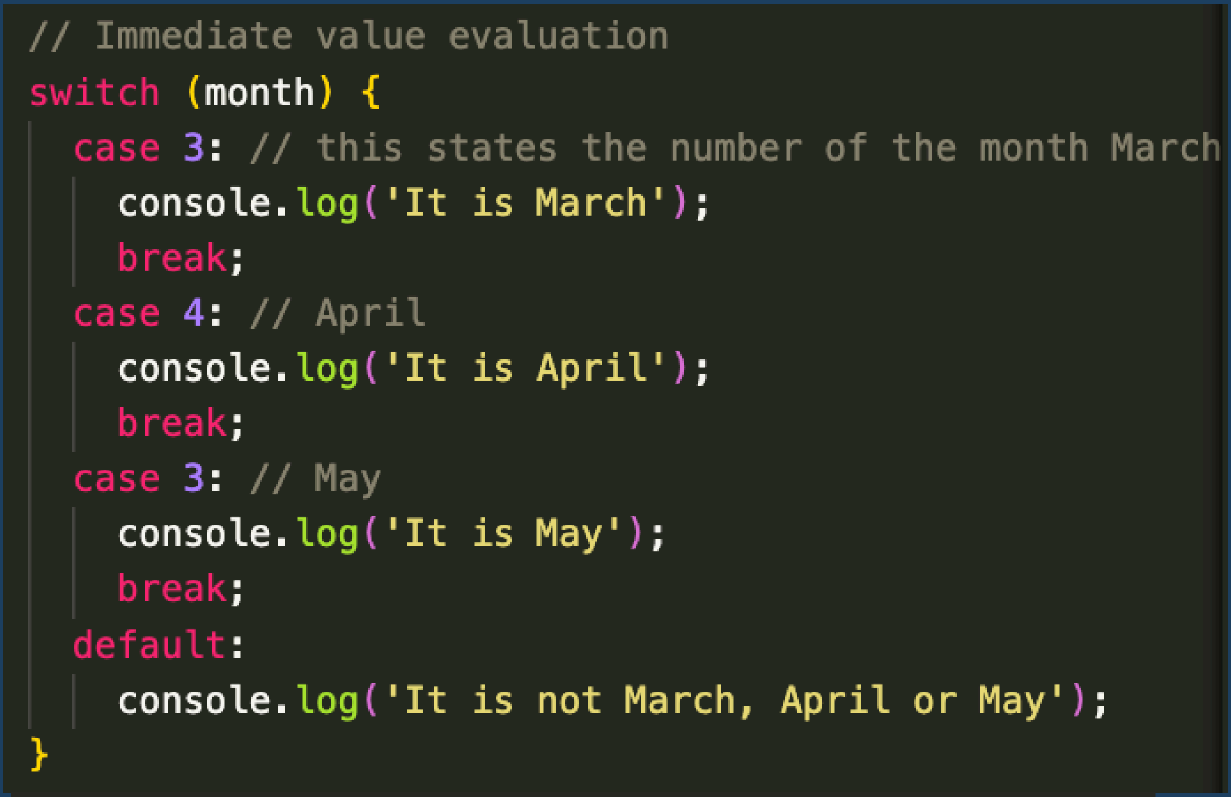


```
// syntax
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

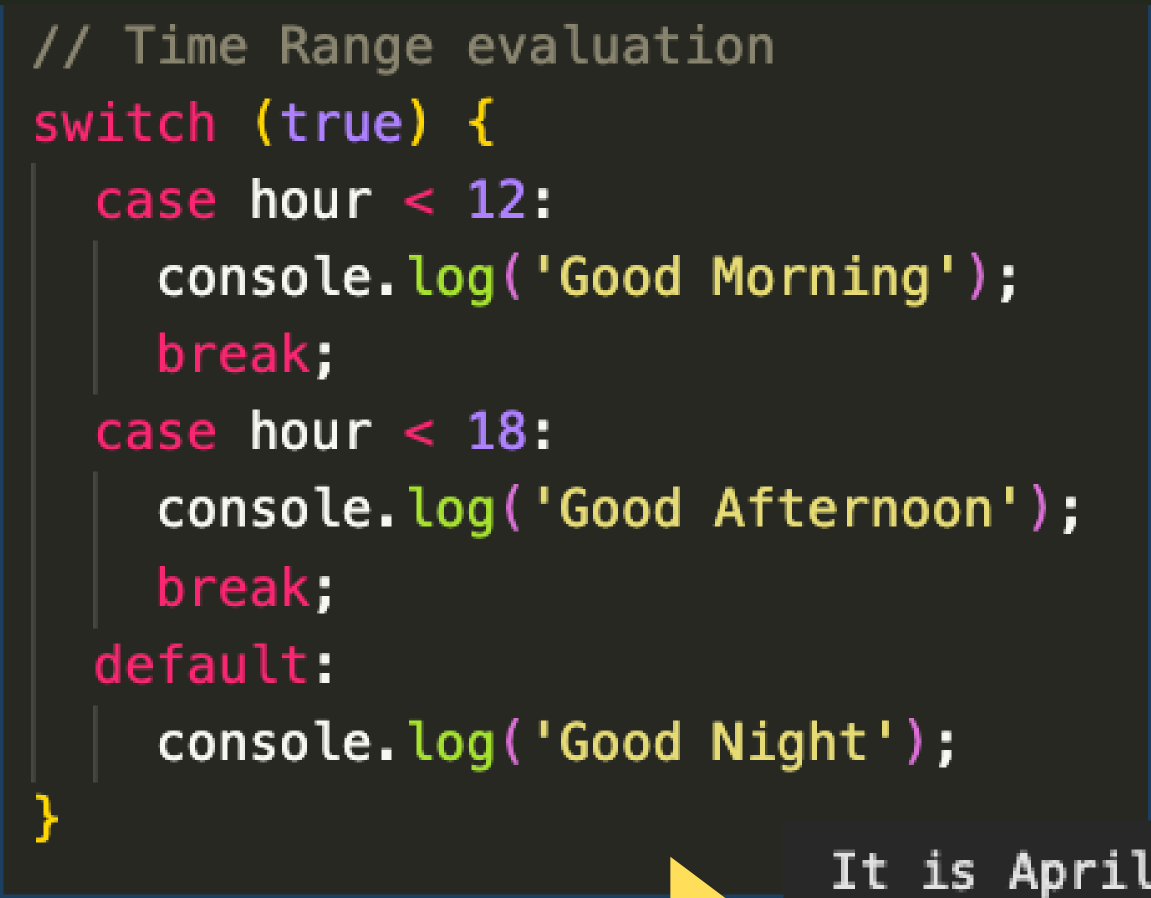
- **The switch expression is evaluated once.**
- **The value of the expression is compared with the values of each case.**
- **If there is a *match*, the associated block of code is executed.**
- **If there is no match, the default code block is executed.**



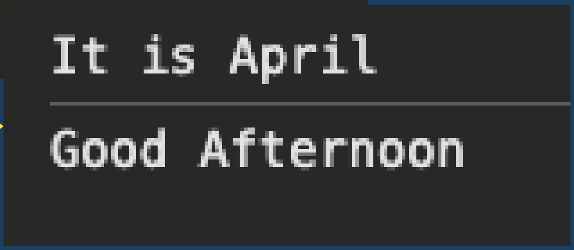
```javascript
const d = new Date(2025, 4, 3, 13, 23, 0);
const month = d.getMonth();
const hour = d.getHours();
```



```javascript
// Immediate value evaluation
switch (month) {
  case 3: // this states the number of the month March
    console.log('It is March');
    break;
  case 4: // April
    console.log('It is April');
    break;
  case 3: // May
    console.log('It is May');
    break;
  default:
    console.log('It is not March, April or May');
}
```



```javascript
// Time Range evaluation
switch (true) {
  case hour < 12:
    console.log('Good Morning');
    break;
  case hour < 18:
    console.log('Good Afternoon');
    break;
  default:
    console.log('Good Night');
}
```

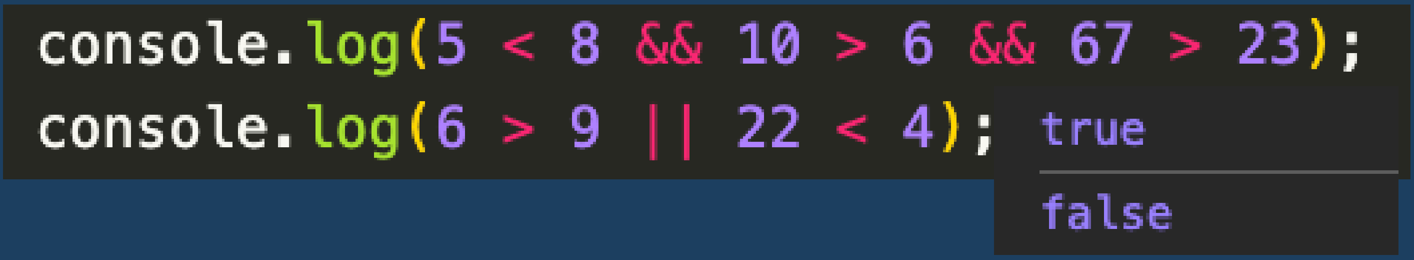


```
It is April
Good Afternoon
```

logical operators

are used to determine the logic between variables or values

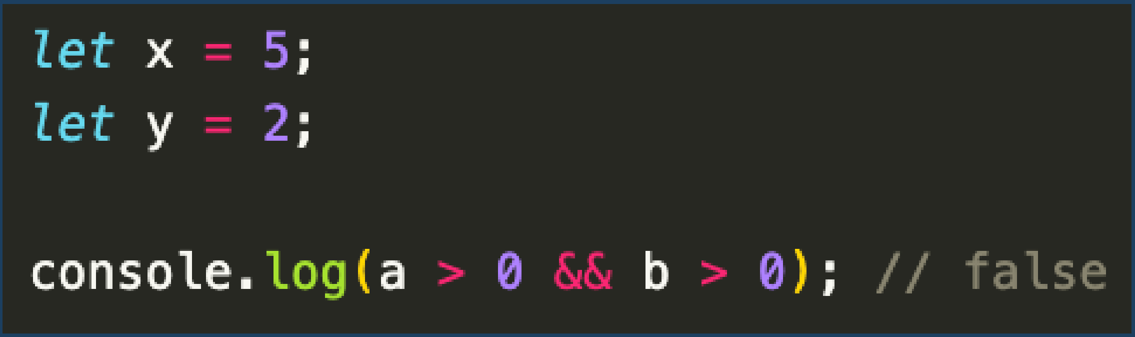
&&	and	(x < 10 && y > 1) is true
||	or	(x == 10 && y == 10) is false
!	not	!(x == y) is true



```
console.log(5 < 8 && 10 > 6 && 67 > 23);
console.log(6 > 9 || 22 < 4);
```

```
true
false
```

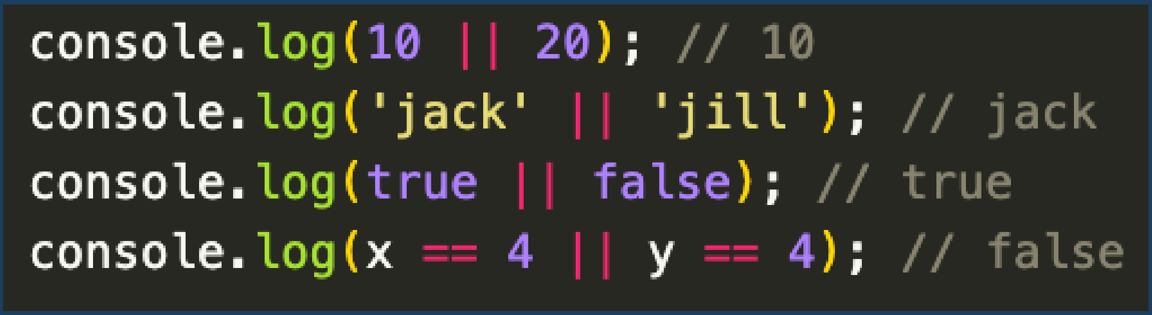
the operator returns the value of the first falsy operand encountered when evaluating from left to right, or the value of the last operand if they are all truthy



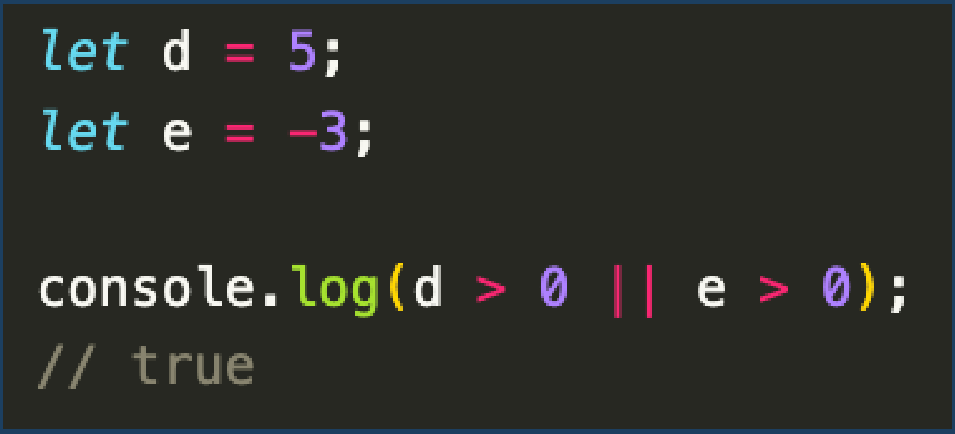
```
let x = 5;
let y = 2;

console.log(a > 0 && b > 0); // false
```

Syntax x || y

If x can be converted to true, returns x; else, returns y.



```
console.log(10 || 20); // 10
console.log('jack' || 'jill'); // jack
console.log(true || false); // true
console.log(x == 4 || y == 4); // false
```



```
let d = 5;
let e = -3;

console.log(d > 0 || e > 0);
// true
```

logical assignment operators

The Logical AND assignment operator is used between two values. If the first value is true, the second value is assigned.

&& =   (x &&= y)   x = x && (x = y)

The Logical OR assignment operator is used between two values. If the first value is false, the second value is assigned.

|| =   (x ||= y)   x = x || (x = y)

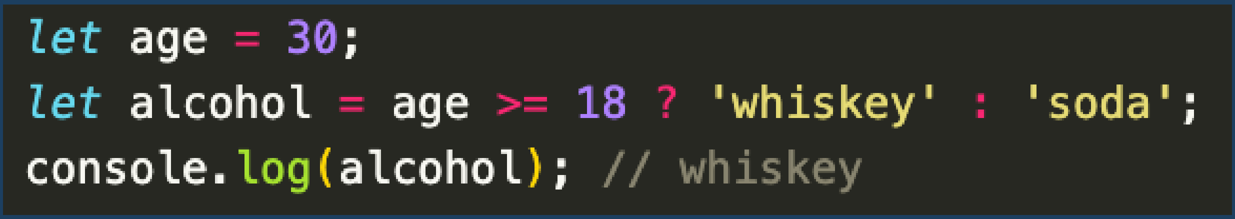
The Nullish coalescing assignment operator is used between two values. If the first value is undefined or null, the second value is assigned.

?? =   (x ??= y)   x = x ?? (x = y)

conditional ternary operator

The conditional (ternary) operator is the only JavaScript operator that takes three operands: a condition followed by a question mark (?), then an expression to execute if the condition is truthy followed by a colon (:), and finally the expression to execute if the condition is falsy. This operator is frequently used as an alternative to an if...else statement

Syntax
condition ? exprIfTrue : exprIfFalse



```
let age = 30;
let alcohol = age >= 18 ? 'whiskey' : 'soda';
console.log(alcohol); // whiskey
```