6.3 Conditionals

If statement

An ***if statement*** executes a group of statements if a condition is true. Braces { } surround the group of statements. *Good practice is to indent statements in braces using a consistent number of spaces.* This material indents 3 spaces.

Construct 6.3.1: if statement.

```
if (condition) {
   // Statements to execute when condition is
true
}
```

Feedback?

PARTICIPATION ACTIVITY 6.3.1: Evaluating if statements.

Start 2x speed

```
let fee = 30;
let age = 12;

if (age < 18) {
   fee -= 5;
}

if (age < 5) {
   fee = 0;
}

console.log("fee is $" + fee);
```

fee	25
age	12

fee is $25

Captions

1. Variable fee is assigned 30, and age is assigned 12.
2. Evaluate the condition: 12 < 18.
3. 12 < 18 is true, so the block under "if" executes, and fee is assigned 30 - 5 = 25.
4. Evaluate the condition: 12 < 5.
5. 12 < 5 is false, so the block under "if" does not execute. Variable fee remains 25.

Feedback?

PARTICIPATION ACTIVITY 6.3.2: If statement.

What is the final value of `numItems`?

1)
```
bonus = 19;
numItems = 1;
if (bonus > 10) {
   numItems = numItems
+ 3;
}
```

4

Check Show answer

Correct

4

`numItems` starts with 1. 19 > 10 is true, so the if's statements in `{ }` execute, assigning `numItems` with 1 + 3, or 4.

2)
```
bonus = 0;
numItems = 1;
if (bonus > 10) {
   numItems = numItems
+ 3;
}
```

1

Check Show answer

Correct

1

`numItems` starts with 1. 0 > 10 is false, so the if's statements in `{ }` do not execute, and `numItems` remains 1.

Feedback?

If-else statement

An ***if-else statement*** executes a block of statements if the statement's condition is true, and executes another block of statements if the condition is false.

Construct 6.3.2: if-else statement.

```
if (condition) {
   // statements to execute when condition is
true
}
else {
   // statements to execute when condition is
false
}
```

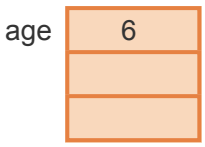
Feedback?

PARTICIPATION ACTIVITY 6.3.3: Evaluating if-else statements.

Start 2x speed

```
let age = 6;
if (age % 2 == 0) {
   console.log("age is even");
}
else {
   console.log("age is odd");
}

if (age > 10) {
   console.log("age is greater than 10");
}
else {
   console.log("age is not greater than 10");
}
```



age is even
age is not greater than 10

Captions

1. Variable age is assigned 6.
2. Evaluate the condition: 6 % 2 == 0.
3. 0 == 0 is true, so the block under "if" executes.
4. Evaluate the condition: age > 10.
5. 6 > 10 is false, so the block under "else" executes.

Feedback?

PARTICIPATION ACTIVITY 6.3.4: If-else statements.

What is the final value of `numItems`?

1)
```
bonus = 5;
if (bonus < 12) {
   numItems = 100;
}
else {
   numItems = 200;
}
```

100

Check Show answer

Correct

100

5 < 12 is true, so the if block executes, and `numItems` is assigned with 100.

2)

```
bonus = 12;
if (bonus < 12) {
   numItems = 100;
}
else {
   numItems = 200;
}
```

200

Check Show answer

Correct

200

12 < 12 is false, so the else block executes, and `numItems` is assigned with 200.

3)

```
bonus = 15;
numItems = 44;
if (bonus < 12) {
   numItems = numItems
+ 3;
}
else {
   numItems = numItems
+ 6;
}
numItems = numItems +
1;
```

51

Check Show answer

Correct

51

`numItems` is first assigned with 44. Since 15 < 12 is false, the else block executes and makes `numItems` = 44 + 6 = 50. The statement after the if-else adds 1 more, so the final value is 51.

4)

```
bonus = 5;
if (bonus < 12) {
   bonus = bonus + 2;
   numItems = 3 *
bonus;
}
else {
   numItems = bonus +
10;
}
```

21

Check Show answer

Correct

21

5 < 12 is true, so both statements in the if block execute. The first assigns `bonus` with 7. The second assigns `numItems` with 3 * 7, or 21.

Feedback?

Using { } around if and else blocks

JavaScript does not require braces `{}` around if or else blocks with a single statement. Good practice is to always use braces, which results in more readable code that is less susceptible to logic errors.

```
// Braces not required around single statements
if (vote == "M")
   memberCount++;
else
   nonMemberCount++;
```

Comparison operators

If and if-else statements commonly use comparison operators. A ***comparison operator*** compares two operands and evaluates to a Boolean value, meaning either true or false.

Table 6.3.1: Comparison operators.

Comparison operator	Name	Example
==	Equality	`2 == 2 // true` `"bat" == "bat" // true`
!=	Inequality	`2 != 3 // true` `"bat" != "zoo" // true`
===	Identity	`2 === 2 // true` `"2" === 2 // false`
!==	Non-identity	`2 !== 2 // false` `"2" !== 2 // true`
<	Less than	`2 < 3 // true` `"bat" < "zoo" // true`
<=	Less than or equal	`2 <= 3 // true` `"bat" <= "bat" // true`

Comparison operator	Name	Example
>	Greater than	`3 > 2 // true` `"zoo" > "bat" // true`
>=	Greater than or equal	`3 >= 2 // true` `"zoo" >= "zoo" // true`

Feedback?

When the equality operator == and inequality != operator compare a number and a string, the string is first converted to a number and then compared. Ex: `3 == "3"` is true because "3" is converted to 3 before the comparison, and 3 and 3 are the same.

The ***identity operator*** === performs ***strict equality***. Two operands are strictly equal if the operands' data types and values are equal. Ex: `3 === 3` is true because both operands are numbers and the same value, but `"3" === 3` is false because "3" is a string, and 3 is a number. The ***non-identity operator*** !== is the opposite of the identity operator. Ex: `"3" !== "3"` is false because both operands are the same type and value, but `"3" !== 3` is true because "3" is a string, and 3 is a number.

Other comparison operators also convert a string to a number when comparing a string with a number. Ex: `2 < "12"` is true because 2 is less than the number 12. When comparing two strings, JavaScript uses Unicode values to compare characters. Ex: `"cat" <= "dog"` is true because "c" has a smaller Unicode value than "d".

A common error when comparing two values for equality is to use a single = instead of == or ===. Ex: `if (name = "Sue")` assigns `name` with "Sue" instead of asking if `name` equals "Sue".

What is Unicode?

Unicode *is a computing industry standard that assigns a unique number to characters in over one hundred different languages, including multiple symbol sets and emoji. The Unicode numbers for capital A-Z range from 65 to 90, and lowercase a-z range from 97 to 122.*

PARTICIPATION ACTIVITY 6.3.5: Comparison operators.

Is the if statement true or false?

1)
```
score = 2;
if (score == "2")
{
   score = 10;
}
```

- (•) true
- ( ) false

Correct

2 == "2" is true because "2" is converted into the number 2 first, and 2 == 2 is true.

2)
```
score = 2;
if (score = "50")
{
   score = 100;
}
```

- (•) true
- ( ) false

Correct

A common error is to use = instead of ==. The = operator assigns `score` with "50" instead of asking if 2 is equal to 50 (2 == 50). When a variable is assigned with a non-zero value in an if statement, the if statement is true.

3)
```
score = 2;
if (score ===
"2") {
   score = 10;
}
```

- ( ) true
- (•) false

Correct

2 === "2" is false because the identity operator returns false when the two operands are not the same data type. `score` is the number 2, but "2" is a string.

4)
```
status = "10";
if (status > 5) {
   length = 0;
}
```

- (•) true
- ( ) false

Correct

A string is compared to a number, so "10" is first converted to 10 before the comparison. 10 > 5 is true.

5)
```
status = "good";
if (status >
"bad") {
   bonus += 2;
}
```

- (•) true
- ( ) false

Correct

The two strings "good" and "bad" are compared, so Unicode values of each character are compared. The Unicode value for "g" is > "b", so "good" > "bad" is true.

6)
```
status =
"charge";
if (status <=
"chance") {
   bonus += 2;
}
```

- ○ true
- ◉ false

Correct

Both strings start with "cha" and are equal until the 4th character. The Unicode value for "r" is > "n", so "charge" <= "chance" is false.

7)
```
lowerCaseLetters
= "abc";
upperCaseLetters
= "ABC";
if
(lowerCaseLetters
>
upperCaseLetters)
{
   length++;
}
```

- ◉ true
- ○ false

Correct

The Unicode value for "a" (97) is > "A" (65), so "abc" > "ABC" is true.

Feedback?

Nested statements and else-if statement

If and else block statements can include any valid statements, including another if or if-else statement. An if or if-else statement that appears inside another if or if-else statement is called a ***nested*** statement.

PARTICIPATION ACTIVITY 6.3.6: Nested if-else statement example.

Start ☐ 2x speed

```
let userAge = 18;
if (userAge <= 12) {
   console.log("Enjoy your early years.");
}
else {
   console.log("You are at least 13.");

   if (userAge >= 18) {
      console.log("You are old enough to vote.");
   }
   else {
      console.log("You are too young to vote.");
   }
}
```

userAge 18

```
You are at least 13.
You are old enough to vote.
```

Captions ^

1. userAge is 18. Since 18 <= 12 is false, the outer if-else statement's else block executes.
2. After outputting to the console, the nested if-else statement executes.
3. 18 >= 18 is true, so the if block executes.
4. No more statements exist in the nested if-else statement or the outer if-else statement.

Feedback?

A common situation is when several nested if-else statements are needed to execute one and only one block of statements. The ***else-if*** statement is an alternative to nested if-else statements that produces an easier-to-read list of statement blocks.

In the example below, the `grade` variable is assigned with A, B, C, D, or F depending on the `score` variable. The code segment on the left uses nested if-else statements. The code segment on the right performs the same logic with else-if statements.

Figure 6.3.1: Nested if-else statements vs. else-if statements.

```
// Nested if-else statements
if (score >= 90) {
   grade = "A";
}
else {
   if (score >= 80) {
      grade = "B";
   }
   else {
      if (score >= 70) {
         grade = "C";
      }
      else {
         if (score >= 60) {
            grade = "D";
         }
         else {
            grade = "F";
         }
      }
   }
}
```

```
// else-if statements
if (score >= 90) {
   grade = "A";
}
else if (score >= 80) {
   grade = "B";
}
else if (score >= 70) {
   grade = "C";
}
else if (score >= 60) {
   grade = "D";
}
else {
   grade = "F";
}
```

Feedback?

PARTICIPATION ACTIVITY 6.3.7: Nested if-else practice.

Use a nested if-else statement or an else-if statement to examine the variable `golfScore`.

- If `golfScore` is above 90, output to the console "Keep trying!"
- Otherwise, if `golfScore` is above 80, output to the console "Nice job!"
- Otherwise, output to the console "Ready to go pro!"

Test your code with values above 90, between 81 and 90, and 80 and below to ensure your logic is correct.

```
let golfScore = 95;

// Write your if-else statements here!

```

Run JavaScript Reset code

Your console output

▼ View solution

Explain

```
--- START FILE: JavaScript ---


let golfScore = 95;

// Write your if-else statements here!
if (golfScore > 90) {
   console.log("Keep trying!");
}
else if (golfScore > 80) {
```

```
    console.log("Nice job!");
}
else {
    console.log("Ready to go pro!");
}


--- END FILE: JavaScript ---
```

Feedback?

PARTICIPATION ACTIVITY 6.3.8: Nested if and if-else statements.

What is `numBoxes` at the end of each code segment?

1)
```
numApples = 2;
numOranges = 5;
numBoxes = 0;
if (numApples % 2 != 0)
{
   numBoxes = 1;
}
else {
   if (numApples +
numOranges > 10) {
      numBoxes = 2;
   }
   else {
      numBoxes = 99;
   }
}
```

Check Show answer

Answer

99

In the outer if-else, 2 % 2 != 0 is false, so the else block executes. In the nested if-else, 2 + 5 = 7, and 7 > 10 is false, so the nested else block executes.

2)

```
numApples = 2;
numOranges = 5;
numBoxes = 0;
if (numApples > 0) {
   if (numOranges > 10)
{
      numBoxes = 4;
   }
   numBoxes++;
}
else {
   numBoxes = 99;
}
```

Check Show answer

Answer

1

`numBoxes` is initially 0. In the outer if-else, 2 > 0 is true, so the if block executes. In the nested if, 5 > 10 is false, so `numBoxes` is not assigned with 4. `numBoxes++` adds one to `numBoxes`, making `numBoxes` 1.

3)

```
produce = "carrots";
if (produce ==
"apples") {
   numBoxes = 1;
}
else if (produce ==
"bananas") {
   numBoxes = 2;
}
else if (produce ==
"carrots") {
   numBoxes = 3;
}
else {
   numBoxes = 4;
}
```

Check Show answer

Answer

3

The first if condition "carrots" == "apples" is false. The second if condition "carrots" == "bananas" is false. The third if condition "carrots" == "carrots" is true, so `numBoxes` is assigned with 3. The else block does not execute.

Feedback?

Logical operators

JavaScript logical operators perform AND, OR, and NOT logic.

Table 6.3.2: Logical operators.

Logical operator	Name	Description	Example
`&&`	And	True if both sides are true	`(1 < 2 && 2 < 3)  // true`
`||`	Or	True if either side is true	`(1 < 2 || 2 < 0)  // true`
`!`	Not	True if expression is not true	`!(2 == 2)  // false`

Feedback?

Multiple `&&` and `||` conditions may be combined into a single ***complex condition***. Ex: `(1 < 2 && 2 < 3 || 3 < 4)`. Complex conditions are evaluated from left to right, but `&&` has higher precedence than `||`, so `&&` is evaluated before `||`. *Good practice is to use parentheses `()` around conditions that use `&&` and `||` to explicitly indicate the order of evaluation. Ex: `(a < 0 || a > 1 && b > 2)` is better expressed as: `(a < 0 || (a > 1 && b > 2))`.*

Logic involving "not" can be difficult for humans to correctly read or understand. Ex: "Are you not hungry?" is more difficult for a human to understand than the equivalent "Are you satisfied?" *Good practice is to avoid using the not operator when possible. Ex: `!(score > 10)` is better expressed as: `score <= 10`.*

PARTICIPATION ACTIVITY 6.3.9: Evaluating complex conditions.

What is `decision` at the end of each code segment?

1)
```
homeTeam = 2;
visitingTeam = 5;
if (homeTeam > 10 ||
visitingTeam > 0) {
   decision = 1;
}
else {
   decision = 0;
}
```

Answer

1

2 > 10 is false, but 5 > 0 is true. (false || true) is true.

Check Show answer

2)

```
homeTeam = 2;
visitingTeam = 5;
if (!(homeTeam > 10 &&
visitingTeam > 0)) {
   decision = 1;
}
else {
   decision = 0;
}
```

1

Check Show answer

Answer

1

homeTeam > 10 is false, and visitingTeam > 0 is true. (false && true) is false, and !false is true. The if statement can be written without the ! as: `if (homeTeam <= 10 || visitingTeam <= 0)`

3)

```
homeTeam = 2;
visitingTeam = 5;
if (homeTeam > 10 ||
(visitingTeam != 2 &&
visitingTeam > 0)) {
   decision = 1;
}
else {
   decision = 0;
}
```

Check Show answer

Answer

1

false || (true && true) = false || true = true

Feedback?

CHALLENGE ACTIVITY 6.3.1: Conditionals.

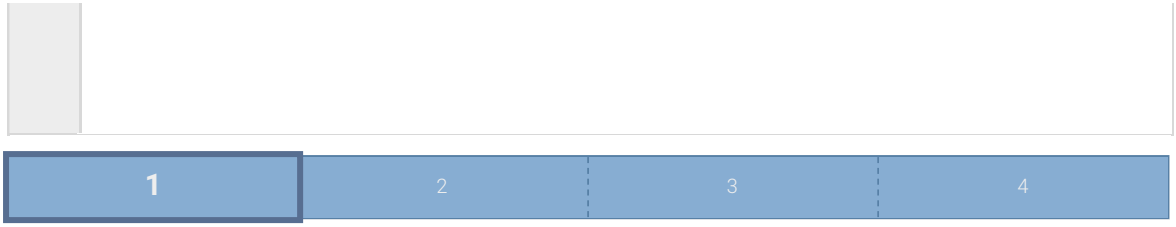
530096.4000608.qx3zqy7

Start

Write an if-else statement that if userTickets is greater than or equal to 5, executes awardPoints = 10. Else, execute awardPoints = userTickets. Ex: If userTickets is 6, then awardPoints = 10.

```
1 let awardPoints = 0;
2 let userTickets = 4; // Code will be tested with values: 4, 5, ar
3
4 /* Your solution goes here */
5
```

1 2 3 4



1 2 3 4

Check Next

View your last submission

Feedback?

How was this section?

Provide section feedback