# 6.4 More conditionals

## Truthy and falsy

A ***truthy*** value is a non-Boolean value that evaluates to `true` in a Boolean context. Ex: `if (18)` evaluates to `true` because non-zero numbers are truthy values. A ***falsy*** value is a non-Boolean value that evaluates to `false` in a Boolean context. Ex: `if (null)` evaluates to `false` because `null` is a falsy value.

Table 6.4.1: Truthy values.

| Example | Description |
|---|---|
| `if (32)` | Non-zero number |
| `if ("cat")` | Non-empty string |
| `if (myObject)` | Object variable |
| `if (myArray)` | Array variable |

Feedback?

Table 6.4.2: Falsy values.

| Example | Description |
|---|---|
| `if (0)` | Zero |
| `if ("")` | Empty string |
| `if (NaN)` | Not a number |
| `if (undefined)` | Variable that has not been assigned a value |
| `if (null)` | No object value |

Feedback?

PARTICIPATION ACTIVITY 6.4.1: Truthy and falsy values. ✓

Indicate if the `if` statement's condition evaluates to `true` or `false`.

1) `if (undefined)` ✓

true [✓ false]

**Correct**

`undefined` is falsy.

2) `if (999)` ✓

[✓ true] false

**Correct**

Non-zero numbers are truthy.

3) `if (0)` ✓

true [✓ false]

**Correct**

Zero is falsy.

4) `if ("")` ✓

true [✓ false]

**Correct**

Empty strings are falsy.

5) `if (" ")` ✓

[✓ true] false

**Correct**

Non-empty strings are truthy.

6) `if ("false")` ✓

**Correct**

[✓ true] false

Quotes around the word `false` creates a non-empty string, which is truthy.

7) `if (myArray)`

[✓ true] false

**Correct**

Objects (arrays are a type of object) are truthy.

Feedback?

## Conditional (ternary) operator

The conditional operator allows developers to write concise conditional statements. The ***conditional operator*** (or ***ternary operator***) has three operands separated by a question mark (`?`) and colon (`:`). If the `condition` evaluates to `true`, then the value of `expression1` is returned, otherwise the value of `expression2` is returned.

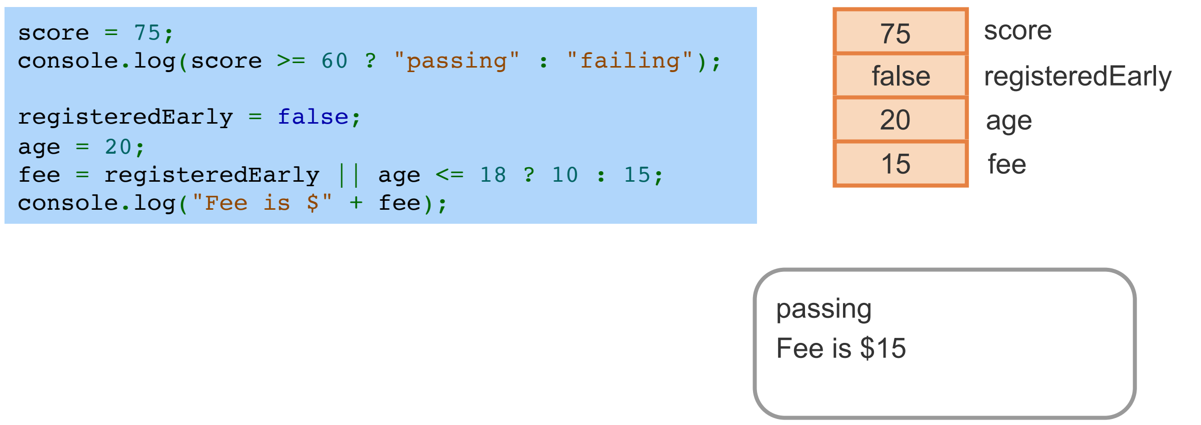
### Construct 6.4.1: Conditional (ternary) operator.

```
condition ? expression1 :
expression2
```

Feedback?

**PARTICIPATION ACTIVITY** 6.4.2: Evaluating the conditional operator.

1 2 3 4 ✓ 2x speed



```
score = 75;
console.log(score >= 60 ? "passing" : "failing");

registeredEarly = false;
age = 20;
fee = registeredEarly || age <= 18 ? 10 : 15;
console.log("Fee is $" + fee);
```

| | |
|---|---|
| 75 | score |
| false | registeredEarly |
| 20 | age |
| 15 | fee |

```
passing
Fee is $15
```

Ternary operator returns 15, so fee is assigned 15 and output to the console.

Captions

1. 75 >= 60 evaluates to true.
2. Ternary operator returns "passing", so "passing" is displayed in the console.
3. false || 20 <= 18 is false.
4. Ternary operator returns 15, so fee is assigned 15 and output to the console.

Feedback?

PARTICIPATION ACTIVITY | 6.4.3: Conditional operator.

1) Complete the code to assign `lateStatus` with "yep" if `currTime` is greater than 60, and "nope" otherwise.

```
lateStatus = currTime >
60 ? "yep" :
"nope";
```

Check Show answer

**Correct**

?

The ? character follows the condition.

2) Complete the code to assign `y` with `x` if `x` is greater than 0, and -1 otherwise.

```
y = (x > 0) ?
x:-1 ;
```

Check Show answer

**Correct**

x : -1

`x` is evaluated and assigned to `y` when `(x > 0)` is true. -1 is evaluated and assigned to `y` when `(x > 0)` is false.

3) What is `boardType` after the following statements?

```
year = 1985;
boardType = year >=
2015 ? "hoverboard" :
"skateboard";
```

skateboard

Check Show answer

**Correct**

skateboard

Since `year` is not >= 2015, the ternary operator returns the second expression.

4) What is `priority` after the following statements?

```
attempt = 4;
priority = 2;
attempt > 3 ?
priority++ : priority-
-;
```

3

Check Show answer

Correct

3

Since `attempt > 3` is true, the conditional operator executes the expression `priority++`, which adds one to `priority`.

Feedback?

## Switch statement

The switch statement is an alternative to writing multiple else-if statements. A ***switch statement*** compares an expression's value to several cases using strict equality (===) and executes the first matching case's statements. If no case matches, an optional default case's statements execute.

The ***break statement*** stops executing a case's statements and causes the statement immediately following the switch statement to execute. Omitting the break statement causes the next case's statements to execute, even though the case does not match.

Construct 6.4.2: switch statement.

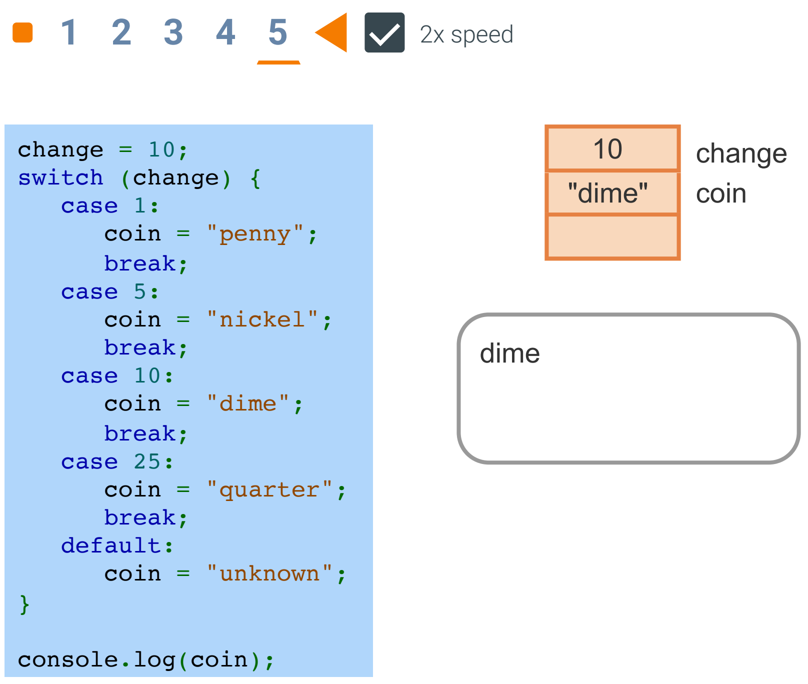
```
switch (expression) {
   case value1:
      // Statements executed when expression's value matches
value1
      break;   // optional
   case value2:
      // Statements executed when expression's value matches
value2
      break;   // optional

   // ...

   default:
      // Statements executed when no cases match
}
```

Feedback?

PARTICIPATION ACTIVITY 6.4.4: Evaluating the switch statement.



```
change = 10;
switch (change) {
   case 1:
      coin = "penny";
      break;
   case 5:
      coin = "nickel";
      break;
   case 10:
      coin = "dime";
      break;
   case 25:
      coin = "quarter";
      break;
   default:
      coin = "unknown";
}

console.log(coin);
```

Break statement stops executing the switch statement. The code after the switch executes, outputting "dime" to the console.

Captions

1. switch statement examines the change variable.
2. change === 1 is false, so the case does not match.
3. change === 5 is false, so the case does not match.
4. change === 10 is true, so the case matches, and the case's statements are executed.
5. Break statement stops executing the switch statement. The code after the switch executes, outputting "dime" to the console.

**Feedback?**

**PARTICIPATION ACTIVITY** 6.4.5: switch statement.

Refer to the switch statement below.

```
switch (item) {
   case "apple":
   case "orange":
      fruits++;
      break;
   case "milk":
      drinks++;
   case "cheese":
      dairy++;
      break;
   case "beef":
   case "chicken":
      meat++;
      break;
   default:
      other++;
}
```

1) If `item` is "beef", what variables are incremented?

- ○ `other`
- ◉ `meat` only ✓
- ○ `meat` and `other`

**Correct**

After incrementing `meat`, the `break` statement stops executing code in the switch statement.

2) If `item` is "milk", what variables are incremented?

- ○ `other`
- ○ `drinks` only
- ◉ `drinks` and `dairy` ✓

**Correct**

The statements under the "milk" and "cheese" cases are executed since the "milk" case does not end with a `break` statement.

3) If `item` is "Apple", what variable is incremented?

- ◉ `other` ✓
- ○ `fruits`
- ○ Nothing is incremented.

**Correct**

"Apple" does not === "apple" or any other cases, so the `default` statement is executed.

Feedback?

**PARTICIPATION ACTIVITY** 6.4.6: Practice with the switch statement.

Convert the group of else-if statements into an equivalent switch statement.

```

// Get a number between 0 and 6 representing the day of the we
let currDay = new Date().getDay();

// Convert into an equivalent switch statement
if (currDay === 1) {
   console.log("I love Mondays!");
}
else if (currDay === 2 || currDay === 3 || currDay === 4) {
   console.log("Working hard!");
}
else if (currDay === 5) {
   console.log("TGIF!");
}
else {
   console.log("Time to relax!");
```

Run JavaScript

Reset code

**Your console output**

```
Working hard!
```

▶ View solution

Feedback?

**CHALLENGE ACTIVITY** 6.4.1: More conditionals.

530096.4000608.qx3zqy7

Jump to level 1

1 2

Write a switch statement that examines inputItem. If 5, print "HTML". If 6, print "CSS". If 7, print "JavaScript". For any other value, print "PHP".

```
// Your code will be tested with 5 and other values
let inputItem = 5;
```

```

/* Your solution goes here */
switch(inputItem){
   case 5:
      console.log("HTML");
      break;
   case 6:
      console.log("CSS");
      break;
   case 7:
      console.log("JavaScript");
      break;
   default:
      console.log("PHP");
```

1 2

Check Try again **Done**. Click any level to practice more. Completion is preserved.

✓

✓ Testing displayed output with inputItem = 5

Yours `HTML`

✓ Testing displayed output with inputItem = 6

Yours `CSS`

✓ Testing displayed output with inputItem = 7

Yours `JavaScript`

✓ Testing displayed output with inputItem = 8

Yours `PHP`

Feedback?

How was this section? Provide section feedback