www.techglobalschool.com

# JavaScript Mock-1 Questions and Answers

## What is JavaScript, and what are its key features?

- JavaScript is a high-level, interpreted programming language primarily used for web development.
- Its key features include being lightweight, easy to learn, and designed to add interactivity to web pages.
- JavaScript is supported by all modern web browsers and can be used on both the client-side (in the browser) and the server-side (with Node.js).

## How is JavaScript different from other programming languages?

- JavaScript is mainly used for web development and has unique features like a loose typing system, dynamic runtime, and closures.
- Unlike some other languages, JavaScript is primarily used to manipulate web page elements, handle events, and perform asynchronous operations.

## Where is JavaScript commonly used in web development?

- JavaScript is used to create interactive and dynamic elements on web pages.
- It's commonly used for form validation, creating interactive menus, implementing sliders, and handling user interactions like button clicks and form submissions.

## What is **ECMAScript**?

- ECMAScript is a standardized scripting language specification on which JavaScript is based. It defines the syntax, types, and semantics of the language.
- JavaScript is the most popular implementation of ECMAScript, and newer versions of JavaScript are often referred to by their corresponding ECMAScript edition (e.g., ES6 for ECMAScript 2015).

## Is JavaScript a **compiled** or **interpreted** language?

- JavaScript is an interpreted language. Unlike compiled languages, JavaScript does not require a separate compilation step.
- The JavaScript code is executed line by line by the JavaScript engine of the browser or runtime environment.

## Is there any relation between **Java** and **JavaScript**?

- Despite the similar names, Java and JavaScript are two entirely different programming languages:
- Java is a statically typed, object-oriented programming language, primarily used for server-side development and standalone applications.
- JavaScript is a dynamically typed, scripting language, primarily used for web development to add interactivity and functionality to web pages.

## What are the differences between **var**, **let**, and **const**?

- They are all used for variable declaration.
- **var** keyword is used in all JavaScript code from 1995 to 2015.
- **let** and **const** keywords were added to JavaScript in 2015.
- Variables defined with **let** can be updated-reassigned with a new value.
- **const** is used if you will not need to change your variable value.
- **const** variables have final values which cannot be changed.

## List and briefly describe the **basic data types** in JavaScript.

- There are 7 primitives: **String**, **Number**, **Bigint**, **Boolean**, **Undefined**, **Null**, **Symbol**
- There are also Reference-Object data types: **arrays**, **objects**, **collections**, etc.
- **String**: A String is a sequence of characters like "John Doe".Strings are written with quotes. You can use single or double quotes but mostly double quotes.
- **Numbers**: All JS numbers are stored as decimal numbers (floating point). Numbers can be written with, or without decimals.
- **BigInt**: All JS numbers are stored in a a 64-bit floating-point format. JavaScript BigInt is a new datatype (2020) that can be used to store integer values that are too big to be represented by a normal JavaScript Number.
- **Boolean**: Boolean can only have two values: true or false. It is used often with conditional statements.
- **Undefined**: In JS, a variable without a value, has the value undefined. The type is also undefined.
- **Array**: JS arrays are used to store multiple data and written with square brackets []. Array items are separated by commas. Array indexes are zero-based.
- **Object**: JS objects are used to store complex data with key-value pairs and written with curly braces {}. Object properties are written as name:value pairs, separated by commas.

## How can you check the **type of** a variable in JavaScript?

- You can use the **typeof** operator to check the type of a variable.

## What are the differences between == and ===?

- **==** performs type coercion, allowing different types to be considered equal. For example, **1 == "1" evaluates to true.**
- **===** performs strict equality comparison, requiring both value and type to be the same. For example, **1 === "1" evaluates to false.**

## How do you convert a string to a number in JavaScript?

- You can use the **parseInt()** or **parseFloat()** functions to convert strings to integers or floating-point numbers.

## What are the differences between **integers** and **floating-point numbers** in JavaScript?

- Integers are whole numbers without fractional parts, while floating-point numbers (also known as "floats") allow for decimal fractions.

## What is an **undefined** property?

- In JavaScript, an undefined property is a property that has not been assigned a value or does not exist in an object.
- When you try to access a property that is not defined, JavaScript will return the value undefined.

## What is **null** value?

- In JavaScript, null is a special value that represents the intentional absence of any object value.
- It is used to explicitly indicate that a variable or object property has no value or is empty.

## What is the difference between **null** and **undefined**?

- The main difference between null and undefined is their meaning:
- null is a value that represents the absence of an object value and is explicitly assigned.
- undefined is a default value given to variables or object properties that have not been initialized or do not exist.

## What are **variable declaration** and **assignment** in JavaScript?

- In JavaScript, a variable is a named container used to store data values.
- Variables in JavaScript are declared using the var, let, or const keywords, followed by the variable name.
- The var keyword was traditionally used to declare variables, but let and const were introduced in newer versions of JavaScript (ES6) and added features like constant variables (immutable).
- Variables can store various data types such as numbers, strings, booleans, objects, arrays, and more.

## What is **isNaN**?

- **isNaN** is a built-in JavaScript function that checks whether a value is "**Not-a-Number**" (**NaN**).
- It returns true if the given value is not a valid number; otherwise, it returns false.

## What are the differences between **undeclared** and **undefined variables**?

- An **undeclared variable** is a variable that has not been declared (created) using the **var**, **let**, or **const** keywords. If you try to access an undeclared variable, JavaScript will throw a **ReferenceError**.
- An **undefined variable** is a variable that has been declared, but no value has been assigned to it. If you access an undefined variable, it will return the value **undefined**.

## What are **identifiers** in JavaScript and what are the rules to follow for identifiers?

- In JavaScript, identifiers are names used to uniquely identify variables, functions, objects, or other entities in the code.
- Identifiers must follow certain rules:
    - They can only start with a letter, underscore (_), or dollar sign ($).
    - Subsequent characters can include letters, digits, underscores, or dollar signs.
    - Identifiers are case-sensitive, meaning **myVariable** and **myvariable** are different identifiers.
- It is good practice to use descriptive and meaningful names for identifiers to improve code readability and maintainability.

## What is **string concatenation** and how do you concatenate two strings in JavaScript?

- String concatenation is forming a new String that is the combination of multiple Strings.
- You can use the **+ operator**, the **concat() method** or **template literals** to concatenate strings.

## What are the differences between **string concatenation** and **interpolation**?

- String concatenation involves joining multiple strings together to create a new string.
- String interpolation, also known as template literals, allows you to embed expressions or variables directly within a string.
- In JavaScript, you can use backticks (`) to create template literals, and you can insert variables or expressions using the ${} syntax.
- String interpolation is a more modern and concise approach for combining strings, especially when you need to include variables or expressions within the string.

- It offers better readability and eliminates the need for explicit string concatenation using the + operator.
- String interpolation is widely used in modern JavaScript applications.

## Can you explain the **concept of escaping characters** in strings?

- Escaping characters allows you to include special characters within a string by using escape sequences.
  - **\n** inserts a new line
  - **\t** inserts a tab space
  - **\"** inserts double quote
  - **\'** inserts single quote
  - **** inserts backslash

## What are the truthy/falsy values in JavaScript?

- **Falsy Values:**
  - These are values that are considered "false" when evaluated in a boolean context. Common examples of falsy values include:
    - The number 0.
    - An empty string ("").
    - null.
    - Undefined.
    - NaN (Not-a-Number).
    - The boolean value false
- **Truthy Values:**
  - These are values that are considered ”true" when evaluated in a boolean context. In most programming languages, values that are not specifically considered "false" are treated as truthy. Common examples of truthy values include:
    - Numbers other than 0.
    - Non-empty strings.
    - Objects and arrays, which are considered true even if they are empty.
    - Functions.
    - The boolean value true

## What is nullish coalescing operator (??)?

- **The nullish coalescing operator (??)** is used to provide a default value when a variable is **null** or **undefined**.
- It's helpful for handling cases where you want to assign a default value only when the variable is explicitly **null** or **undefined**, but not for other falsy values like 0, false, or an empty string.

- The key difference from the logical OR (||) operator is that ?? only falls back to the default value when the variable is null or undefined, not for other falsy values.
- The nullish coalescing operator is particularly useful when dealing with optional properties in objects or when setting default values for function parameters.

## What is the use/purpose of **functions** in JavaScript?

- Functions in JavaScript are reusable blocks of code.
- They allow grouping a set of instructions and executing them whenever needed.
- Functions promote code reusability and maintainability.
- They help break down complex tasks into smaller, more manageable units.
- They can be used multiple times throughout the codebase, reducing duplication.
- Functions accept parameters and return values, enabling dynamic and flexible code execution.
- JavaScript functions enhance code readability and enable efficient problem-solving.

## What are some **string functions** you mostly use in JavaScript?

- Some methods you could mention are listed below:
- **Extracting characters**
    - charAt() - returns the character at a specified index (position) in a string.
    - charCodeAt() - returns the unicode of the character at a specified index in a string.
- **Concatenating & interpolating**
    - concat() – concatenate multiple strings into a new string.
- **Extracting strings**
    - split() – split a string into an array of substrings.
    - substring() – extract a substring from a string.
    - slice() – extract a part of a string.
- **Searching**
    - search() – searches a string for a string (or a regular expression) and returns the position of the match.
    - indexOf() – get the index of the first occurrence of a substring in a string.
    - lastIndexOf() – find the index of the last occurrence of a substring in a string.
    - includes() – check if a string contains a substring.
    - startsWith() – check if a string starts with another string.
    - endsWith() – determine if a string ends with another string.
    - match() - returns an array containing the results of matching a string against a string (or a regular expression).
    - matchAll() - returns an iterator of results after matching a string against a regular expression.
- **Trimming**
    - trim() – remove whitespace characters from a string.
    - trimStart() – remove the leading whitespace characters of a string.

  - trimEnd() – remove the ending whitespace characters of a string.
- **Replacing**
  - replace() – replace a substring in a string with a new one.
  - replaceAll() – replace all occurrences of a substring that matches a pattern with a new one.
- **Changing cases**
  - toUpperCase() – return a string with all characters converted to uppercase.
  - toLowerCase() – return a string with all characters converted to lowercase.
- **Padding**
  - padStart() & padEnd() – pad a string with another string until the result string reaches the given length.

## What are some **number functions** you mostly use in JavaScript?

- Some methods you could mention are listed below:
- **Converting numbers to other data types**
  - toString() - returns a number as a string.
  - toFixed() - returns a number written with a number of decimals.
  - toPrecision() - returns a number written with a specified length.
- **Converting variables to numbers**
  - Number() - returns a number converted from its argument.
  - parseFloat() - parses its argument and returns a floating point number.
  - parseInt() - parses its argument and returns a whole number.
- **Number object functions**
  - Number.isInteger() - returns true if the argument is an integer.
  - Number.isSafeInteger() - returns true if the argument is a safe integer.
  - Number.parseFloat() - converts a string to a number.
  - Number.parseInt() - converts a string to a whole number.

## What are **if-else statements** in JavaScript and what are they used for?

- JavaScript if-else statements are used to test specific conditions and control the flow of the program based on those conditions.
- They execute a block of code if a given condition is true, and another block of code if the condition is false.

## What is the **ternary operator** in JavaScript?

- The ternary operator in JavaScript is a short-hand if-else expression that can be used instead of simple if-else statements.
- It helps to replace multiple lines of code with a single line.
- **Syntax:** variable = (condition) ? expressionTrue : expressionFalse;

## What is the **switch statement** in JavaScript?

- The switch statement in JavaScript is used to select one of many code blocks to be executed based on the value of an expression.
- It can be used with number, string, and enum data types in JavaScript.
- JavaScript allows any number of cases within a switch statement.
- **How it works:**
- 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.
- The break keyword is optional and used to break the switch statement after a case is matched.
- The default keyword is also optional and usually used at the end of the switch statement. It is executed if none of the cases match.
- No break is needed in the default case.

## What is the **break statement** in JavaScript?

- The **break** statement in JavaScript is used for termination.
- It is commonly used with switch statements and loops (for, while, do-while) to exit the current block of code or loop iteration.

## How will the following expression be evaluated? Please provide a detailed explanation of the result.

- (3 == "3" || 2 + 5 === "7") && !(5 < 10 || !false)
- (!true && !false) || 5 * 2 === true * 10
- "3" * "5" >= 15 && true == 1 && "  " == false && "" === 0
- !(5 ** (10 / 5) === 25 || 5 === "5" || 3 >= 5)
- !(2 + 2 === 8 / 2 && !false && 10 % 2 === 0) || 25 % 4 == "1"
- (true && false) || (10 + 5 === "15" && "apple" !== "orange")
- !(10 <= 20 && 3 * 2 === "6") && (5 + 5 !== 11 || false)
- ("Hello" + "World" === "HelloWorld" || 5 > 3) && (true && !false)
- (4 >= 3 && 5 * 2 === "10") || !(7 !== "7" && "dog" === "cat")
- (20 % 6 !== 0 && "apple" === "orange") || !(2 + 2 === 4 && true)
- (!true && !false) && ("car" === "vehicle" || 3 ** 2 <= 10)
- (5 + 3 >= "8" && 2 ** 3 === "8") || !(false || true)
- (10 / 2 === 5 || "hello" + "world" === "helloworld") && (5 === "5" || true)
- ("hello" !== "world" && "apple" + "pie" === "applepie") || (5 + 5 !== "10" && 7 % 2 === 0)
- (5 * 2 === 9 || true) && !(10 - 5 !== 5 && 10 % 3 === 1)

## Find if a number is even or not.

Write a program that generates a random number between 1 and 10 (both inclusive).
If the random number is even, print true.
Otherwise, print false.

Examples:
1 -> false
2 -> true
5 -> false
10 -> true

## Find if a number is odd or not.

Write a program that generates a random number between 1 and 10 (both inclusive).
If the random number is odd, print true.
Otherwise, print false.

Examples:
1 -> true
2 -> false
5 -> true
10 -> false

## Find if a number is positive or not.

Write a program that generates a random number between -5 and 5 (both inclusive).
If the random number is positive, print true.
Otherwise, print false.

Examples:
-5 -> false
-1 -> false
0 -> false
1 -> true
5 -> true

## Find if a number is negative or not.

Write a program that generates a random number between -5 and 5 (both inclusive).
If the random number is negative, print true.
Otherwise, print false.

Examples:
-5 -> true
-1 -> true
0 -> false
1 -> false
5 -> false

## Find if a number is divisible by 5 or not.

Write a program that generates a random number between 1 and 50 (both inclusive).
If the random number is divisible by 5, print true.
Otherwise, print false.

Examples:
1 -> false
5 -> true
20 -> true
37 -> false
50 -> true

## Find if a number is divisible by 7 or not.

Write a program that generates a random number between 1 and 50 (both inclusive).
If the random number is divisible by 7, print true.
Otherwise, print false.

Examples:
1 -> false
7 -> true
35 -> true
49 -> true
50 -> false

## Calculate the sum of 2 random numbers.

Write a program that generates 2 random numbers between 1 and 10 (both inclusive).
Calculate the sum of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5 | -> 8 |
| 7, 3 | -> 10 |
| 5, 5 | -> 10 |
| 1, 10 | -> 11 |
| 10, 1 | -> 11 |

## Calculate the absolute difference between 2 random numbers.

Write a program that generates 2 random numbers between 1 and 10 (both inclusive).
Calculate the absolute difference of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5 | -> 2 |
| 7, 3 | -> 4 |
| 5, 5 | -> 0 |
| 1, 10 | -> 9 |
| 10, 1 | -> 9 |

## Calculate the product of 2 random numbers.

Write a program that generates 2 random numbers between 1 and 10 (both inclusive).
Calculate the product of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5 | -> 15 |
| 7, 3 | -> 21 |
| 5, 5 | -> 25 |
| 1, 10 | -> 10 |
| 10, 1 | -> 10 |

## Calculate the square of a number.

Write a program that generates a random number between 1 and 10 (both inclusive).
Calculate the square of the number and print it.

Examples:
1 -> 1
2 -> 4
5 -> 25
10 -> 100

## Calculate the cube of a number.

Write a program that generates a random number between 1 and 10 (both inclusive).
Calculate the cube of the number and print it.

Examples:
1 -> 1
2 -> 8
5 -> 125
10 -> 1000

## Convert miles to kilometers.

Write a program that generates a random number between 1 and 10 (both inclusive) to be considered as a mile unit.
Convert miles unit to kilometers and print it.
Please assume that 1 mile equals 1.6 kilometers.

Examples:
1 -> 1.6
2 -> 3.2
5 -> 8

## Convert kilograms to pounds.

Write a program that generates a random number between 1 and 100 (both inclusive) to be considered as a kilogram unit.
Convert kilogram unit to pounds and print it.
Please assume that 1 kilogram equals 2.2 pounds.

Examples:
1 -> 2.2
20 -> 44
75 -> 165
100 -> 220

## Find if 2 numbers are equal or not.

Write a program that generates 2 random numbers between 1 and 3 (both inclusive).
If the numbers are equal, print true.
Otherwise, print false.

Examples:
1, 1 -> true
1, 2 -> false
2,3 -> false
2,2 -> true
3, 3 -> false

## Find if an age is allowed to get Driver License or not.

Write a program that generates a random number between 1 and 100 (both inclusive) to be considered as an age.
If the age is more than or equal to 16, print true.
Otherwise, print false.

Examples:
1 -> false
15 -> false
16 -> true
45 -> true
100 -> true

## Find the greatest number between 2 random numbers.

Write a program that generates 2 random numbers between 1 and 10 (both inclusive).
Find the greatest of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5 | -> 5 |
| 7, 3 | -> 7 |
| 5, 5 | -> 5 |
| 1, 10 | -> 10 |
| 10, 1 | -> 10 |

## Find the greatest number between 3 random numbers.

Write a program that generates 3 random numbers between 1 and 10 (both inclusive).
Find the greatest of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5, 2 | -> 5 |
| 7, 3, 1 | -> 7 |
| 5, 5, 5 | -> 5 |
| 1, 10, 9 | -> 10 |
| 10, 1, 2 | -> 10 |

## Find the smallest number between 2 random numbers.

Write a program that generates 2 random numbers between 1 and 10 (both inclusive).
Find the smallest of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5 | -> 3 |
| 7, 3 | -> 3 |
| 5, 5 | -> 5 |
| 1, 10 | -> 1 |
| 10, 1 | -> 1 |

## Find the smallest number between 3 random numbers.

Write a program that generates 3 random numbers between 1 and 10 (both inclusive).
Find the smallest of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5, 2 | -> 2 |
| 7, 3, 1 | -> 1 |
| 5, 5, 5 | -> 5 |
| 1, 10, 9 | -> 1 |
| 10, 1, 2 | -> 1 |

## Calculate the average of 3 random numbers.

Write a program that generates 3 random numbers between 1 and 10 (both inclusive).
Calculate the average of the numbers and print it.

Examples:

| | |
|---|---|
| 3, 5, 7 | -> 5 |
| 7, 3, 2 | -> 4 |
| 5, 5, 5 | -> 5 |
| 1, 10, 7 | -> 6 |
| 10, 1, 1 | -> 4 |

## Calculate the absolute difference between max and min of 3 random numbers.

Write a program that generates 3 random numbers between 1 and 10 (both inclusive).
Calculate the greatest and the smallest numbers and print their absolute difference.

Examples:

| | |
|---|---|
| 3, 5, 2 | -> 3 |
| 7, 3, 1 | -> 6 |
| 5, 5, 5 | -> 0 |
| 1, 10, 9 | -> 9 |
| 10, 1, 2 | -> 9 |

## Find the quarter of a random number between 1 and 100.

Write a program that generates a random number between 1 and 100 (both inclusive).
Find which quarter of the range the number falls into and print it.
1st quarter is 1-25
2nd quarter is 26-50
3rd quarter is 51-75
4th quarter is 76-100

Examples:

| | |
|---|---|
| 15 | -> 1st quarter |
| 73 | -> 3rd quarter |
| 39 | -> 2nd quarter |
| 87 | -> 4th quarter |

## Find the midpoint of a random number between 1 and 100.

Write a program that generates a random number between 1 and 100 (both inclusive).
Find which half of the range the number falls into and print it.
1st half is 1-50
2nd half is 51-100

Examples:

| | |
|---|---|
| 15 | -> 1st half |
| 50 | -> 1st half |
| 51 | -> 2nd half |
| 87 | -> 2nd half |
| 100 | -> 2nd half |

## Find if sum of 2 random numbers is even or not.

Write a program that generates 2 random numbers between 1 and 10 (both inclusive).
If the sum of the random numbers is even, print true.
Otherwise, print false.

Examples:

| | |
|---|---|
| 3, 5 | -> true |
| 7, 3 | -> true |
| 5, 5 | -> true |
| 1, 10 | -> false |
| 10, 1 | -> false |

## Find if product of 2 random numbers is odd or not.

Write a program that generates 2 random numbers between 1 and 10 (both inclusive).
If the product of the random numbers is odd, print true.
Otherwise, print false.

Examples:

| | |
|---|---|
| 3, 5 | -> true |
| 7, 3 | -> true |
| 5, 5 | -> true |
| 1, 10 | -> false |
| 10, 1 | -> false |

## Area of a rectangle

Write a function named as rectangleArea()which calculates the area of a rectangle when invoked.
NOTE: Assume the sides of the rectangle are x and y.
Conversion Formula: Area = x * y

Examples:

| | |
|---|---|
| rectangleArea(5, 4) | -> 20 |
| rectangleArea(3, 7) | -> 21 |
| rectangleArea(6, 10) | -> 60 |

## Perimeter of a rectangle.

Write a function named as rectanglePerimeter() which calculates the perimeter of a rectangle when invoked.
NOTE: Assume the sides of the rectangle are x and y.
Conversion Formula: Perimeter = 2 * (x + y)

Examples:

| | |
|---|---|
| rectanglePerimeter(5, 4) | -> 18 |
| rectanglePerimeter(3, 7) | -> 20 |
| rectanglePerimeter(6, 10) | -> 32 |

## Area of a square

Write a function named as squareArea() which calculates the area of a square when invoked.
NOTE: Assume the side of the square is x.
Conversion Formula:Area = x * x

Examples:
squareArea(5) -> 25
squareArea(3) -> 9
squareArea(6) -> 36

## Perimeter of a square.

Write a function named as squarePerimeter() which calculates the perimeter of a square when invoked.
NOTE: Assume the side of the square is x.
Conversion Formula: Perimeter = 4 * x

Examples:
squarePerimeter(5) -> 20
squarePerimeter(3) -> 12
squarePerimeter(6) -> 24

## Double The Word

Write a function named as doubleWord() which takes a string word as an argument and returns the given word back doubled when invoked.
NOTE: Assume you will not be given an empty word.

Examples:
doubleWord("Tech") -> "TechTech"
doubleWord("Global") -> "GlobalGlobal"

## First Character

Write a function named as firstCharacter() which takes a string word as an argument and returns the first character of the given word when invoked.
NOTE: Assume you will not be given an empty word.

Examples:
firstCharacter("Tech") -> "T"
firstCharacter("Global") -> "G"

## First Two Characters

Write a function named as firstTwoCharacters() which takes a string word as an argument and returns the first two characters of the given word when invoked.
NOTE: If the given word does not have 2 or more characters, then return the given string back.

Examples:
firstTwoCharacters("Tech") -> "Te"
firstTwoCharacters("Global") -> "Gl"
firstTwoCharacters("") -> ""
firstTwoCharacters(" ") -> " "
firstTwoCharacters("1") -> "1"

## Last Character

Write a function named as lastCharacter() which takes a string word as an argument and returns the last character of the given word when invoked.
NOTE: Assume you will not be given an empty word.

Examples:
lastCharacter("Tech") -> "h"
lastCharacter("Global") -> "l"
lastCharacter(" ") -> " "
lastCharacter("123") -> "3"

## Last Two Characters

Write a function named as lastTwoCharacters() which takes a string word as an argument and returns the last two characters of the given word when invoked.
NOTE: If the given word does not have 2 or more characters, then return the string back.

Examples:
lastTwoCharacters("Tech") -> "ch"
lastTwoCharacters("Global") -> "al"
lastTwoCharacters("") -> ""
lastTwoCharacters(" ") -> " "
lastTwoCharacters("1") -> "1"

## First and Last Characters

Write a function named as firstLast() which takes a string word as an argument and returns the first and the last characters of the given word when invoked.
NOTE: If the given word does not have 2 or more characters, then return the string back.

Examples:
firstLast("Tech") -> "Th"
firstLast("TechGlobal") -> "Tl"
firstLast("") -> ""
firstLast(" ") -> " "
firstLast("1") -> "1"
firstLast("abcde") -> ae

## Has Five

Write a function named as hasFive() which takes a string word as an argument and returns true if given string has at least 5 characters, and false otherwise when invoked.

Examples:
hasFive("Tech") -> false
hasFive("Global") -> true
hasFive("") -> false
hasFive("12345") -> true
hasFive("hello") -> true

## Middle

Write a function named as middle() which takes a string word as an argument and returns the middle character if the string has odd length, and returns the middle two characters if the string has even length when invoked.
NOTE: If the given word is empty, then return the empty string back.

Examples:
middle("Tech") -> "ec"
middle("Global") -> "ob"
middle("abcde") -> "c"
middle("1") -> "1"
middle("abc") -> "b"
middle(“1234”) -> "23"

## Longer String

Write a function named as longer() which takes two string words as arguments and returns the string that has more characters when invoked.
NOTE: If both of the words have the same length, then return the first string.

Examples:
longer("Tech", "Global") -> "Global"
longer("Hello", "Hi") -> "Hello"
longer("Hello", "World") -> "Hello"

## Shorter String

Write a function named as shorter() which takes two String words as arguments and returns the String has less characters when invoked.
NOTE: if both of the words have the same length, then return the second String.

Examples:
shorter("Tech", "Global") -> "Tech"
shorter("Hello", "Hi") -> "Hi"
shorter("Hello", "World") -> "World"

## Concat Two String

Write a function named as concat() which takes two string words as arguments and returns the words concatenated when invoked.
NOTE: Concatenation should always be as first string + second string .

Examples:
concat("Tech", "Global") -> "TechGlobal"
concat("Hello", "World") -> "HelloWorld"
concat("", "abc") -> "abc"
concat("123", "1234") -> "1231234"

## Starts With Vowel

Write a function named as startsVowel() which takes a string word as an argument and returns true if given string starts with a vowel letter, and false otherwise when invoked.
NOTE: Vowel letters are A, E, O, U, I, a, e, o, u, i.

Examples:
startsVowel("Tech") -> false
startsVowel("Apple") -> true
startsVowel("abc") -> true