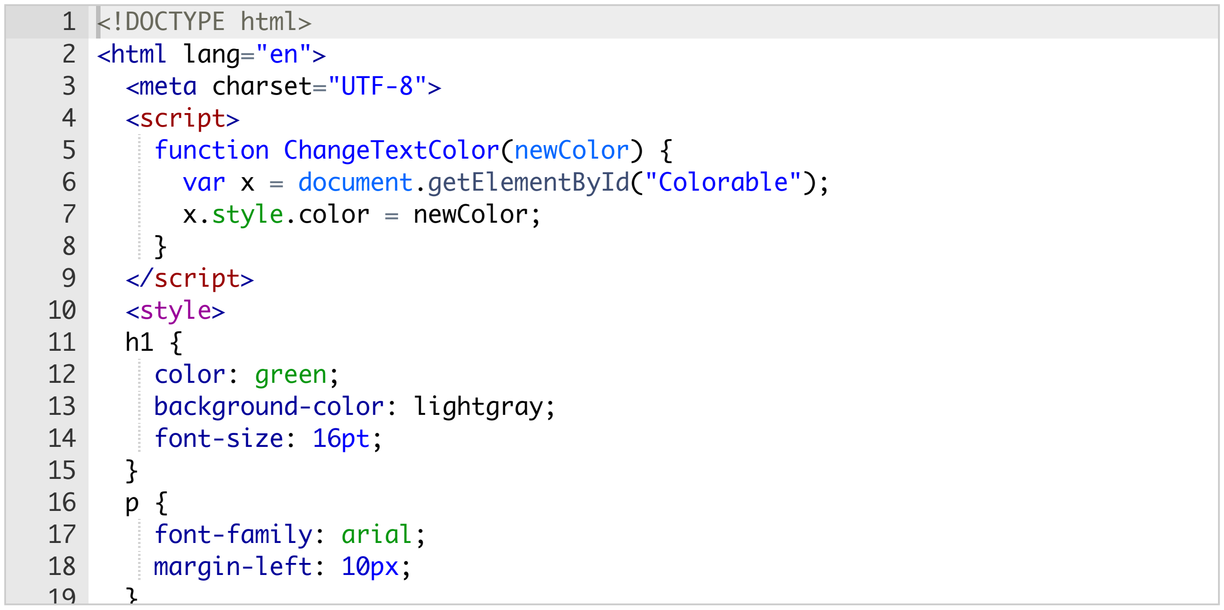
8.1 Introduction to JavaScript

JavaScript is a programming language that runs in a browser, enabling web pages supporting actions like responding to a button click. JavaScript can be included in the HTML file's header part.

PARTICIPATION ACTIVITY 8.1.1: JavaScript to change colors.

Click the buttons. Try adding a third button for "blue".



```
<!DOCTYPE html>
<html lang="en">
  <meta charset="UTF-8">
  <script>
    function ChangeTextColor(newColor) {
      var x = document.getElementById("Colorable");
      x.style.color = newColor;
    }
  </script>
  <style>
  h1 {
    color: green;
    background-color: lightgray;
    font-size: 16pt;
  }
  p {
    font-family: arial;
    margin-left: 10px;
  }
```

Render web page   Reset code

Your web page

In the HTML above:

- The script tags indicates JavaScript code, consisting of a function named ChangeTextColor. A JavaScript ***function*** is a named group of statements that can be run by referring to that name.
- Two button elements are created, each with an attribute named onclick. The onclick attribute is set to the function ChangeTextColor. Thus, when a button is clicked, the function ChangeTextColor is run, using the value passed to the function (either 'white' or 'green').
- The h1 heading has an id of Colorable. The ChangeTextColor function's statements change the h1's color. The function uses document.getElementById("idName"), which searches the HTML document for and returns an element whose id="idName". newColor and x are both examples of variables. A ***variable*** stores a value or a link to an element of a web page. Using the link stored in a variable allows JavaScript to directly modify the properties of an element defined elsewhere in the HTML document. Ex: x stores the element with id="Colorable".

PARTICIPATION ACTIVITY 8.1.2: The colorable JavaScript example.

x = document.getElementById("Colorable")

onclick="ChangeTextColor('white')"

<button type="button" onclick="ChangeTextColor('blue')">Blue</button>

<h1 id="ColorableText">...

	Gives this h1 heading a label, so that a function can find the heading to change the heading's color.
	Indicates that the JavaScript function ChangeTextColor should be executed, with the value 'white'.
	Finds the element with id 'Colorable', and sets x to that element. x can then be used to change that element's attributes, like color.
	The HTML that should be added to create a third button.

Reset

The JavaScript example below shows a function with an "if-else" statement for setting the color of the rating stars based on the value passed to the function. The HTML below defines five span elements, which are inline containers used to manage HTML content. Each span element has a unique id and contains a single * for the rating star. The JavaScript code can change each rating stars' color by changing the span's color.

PARTICIPATION ACTIVITY 8.1.3: Updating user ratings.

Click the buttons. Try adding three more buttons for ratings 3, 2, and 1. Try replacing the * with a star (★). To specify a star, use &#9733;, which is the HTML entity for displaying a star.

```
<!DOCTYPE html>
<html>
  <meta charset="UTF-8">
  <script>
    function UpdateRating(newRating) {
      var star1 = document.getElementById("rating1");
      var star2 = document.getElementById("rating2");
      var star3 = document.getElementById("rating3");
      var star4 = document.getElementById("rating4");
      var star5 = document.getElementById("rating5");

      if (newRating == 5) {
        star5.style.color = "blue";
        star4.style.color = "blue";
        star3.style.color = "blue";
        star2.style.color = "blue";
        star1.style.color = "blue";
      }
      else if (newRating == 4) {
```

Render web page   Reset code

Your web page

PARTICIPATION ACTIVITY 8.1.4: JavaScript for updating user ratings.

Refer to the JavaScript example above

1) What is the id of the span containing the third rating star?

Check   Show answer

2) If the user clicks the "Rate 4" button, to what color is the fourth rating star set?

Check Show answer

More advanced interactive web pages, such as a user-entry form or a browser-based video game, may involve hundreds or thousands of JavaScript statements. JavaScript programs are thus commonly placed in a separate file, typically ending in .js, and linked to in an HTML file's head part.

Here are two versions of the popular game Tetris, written in JavaScript: Tetris1, Tetris2.

PARTICIPATION ACTIVITY | 8.1.5: JavaScript example: Analog clock.

JavaScript can also be used to draw graphics. Play around with this clock by changing the time. Note: No changes are needed.

```
<!DOCTYPE html>
<html>
  <meta charset="UTF-8">
  <style>
  .hour-input {
    color: blue;
    margin-right: 5px;
    width: 30px
  }
  .minute-input {
    color: red;
    margin-left: 5px;
    width: 30px
  }
  </style>
  <body>
    <canvas id='clockCanvas' width='200' height='200'></canvas>
    <div class='button-container'>
      <button onclick='currentTime()'>Draw current time</button>
```

Render web page Reset code

Your web page

8.2 Syntax and variables

In 1995, Brendan Eich created JavaScript so the Netscape Navigator browser could dynamically respond to user events. Ex: The web page's content could change when the user clicked a button. JavaScript was standardized by Ecma International in 1997 and called ECMAScript. ***ECMAScript*** is the standardized language that has been improved over the years, and JavaScript is an implementation of ECMAScript. The latest version of ECMAScript is version 7 (***ES7***) and was released in 2016.

Today, JavaScript is one of the most popular programming languages. JavaScript is supported by every major web browser and makes web applications like Gmail and Google Maps possible. JavaScript is also popular outside of the web browser. Ex: Node.js, which runs JavaScript, is a popular technology for creating server-side web applications.

JavaScript is executed by an interpreter. An ***interpreter*** executes programming statements without first compiling the statements into machine language. Modern JavaScript interpreters (also called ***JavaScript engines***) use ***just-in-time (JIT) compilation*** to compile the JavaScript code at execution time into another format that can be executed quickly.

PARTICIPATION ACTIVITY 8.2.1: JavaScript background.

1) ECMAScript and JavaScript are the same thing.

○ True

○ False

2) JavaScript is only used for programs that run in a web browser.

○ True

○ False

3) Chrome and Firefox use the same JavaScript engine.

○ True

○ False

4) JavaScript is syntactically identical to the Java programming language.

○ True

○ False

ECMAScript name

The name "ECMAScript" was a compromise between Netscape, Microsoft, and other organizations involved in the standardization of JavaScript. Brendan Eich once commented that "ECMAScript was always an unwanted trade name that sounds like a skin disease." Despite ECMAScript's similarity to eczema (a group of related skin diseases), the name has stuck.

JavaScript is syntactically similar to many programming languages like C++ and Java. However, JavaScript programming is simpler in some ways. For example, variables are declared with the keyword ***var***. A ***variable*** is a named container that stores a value. A variable does not have to be declared before being assigned a value.

Figure 8.2.1: Declaring variables.

```
// Declaring variable x
var x;

// x is assigned a number
x = 5;

// x is dynamically re-assigned a string
x = "hello";

// y may be assigned a value without first being declared
```

```
y = 10;
```

A name created by a programmer for an item like a variable or function is called an ***identifier***. JavaScript imposes the following rules for identifiers:

- An identifier can be any combination of letters, digits, underscores, or $.
- An identifier may not start with a digit.
- An identifier may not be a reserved word like `var`, `function`, or `while`.

A JavaScript coding convention is to name JavaScript variables with camel casing, where the identifier starts with a lowercase letter, and subsequent words begin with a capital letter. Ex: `lastPrice` is preferred over `LastPrice` or `last_price`.

PARTICIPATION ACTIVITY 8.2.2: Declaring and naming variables.

1) Which statement declares the variable `sum` without assigning a value to `sum`?

- ○ `sum;`
- ○ `var sum;`
- ○ `sum = 0;`

2) Which identifier is illegally named?

- ○ `star_destroyer`
- ○ `$save`
- ○ `9to5`

3) Which variable is named with the preferred JavaScript naming conventions?

- ○ `total_points`
- ○ `$totalPoints`
- ○ `totalPoints`

Variables are not explicitly assigned a data type. JavaScript uses dynamic typing, which allows the same JavaScript variable to be assigned different data types. ***Dynamic typing*** determines a variable's type at run-time. Every variable is assigned one of the data types listed in the table below.

Table 8.2.1: JavaScript data types.

Data type	Description	Example
string	Strings can be delimited with 'single' or "double" quotes	`var name = "Sam";` `var quote = 'The computer asked, "Shall we play a game?"';`
number	Numbers may have decimal places or not	`var highScore = 950;` `var pi = 3.14;`
boolean	true or false	`var hungry = true;` `var thirsty = false;`
array	List of items	`var teams = ["Broncos", "Cowboys", "49ers"];`
object	Collection of property and value pairs	`var movie = {title:"Life is Beautiful", rating:"PG-13"};`
undefined	Variable without a value	`var message;`
null	Intentionally absent of any object value	`var book = null;`

PARTICIPATION ACTIVITY 8.2.3: Variable data types.

1) What is the data type of `z`?

```
var z;
```

- undefined
- string
- number

2) What is the data type of `x`?

```
y = false;
x = y;
```

- string

○ boolean

○ number

3) What is syntactically wrong with the following code?

```
name = 'Danny O'Sullivan';
```

○ `name` is assigned a value without being declared first.

○ Variables may not be assigned strings delimited with single quotes.

○ The single quotation mark in O'Sullivan is a syntax error.

Constants

*A **constant** is an initialized variable whose value cannot change. ECMAScript 6 introduced the `const` keyword to define a constant. Most modern browsers support or partially support `const`.*

```
const PI = 3.14;
const TAX_MONTH = "April";
```

JavaScript uses the `//` and `/* */` operators to produce comments in code, similar to C++ and Java. A ***comment*** is any text intended for humans that is ignored by the JavaScript interpreter.

Figure 8.2.2: Comments.

```
// Single line comment

/* Multi-line
   comment
*/
```

JavaScript does not require that statements be terminated with a semicolon. Only when two statements reside on the same line must a semicolon separate the two statements. Good practice is to avoid placing two statements on the same line. Some developers prefer to use semicolons at the end of statements, and others do not. Good practice is to consistently use semicolons or not throughout the code.

Figure 8.2.3: Using semicolons.

```
var totalPoints = 10;

// No semicolon is required
var totalLives = 3

// Two statements on the same line require a semicolon
totalPoints = 5; totalLives = 2
```

PARTICIPATION ACTIVITY 8.2.4: Detect the error.

Indicate if the statements contain an error or not.

1)
```
/* a is assigned 2
a = 2;
```
○ Error
○ No error

2)
```
3.12 = pi;
```
○ Error
○ No error

3)
```
x = 10; var y = 20;
```
○ Error
○ No error

A JavaScript program may obtain text input from the user with the `prompt()` function. The ***prompt()*** function prompts the user with a dialog box that allows the user to type a single line of text and press OK or Cancel. The `prompt()` function returns the string the user typed or `null` if the user pressed Cancel.

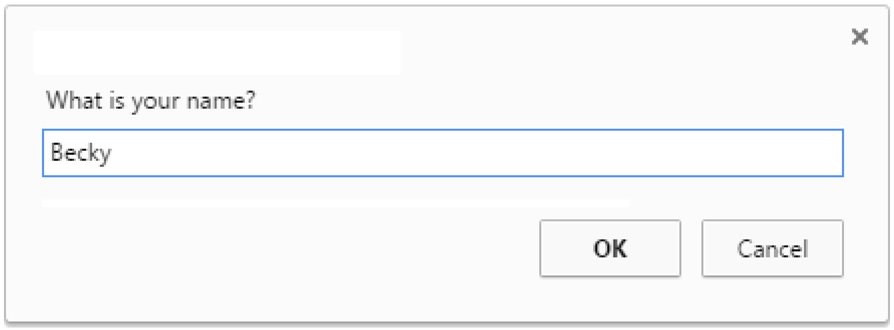
Output may be produced using the function ***console.log()***, which displays text or numbers in the console. The ***console*** is a location where text output is displayed. Web browsers have a console (accessible from the brower's development tools) that displays output from code the browser executes. This chapter's activities display the console output in the web page.

Figure 8.2.4: Using prompt() to get input from

the user.

```
// Display the prompt dialog box
var name = prompt("What is your name?");

// Output to the console
console.log("Hello, " + name + "!");
```


What is your name?

Becky

```
Hello, Becky!
```

PARTICIPATION ACTIVITY | 8.2.5: prompt() and console.log().

1) Write the function call to display the `question` variable to the user and retrieve the user's age.

```
question = "How old are you?";
age = ______;
```

Check Show answer

2) Write the code to display "You are X", where X is the value of the `age` variable.

```
age = 21;
console.log(______);
```

Check Show answer

PARTICIPATION ACTIVITY 8.2.6: JavaScript practice.

The JavaScript code below initializes the variable `tvShow` to a popular TV show. Then, an `if` statement displays a message in the console if `tvShow` is `null`, otherwise the value of `tvShow` is displayed in the console. Change the code to prompt the user for the user's favorite TV show. Then, display "____ is your favorite TV show!" in the console. Press "Run JavaScript" to run your code.

Note: The console will display an error message if the JavaScript interpreter detects a syntax error. A ***syntax error*** is the incorrect typing of a programming statement. Ex: Forgetting to place "quotes" around a string value is a syntax error.

```
// Use prompt to get the user's favorite TV show
var tvShow = "Sherlock";

if (tvShow === null) {
   console.log("You did not enter a TV show.");
}
else {
   console.log(tvShow);
}

```

Run JavaScript   Reset code

Your console output

CHALLENGE ACTIVITY 8.2.1: prompt() and console.log().

Start

Write a statement that displays: This is awesome!

```

/* Your solution goes here */

```

1 2 3

Check Next

Exploring further:

- A Short History of JavaScript from W3C.org
- JavaScript Lexical Grammar from MDN

8.3 Arithmetic

JavaScript arithmetic operators perform arithmetic computations. Multiplication (`*`) and division (`/` and `%`) have higher precedence than addition (`+`) and subtraction (`-`). Precedence may be changed with parenthesis using the same rules as basic arithmetic. Ex: `7 + 3 * 2 = 7 + 6 = 13` because `*` has precedence over `+`, but `(7 + 3) * 2 = 10 * 2 = 20` because `()` is calculated before `*`.

Table 8.3.1: JavaScript arithmetic operators.

Arithmetic operator	Description	Example

`+`	Add	`// x = 3`
		`x = 1 + 2;`

`-`	Subtract	`// x = 1`
		`x = 2 - 1;`

`*`	Multiply	`// x = 6`
		`x = 2 * 3;`

`/`	Divide	`// x = 0.5`
		`x = 1 / 2;`

`%`	Modulus (remainder)	`// x = 0`
		`x = 4 % 2;`

`++`	Increment	`// Same as x = x + 1`
		`x++;`

`--`	Decrement	`// Same as x = x - 1`
		`x--;`

PARTICIPATION ACTIVITY 8.3.1: Arithmetic practice.

1) `3 + 5 * 2 = ?`

Check Show answer

2) `(3 + 4) % 5 = ?`

Check Show answer

3)
```
var points = 10;
points--;
```

What is `points`?

Check Show answer

4) Using the ++ operator, write a statement to add 1 to `points`.

Check Show answer

Exponentiation

*ES7 introduces the **exponentiation operator** `**`, which returns the result of raising the first operand to the power of the second operand. Ex: `2 ** 3` = 2^3 = 8. The exponentiation operator is not yet supported by all browsers.*

A ***compound assignment operator*** combines an assignment statement with an arithmetic operation. A collection of JavaScript compound assignment operators are summarized in the table below.

Table 8.3.2: Compound assignment operators.

Assignment operator	Description	Example

`+=`	Add to	`// Same as x = x + 2`
		`x += 2;`

`-=`	Subtract from	`// Same as x = x - 2`
		`x -= 2;`

`*=`	Multiply by	`// Same as x = x * 3`
		`x *= 3;`

`/=`	Divide by	`// Same as x = x / 3`
		`x /= 3;`

`%=`	Mod by	`// Same as x = x % 4`
		`x %= 4;`

PARTICIPATION ACTIVITY 8.3.2: Practice with compound assignment operators.

1)
```
points = 5;
points ___ 2;
```

What compound assignment operator makes `points` become 2.5?

Check Show answer

2)
```
points = 2;
points *= 3 + 1;
```

What is `points`?

Check Show answer

3)
```
points = 4;
points %= 2;
```

What is `points`?

Check Show answer

The + operator is also the string concatenation operator. ***String concatenation*** appends one string after the end of another string, forming a single string. Ex: `"back" + "pack"` is `"backpack"`. The JavaScript interpreter determines if + means "add" or "concatenate" based on the operands on either side of the operator.

- If both operands are numbers, + performs addition. Ex: 2 + 3 = 5.
- If both operands are strings, + performs string concatenation. Ex: "2" + "3" = "23".
- If one operand is a number and the other a string, + performs string concatenation. The number is converted into a string, and the two strings are concatenated into a single string. Ex: "2" + 3 = "2" + "3" = "23".

For all other arithmetic operators, combining a number and a string in an arithmetic expression converts the string operand to a number and then performs the arithmetic operation. Ex: "2" * 3 = 2 * 3 = 6.

PARTICIPATION ACTIVITY 8.3.3: Type conversion in arithmetic operations.

Animation captions:

1. number + number = number
2. number + string = string
3. number * number = number
4. number * string = number

The JavaScript functions ***parseInt()*** and ***parseFloat()*** convert strings into numbers. Ex: `parseInt("5") + 2 = 5 + 2 = 7`, and `parseFloat("2.4") + 6 = 2.4 + 6 = 8.4`.

If `parseInt()` or `parseFloat()` are given a non-number to parse, the functions return `NaN`. ***NaN*** is a JavaScript value that means Not a Number. Ex: `parseInt("dog")` is `NaN`. The JavaScript function ***isNaN()*** returns `true` if the argument is not a number, `false` otherwise. Ex: `isNaN("dog")` is `true`.

PARTICIPATION ACTIVITY 8.3.4: Arithmetic practice with numbers and strings.

Determine the result of the following expressions. Type "quotes" around strings. If not a number, type NaN.

1) `10 + "ten"`

Check Show answer

2) `"3" / "6"`

Check Show answer

3) `"3" + 5 * 2`

Check Show answer

4) `parseFloat("3.2") + parseInt("2.7")`

Check Show answer

5) `3 + parseInt("pig")`

Check Show answer

6) `2 + isNaN("oink") + isNaN("5")`

Check Show answer

CHALLENGE ACTIVITY 8.3.1: Arithmetic operators.

Start

Write a statement that assigns finalValue with the division of value1 by value2. Ex: If value1 is 6 and value2 is 2, finalValue is 3.

```
var value1 = 6; // Code tested with values: 6 and 4
var value2 = 2; // Code tested with values: 2 and -2

var finalValue = 0;

```

1 2 3 4

Check Next

Exploring further:

- Arithmetic operators from MDN
- MDN documentation for parseInt(), parseFloat(), and isNaN()

8.4 Conditionals

if-else statement

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

Construct 8.4.1: if-else statement.

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

PARTICIPATION ACTIVITY | 8.4.1: Evaluating if-else statements.

Animation captions:

1. Variable x is assigned 6.
2. Evaluate the expression: 6 % 2 = 0.
3. 0 == 0 is true, so block under "if" executes.
4. Variable y is assigned 5.
5. Evaluate the expression: 5 % 2 = 1.
6. 1 == 0 is false, so block under "else" executes.

Relational and equality operators

An if-else statement commonly involves a relational operator or equality operator. Each operator involves two operands and evaluates to a Boolean value meaning either true or false.

Table 8.4.1: Relational and equality operators.

Relational 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`

Relational operator	Name	Example

<	Less than	`2 < 3 // true`
		`"bat" < "zoo" // true`

<=	Less than or equal	`2 <= 3 // true`
		`"bat" <= "bat" // true`

>	Greater than	`3 > 2 // true`
		`"zoo" > "bat" // true`

>=	Greater than or equal	`3 >= 2 // true`
		`"zoo" >= "zoo" // true`

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.

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.

Relational operators also convert a string to a number when comparing a number with a string. 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".

PARTICIPATION ACTIVITY 8.4.2: Evaluating if statements.

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

1)
```
a = 2;
x = a > 10;
```

Check Show answer

2)
```
x = 0;
if (x > 10) {
   x = 1;
}
```

Check Show answer

3)
```
if ("good" > "bad") {
   x = 7;
}
else {
   x = -1;
}
```

Check Show answer

4)
```
if ("charge" <= "chance") {
   x = 2;
}
else {
   x = 4;
}
```

Check Show answer

5)
```
if ("abc" > "ABC") {
   x = 1;
}
else {
   x = 0;
}
```

Check Show answer

6)
```
x = 0;
if ("10" > 5) {
   x = 1;
}
```

Check Show answer

7)
```
a = 2;
if (a == "2") {
   if (a === "2") {
      x = 1;
   }
   else {
      x = 0;
   }
}
```

Check Show answer

PARTICIPATION ACTIVITY 8.4.3: Conditional practice.

Write a series of `if-else` statements 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.

```
var golfScore = 95;

// Write your if-else statements here!

```

Run JavaScript Reset code

Your console output

CHALLENGE ACTIVITY | 8.4.1: Conditionals.

Start

Write an if-else statement that if userTickets is greater than or equal to 5, executes awardPoints = 20. Else, execute awardPoints = userTickets. Ex: If userTickets is 14, then awardPoints = 20.

```
var awardPoints = 0;
var userTickets = 4; // Code will be tested with values: 4, 5 and 6

/* Your solution goes here */

```

1 2 3 4

Check Next

Logical operators

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

Table 8.4.2: 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`

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 parenthesis `()` 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))`.

PARTICIPATION ACTIVITY 8.4.4: Evaluating complex if statements.

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

1)
```
a = 2;
b = 5;
if (a > 10 || b > 0) {
   x = 1;
}
else {
   x = 0;
}
```

Check Show answer

2)

```
a = 2;
b = 5;
if (!(a > 10 && b > 0)) {
   x = 1;
}
else {
   x = 0;
}
```

Check Show answer

3)
```
a = 2;
b = 5;
if (a > 10 || (b != 2 && b > 0)) {
   x = 1;
}
else {
   x = 0;
}
```

Check Show answer

4)
```
a = 2;
b = 5;
if (a % 2 != 0) {
   x = 1;
}
else if (a + b > 10) {
   x = 0;
}
else {
   x = 99;
}
```

Check Show answer

Using { } around if and else blocks

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

```
// Curly braces not required around single statements
if (x > 10)
   x++;
else
   x--;
```

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.

PARTICIPATION ACTIVITY 8.4.5: Truthy and falsy values.

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

1) `if (undefined)`
- ○ true
- ○ false

2) `if (999)`
- ○ true
- ○ false

3) `if (0)`
- ○ true
- ○ false

4) `if ("")`
- ○ true
- ○ false

5) `if (" ")`
- ○ true
- ○ false

6) `if (NaN)`
- ○ true
- ○ false

7) `if (myArray)`
- ○ true
- ○ false

Conditional (ternary) operator

The conditional operator allows developers to write concise conditional statements. The ***conditional operator*** (or ***ternary operator***) has three operands. If the `condition` evaluates to `true`, then the value of `expression1` is returned, otherwise the value of `expression2` is returned.

Construct 8.4.2: Conditional (ternary) operator.

```
condition ? expression1 : expression2
```

PARTICIPATION ACTIVITY 8.4.6: Evaluating the conditional operator.

Animation 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.

PARTICIPATION ACTIVITY 8.4.7: 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

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

```
y = (x > 0) ?
               ;
```

Check Show answer

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

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

Check Show answer

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

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

Check Show answer

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 8.4.3: 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
}
```

PARTICIPATION ACTIVITY 8.4.8: Evaluating the switch statement.

Animation captions:

1. switch statement examines the value variable.
2. value === 1 is false, so the case does not match.
3. value === 5 is false, so the case does not match.
4. value === 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.

PARTICIPATION ACTIVITY 8.4.9: switch statement.

```
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`

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

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

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

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

PARTICIPATION ACTIVITY 8.4.10: Practice with the conditional operator and switch statement.

Convert the first `if-else` statement into an equivalent statement using the conditional operator. Convert the second group of `if-else` statements into an equivalent `switch` statement.

```
var password = "qwerty";

// Convert into equivalent ternary statement
if (password === "opensesame") {
   console.log("Welcome!");
}
else {
   console.log("Invalid password");
}

// Get a number between 0 and 6 representing the day of the week (0 = Sunday,
var 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!");
```

Run JavaScript   Reset code

Your console output

```
```