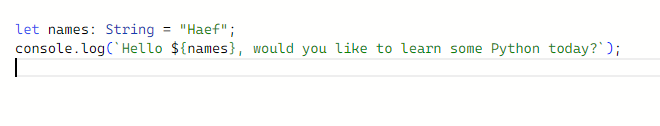
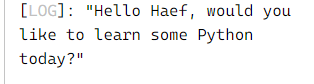
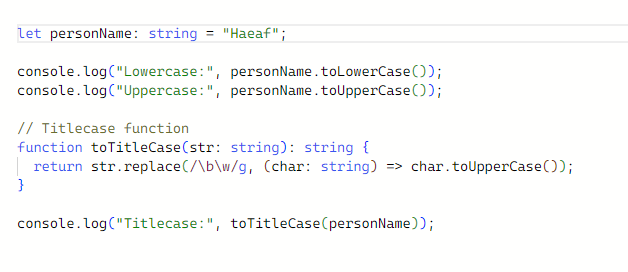
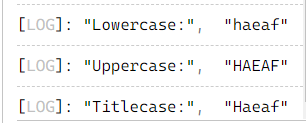
1. Personal Message: Store a person’s name in a variable, and print a message to that person. Your message should be simple, such as, “Hello Eric, would you like to learn some Python today?”





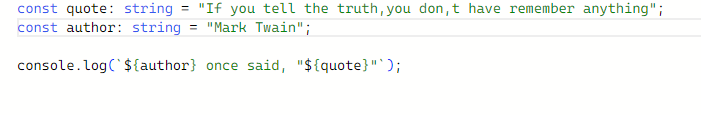
1. Name Cases: Store a person’s name in a variable, and then print that person’s name in lowercase, uppercase, and titlecase.

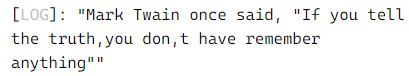




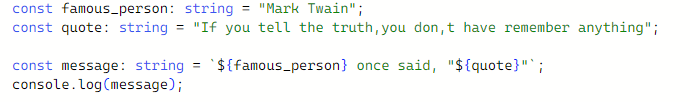
1. Famous Quote: Find a quote from a famous person you admire. Print the quote and the name of its author. Your output should look something like the following, including the quotation marks:

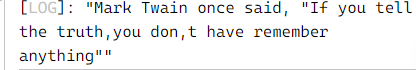
Albert Einstein once said, “A person who never made a mistake never tried anything new.”



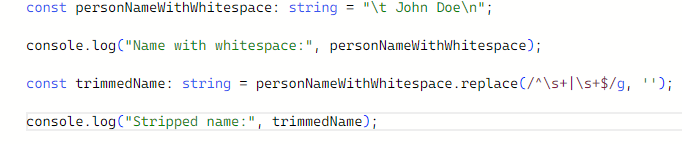


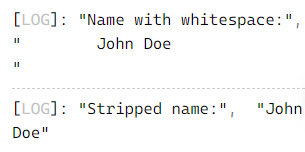
1. Famous Quote 2: Repeat Exercise 4, but this time store the famous person’s name in a variable called famous\_person. Then compose your message and store it in a new variable called message. Print your message.

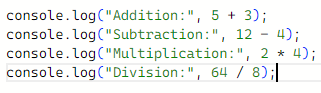


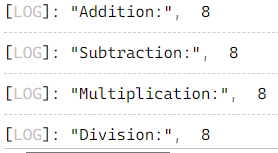


1. Stripping Names: Store a person’s name, and include some whitespace characters at the beginning and end of the name. Make sure you use each character combination, "\t" and "\n", at least once. Print the name once, so the whitespace around the name is displayed. Then print the name after striping the white spaces.





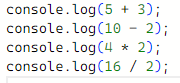
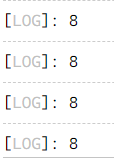




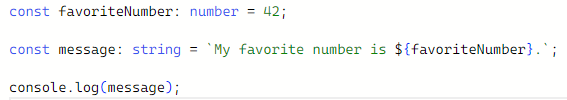
1. You should create four lines that look like this:

console.log(5 + 3)

Your output should simply be four lines with the number 8 appearing once on each line.

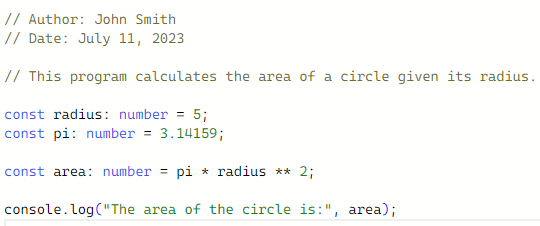


1. Favorite Number: Store your favorite number in a variable. Then, using that variable, create a message that reveals your favorite number. Print that message.



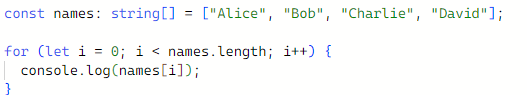


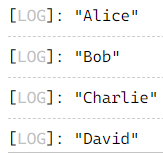
1. Adding Comments: Choose two of the programs you’ve written, and add at least one comment to each. If you don’t have anything specific to write because your programs are too simple at this point, just add your name and the current date at the top of each program file. Then write one sentence describing what the program does.



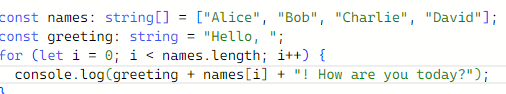


1. Names: Store the names of a few of your friends in a array called names. Print each person’s name by accessing each element in the list, one at a time



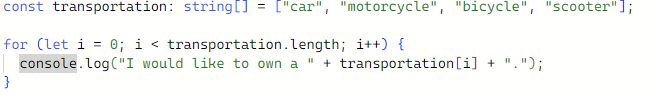
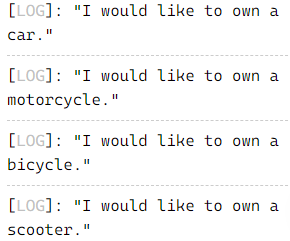


1. Greetings: Start with the array you used in Exercise 11, but instead of just printing each person’s name, print a message to them. The text of each message should be the same, but each message should be personalized with the person’s name.

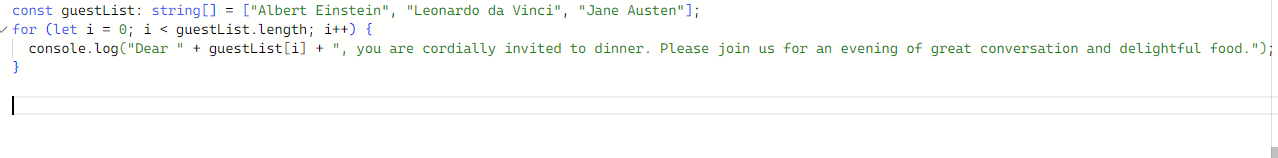


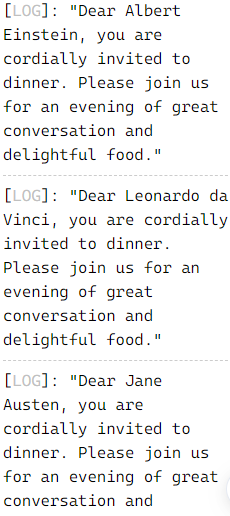


1. Your Own Array: Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples. Use your list to print a series of statements about these items, such as “I would like to own a Honda motorcycle.”



1. Guest List: If you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you’d like to invite to dinner. Then use your list to print a message to each person, inviting them to dinner.

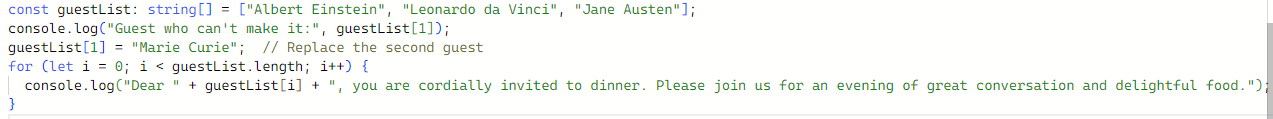


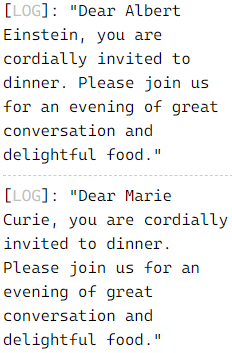


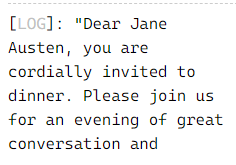
1. Changing Guest List: You just heard that one of your guests can’t make the dinner, so you need to send out a new set of invitations. You’ll have to think of someone else to invite.

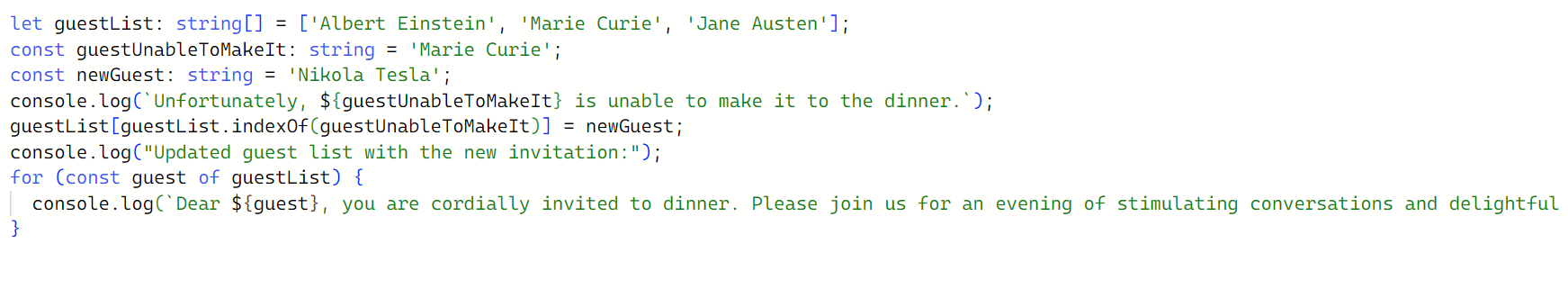
• Start with your program from Exercise 14. Add a print statement at the end of your program stating the name of the guest who can’t make it.

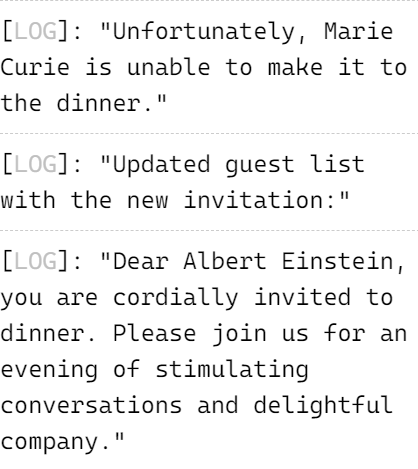
• Modify your list, replacing the name of the guest who can’t make it with the name of the new person you are inviting.

• Print a second set of invitation messages, one for each person who is still in your list







1. More Guests: You just found a bigger dinner table, so now more space is available. Think of three more guests to invite to dinner.

• Start with your program from Exercise 15. Add a print statement to the end of your program informing people that you found a bigger dinner table.

• Add one new guest to the beginning of your array.

• Add one new guest to the middle of your array. • Use append() to add one new guest to the end of your list. • Print a new set of invitation messages, one for each person in your list.

let guestList: string[] = ['Albert Einstein', 'Marie Curie', 'Leonardo da Vinci'];

const guestUnableToMakeIt: string = 'Marie Curie';

const newGuest: string = 'Nikola Tesla';

console.log(`Unfortunately, ${guestUnableToMakeIt} is unable to make it to the dinner.`);

guestList[guestList.indexOf(guestUnableToMakeIt)] = newGuest;

console.log("Updated guest list with the new invitation:");

for (const guest of guestList) {

  console.log(`Dear ${guest}, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company.`);

}

console.log("Good news! We found a bigger dinner table.");

const newGuestBeginning: string = 'Isaac Newton';  // New guest added to the beginning of the list

guestList.unshift(newGuestBeginning);

const middleIndex: number = Math.floor(guestList.length / 2);

const newGuestMiddle: string = 'Ada Lovelace';  // New guest added to the middle of the list

guestList.splice(middleIndex, 0, newGuestMiddle);

const newGuestEnd: string = 'Marie Curie 2.0';  // New guest added to the end of the list

guestList.push(newGuestEnd);

console.log("Updated guest list with additional invitations:");

for (const guest of guestList) {

  console.log(`Dear ${guest}, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company.`)

[LOG]: "Unfortunately, Marie Curie is unable to make it to the dinner."

[LOG]: "Updated guest list with the new invitation:"

[LOG]: "Dear Albert Einstein, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Nikola Tesla, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Leonardo da Vinci, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Good news! We found a bigger dinner table."

[LOG]: "Updated guest list with additional invitations:"

[LOG]: "Dear Isaac Newton, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Albert Einstein, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Ada Lovelace, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Nikola Tesla, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Leonardo da Vinci, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Marie Curie 2.0, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

1. Shrinking Guest List: You just found out that your new dinner table won’t arrive in time for the dinner, and you have space for only two guests.
2. let guestList: string[] = ['Albert Einstein', 'Marie Curie', 'Leonardo da Vinci'];
3. const guestUnableToMakeIt: string = 'Marie Curie';
4. const newGuest: string = 'Nikola Tesla';
5. console.log(`Unfortunately, ${guestUnableToMakeIt} is unable to make it to the dinner.`);
6. guestList[guestList.indexOf(guestUnableToMakeIt)] = newGuest;
7. console.log("Updated guest list with the new invitation:");
8. for (const guest of guestList) {
9. console.log(`Dear ${guest}, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company.`);
10. }
11. console.log("Good news! We found a bigger dinner table.");
12. const newGuestBeginning: string = 'Isaac Newton';  // New guest added to the beginning of the list
13. guestList.unshift(newGuestBeginning);
14. const middleIndex: number = Math.floor(guestList.length / 2);
15. const newGuestMiddle: string = 'Ada Lovelace';  // New guest added to the middle of the list
16. guestList.splice(middleIndex, 0, newGuestMiddle);
17. const newGuestEnd: string = 'Marie Curie 2.0';  // New guest added to the end of the list
18. guestList.push(newGuestEnd);
19. console.log("Updated guest list with additional invitations:");
20. for (const guest of guestList) {
21. console.log(`Dear ${guest}, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company.`);
22. }
23. console.log("Unfortunately, the new dinner table won't arrive in time. We can accommodate only two guests.");
24. const removedGuests = guestList.splice(2);  // Remove guests beyond index 1
25. for (const removedGuest of removedGuests) {
26. console.log(`Sorry, ${removedGuest}, we have limited space and cannot accommodate you for dinner.`);
27. }
28. console.log("Updated guest list with two remaining guests:");
29. for (const guest of guestList) {
30. console.log(`Dear ${guest}, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company.`);
31. }

[LOG]: "Unfortunately, Marie Curie is unable to make it to the dinner."

[LOG]: "Updated guest list with the new invitation:"

[LOG]: "Dear Albert Einstein, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Nikola Tesla, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Leonardo da Vinci, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Good news! We found a bigger dinner table."

[LOG]: "Updated guest list with additional invitations:"

[LOG]: "Dear Isaac Newton, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Albert Einstein, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Ada Lovelace, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Nikola Tesla, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Leonardo da Vinci, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Marie Curie 2.0, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Unfortunately, the new dinner table won't arrive in time. We can accommodate only two guests."

[LOG]: "Sorry, Ada Lovelace, we have limited space and cannot accommodate you for dinner."

[LOG]: "Sorry, Nikola Tesla, we have limited space and cannot accommodate you for dinner."

[LOG]: "Sorry, Leonardo da Vinci, we have limited space and cannot accommodate you for dinner."

[LOG]: "Sorry, Marie Curie 2.0, we have limited space and cannot accommodate you for dinner."

[LOG]: "Updated guest list with two remaining guests:"

[LOG]: "Dear Isaac Newton, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

[LOG]: "Dear Albert Einstein, you are cordially invited to dinner. Please join us for an evening of stimulating conversations and delightful company."

1. Seeing the World: Think of at least five places in the world you’d like to visit.

• Store the locations in a array. Make sure the array is not in alphabetical order.

• Print your array in its original order.

• Print your array in alphabetical order without modifying the actual list.

• Show that your array is still in its original order by printing it.

• Print your array in reverse alphabetical order without changing the order of the original list.

• Show that your array is still in its original order by printing it again.

• Reverse the order of your list. Print the array to show that its order has changed.

• Reverse the order of your list again. Print the list to show it’s back to its original order.

• Sort your array so it’s stored in alphabetical order. Print the array to show that its order has been changed.

• Sort to change your array so it’s stored in reverse alphabetical order. Print the list to show that its order has changed.

/ Declare the array of locations

let places: string[] = ["Tokyo", "Paris", "Machu Picchu", "Sydney", "Cairo"];

// Print the array in its original order

console.log("Original Order:");

console.log(places);

// Print the array in alphabetical order without modifying the actual list

console.log("\nAlphabetical Order:");

console.log([...places].sort());

// Print the array to show it's still in its original order

console.log("\nOriginal Order (unchanged):");

console.log(places);

// Print the array in reverse alphabetical order without changing the order of the original list

console.log("\nReverse Alphabetical Order:");

console.log([...places].sort().reverse());

// Print the array to show it's still in its original order

console.log("\nOriginal Order (unchanged):");

console.log(places);

// Reverse the order of the list

places.reverse();

// Print the array to show its order has changed

console.log("\nReversed Order:");

console.log(places);

// Reverse the order of the list again

places.reverse();

// Print the array to show it's back to its original order

console.log("\nOriginal Order (restored):");

console.log(places);

// Sort the array in alphabetical order

places.sort();

// Print the array to show its order has been changed

console.log("\nAlphabetical Order:");

console.log(places);

// Sort the array in reverse alphabetical order

places.sort((a, b) => b.localeCompare(a));

// Print the array to show its order has changed

console.log("\nReverse Alphabetical Order:");

console.log(places);

[LOG]: "Original Order:"

[LOG]: ["Tokyo", "Paris", "Machu Picchu", "Sydney", "Cairo"]

[LOG]: " Alphabetical Order:"

[LOG]: ["Cairo", "Machu Picchu", "Paris", "Sydney", "Tokyo"]

[LOG]: " Original Order (unchanged):"

[LOG]: ["Tokyo", "Paris", "Machu Picchu", "Sydney", "Cairo"]

[LOG]: " Reverse Alphabetical Order:"

[LOG]: ["Tokyo", "Sydney", "Paris", "Machu Picchu", "Cairo"]

[LOG]: " Original Order (unchanged):"

[LOG]: ["Tokyo", "Paris", "Machu Picchu", "Sydney", "Cairo"]

[LOG]: " Reversed Order:"

[LOG]: ["Cairo", "Sydney", "Machu Picchu", "Paris", "Tokyo"]

[LOG]: " Original Order (restored):"

[LOG]: ["Tokyo", "Paris", "Machu Picchu", "Sydney", "Cairo"]

[LOG]: " Alphabetical Order:"

[LOG]: ["Cairo", "Machu Picchu", "Paris", "Sydney", "Tokyo"]

[LOG]: " Reverse Alphabetical Order:"

[LOG]: ["Tokyo", "Sydney", "Paris", "Machu Picchu", "Cairo"]

1. Dinner Guests: Working with one of the programs from Exercises 14 through 18, print a message indicating the number of people you are inviting to dinner.

// Example program modified from Exercise 14

let guests: string[] = ["John", "Jane", "Mark", "Emily", "Mike"];

// Print the number of guests

console.log(`Number of guests invited to dinner: ${guests.length}`);

***t***

[LOG]: "Number of guests invited to dinner: 5"

1. .Think of something you could store in a array. For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you’d like. Write a program that creates a list containing these items.

// Create an array of mountains

let mountains: string[] = ["Mount Everest", "K2", "Kangchenjunga", "Matterhorn", "Mount Kilimanjaro"];

// Print the array

console.log(mountains);

[LOG]: ["Mount Everest", "K2", "Kangchenjunga", "Matterhorn", "Mount Kilimanjaro"]

1. They think of something you could store in a TypeScript Object. Write a program that creates Objects containing these items.

// Create an object for storing country information

let constcountry: { name: string, capital: string, population: number, language: string } = {

  name: "United States",

  capital: "Washington, D.C.",

  population: 331002651,

  language: "English"

};

[LOG]: { "name": "United States", "capital": "Washington, D.C.", "population": 331002651, "language": "English" }

1. Intentional Error: If you haven’t received an array index error in one of your programs yet, try to make one happen. Change an index in one of your programs to produce an index error. Make sure you correct the error before closing the program.

// Create an array

let numbers: number[] = [1, 2, 3, 4, 5];

// Try to access an index that is out of bounds

console.log(numbers[10]);

[LOG]: undefined

// Create an array

let numbers: number[] = [1, 2, 3, 4, 5];

// Try to access an index that is out of bounds

console.log(numbers[4]);

[LOG]: 5

1. Conditional Tests: Write a series of conditional tests. Print a statement describing each test and your prediction for the results of each test. Your code should look something like this:

let car = 'subaru';

console.log("Is car == 'subaru'? I predict True.")

console.log(car == 'subaru')

• Look closely at your results, and make sure you understand why each line evaluates to True or False.

• Create at least 10 tests. Have at least 5 tests evaluate to True and another 5 tests evaluate to False.

let car: string = 'subaru';

console.log("Is car == 'subaru'? I predict true.");

console.log(car == 'subaru');

console.log("Is car == 'honda'? I predict false.");

console.log(car == 'honda');

console.log("Is car === 'Subaru'? I predict false.");

console.log(car === 'Subaru');

console.log("Is car != 'ford'? I predict true.");

console.log(car != 'ford');

console.log("Is car !== 'subaru'? I predict false.");

console.log(car !== 'subaru');

console.log("Is car.length > 4? I predict true.");

console.log(car.length > 4);

console.log("Is car.length < 6? I predict true.");

console.log(car.length < 6);

[LOG]: "Is car == 'subaru'? I predict true."

[LOG]: true

[LOG]: "Is car == 'honda'? I predict false."

[LOG]: false

[LOG]: "Is car === 'Subaru'? I predict false."

[LOG]: false

[LOG]: "Is car != 'ford'? I predict true."

[LOG]: true

[LOG]: "Is car !== 'subaru'? I predict false."

[LOG]: false

[LOG]: "Is car.length > 4? I predict true."

[LOG]: true

[LOG]: "Is car.length < 6? I predict true."

[LOG]: false

[LOG]: "Is car.length >= 6? I predict false."

[LOG]: true

[LOG]: "Is car.length <= 5? I predict true."

[LOG]: false

[LOG]: "Is car.startsWith('sub')? I predict true."

[LOG]: true

[LOG]: "Is car.endsWith('ru')? I predict true."

[LOG]: true

1. More Conditional Tests: You don’t have to limit the number of tests you create to 10. If you want to try more comparisons, write more tests. Have at least one True and one False result for each of the following:

• Tests for equality and inequality with strings

• Tests using the lower case function

• Numerical tests involving equality and inequality, greater than and less than, greater than or equal to, and less than or equal to

• Tests using "and" and "or" operators

• Test whether an item is in a array

• Test whether an item is not in a array

let string1: string = 'hello';

let string2: string = 'world';

let number1: number = 10;

let number2: number = 5;

let array: number[] = [1, 2, 3, 4, 5];

console.log("Test for equality and inequality with strings:");

console.log("Is string1 == string2? I predict false.");

console.log(string1 == string2);

console.log("Is string1 != string2? I predict true.");

console.log(string1 != string2);

console.log("\nTest using the lowercase function:");

console.log("Is string1.toLowerCase() == 'hello'? I predict true.");

console.log(string1.toLowerCase() == 'hello');

console.log("Is string2.toLowerCase() == 'Hello'? I predict false.");

console.log(string2.toLowerCase() == 'Hello');

console.log("\nNumerical tests:");

console.log("Is number1 > number2? I predict true.");

console.log(number1 > number2);

console.log("Is number1 < number2? I predict false.");

console.log(number1 < number2);

console.log("Is number1 >= number2? I predict true.");

console.log(number1 >= number2);

console.log("Is number1 <= number2? I predict false.");

console.log(number1 <= number2);

console.log("\nTests using 'and' and 'or' operators:");

console.log("Is number1 > 0 and number2 < 0? I predict false.");

console.log(number1 > 0 && number2 < 0);

console.log("Is number1 > 0 or number2 < 0? I predict true.");

console.log(number1 > 0 || number2 < 0);

console.log("\nTest whether an item is in an array:");

console.log("Is 3 in array? I predict true.");

console.log(array.includes(3));

console.log("Is 6 in array? I predict false.");

console.log(array.includes(6));

console.log("\nTest whether an item is not in an array:");

console.log("Is 4 not in array? I predict false.");

console.log(!array.includes(4));

console.log("Is 7 not in array? I predict true.");

console.log(!array.includes(7));

[LOG]: "Is car.endsWith('ru')? I predict true."

[LOG]: true

[LOG]: "Test for equality and inequality with strings:"

[LOG]: "Is string1 == string2? I predict false."

[LOG]: false

[LOG]: "Is string1 != string2? I predict true."

[LOG]: true

[LOG]: " Test using the lowercase function:"

[LOG]: "Is string1.toLowerCase() == 'hello'? I predict true."

[LOG]: true

[LOG]: "Is string2.toLowerCase() == 'Hello'? I predict false."

[LOG]: false

[LOG]: " Numerical tests:"

[LOG]: "Is number1 > number2? I predict true."

[LOG]: true

[LOG]: "Is number1 < number2? I predict false."

[LOG]: false

[LOG]: "Is number1 >= number2? I predict true."

[LOG]: true

[LOG]: "Is number1 <= number2? I predict false."

[LOG]: false

[LOG]: " Tests using 'and' and 'or' operators:"

[LOG]: "Is number1 > 0 and number2 < 0? I predict false."

[LOG]: false

[LOG]: "Is number1 > 0 or number2 < 0? I predict true."

[LOG]: true

[LOG]: " Test whether an item is in an array:"

[LOG]: "Is 3 in array? I predict true."

[LOG]: true

[LOG]: "Is 6 in array? I predict false."

[LOG]: false

[LOG]: " Test whether an item is not in an array:"

[LOG]: "Is 4 not in array? I predict false."

[LOG]: false

[LOG]: "Is 7 not in array? I predict true."

[LOG]: true

1. Alien Colors #1: Imagine an alien was just shot down in a game. Create a variable called alien\_color and assign it a value of 'green', 'yellow', or 'red'.

• Write an if statement to test whether the alien’s color is green. If it is, print a message that the player just earned 5 points.

• Write one version of this program that passes the if test and another that fails. (The version that fails will have no output.)

let alien\_color: string = 'green';

// Version that passes the if test

if (alien\_color === 'green') {

  console.log("The player just earned 5 points!");

}

// Version that fails the if test

if (alien\_color === 'red') {

  console.log("The player just earned 5 points!");

}

[LOG]: "The player just earned 5 points!"

1. Alien Colors #2: Choose a color for an alien as you did in Exercise 25, and write an if-else chain.

• If the alien’s color is green, print a statement that the player just earned 5 points for shooting the alien.

• If the alien’s color isn’t green, print a statement that the player just earned 10 points.

• Write one version of this program that runs the if block and another that runs the else block.

let alien\_color: string = 'green';

// Version that passes the if test

if (alien\_color === 'green') {

  console.log("The player just earned 5 points!");

}

// Version that fails the if test

if (alien\_color === 'red') {

  console.log("The player just earned 5 points!");

}

// Version that runs the if block

if (alien\_color === 'green') {

  console.log("The player just earned 5 points for shooting the alien!");

} else {

  console.log("The player just earned 10 points!");

}

// Version that runs the else block

alien\_color = 'yellow';

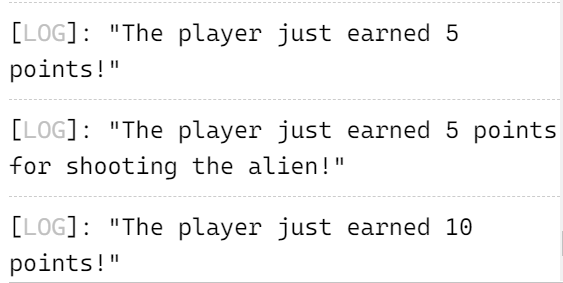
if (alien\_color === 'green') {

  console.log("The player just earned 5 points for shooting the alien!");

} else {

  console.log("The player just earned 10 points!");

}



1. Alien Colors #3: Turn your if-else chain from Exercise 5-4 into an if-else chain.

• If the alien is green, print a message that the player earned 5 points.

• If the alien is yellow, print a message that the player earned 10 points.

• If the alien is red, print a message that the player earned 15 points.

• Write three versions of this program, making sure each message is printed for the appropriate color alien.

let alien\_color: string = 'green';

// Version that passes the if test

if (alien\_color === 'green') {

  console.log("The player just earned 5 points!");

}

// Version that fails the if test

if (alien\_color === 'red') {

  console.log("The player just earned 5 points!");

}

// Version that runs the if block

if (alien\_color === 'green') {

  console.log("The player just earned 5 points for shooting the alien!");

} else {

  console.log("The player just earned 10 points!");

}

// Version that runs the else block

alien\_color = 'yellow';

if (alien\_color === 'green') {

  console.log("The player just earned 5 points for shooting the alien!");

} else {

  console.log("The player just earned 10 points!");

}

const : string = 'green';

// Version for green alien

if (alien\_color === 'green') {

  console.log("The player earned 5 points!");

} else if (alien\_color === 'yellow') {

  console.log("The player earned 10 points!");

} else if (alien\_color === 'red') {

  console.log("The player earned 15 points!");

}

alien\_color = 'yellow';

// Version for yellow alien

if (alien\_color === 'green') {

  console.log("The player earned 5 points!");

} else if (alien\_color === 'yellow') {

  console.log("The player earned 10 points!");

} else if (alien\_color === 'red') {

  console.log("The player earned 15 points!");

}

alien\_color = 'red';

// Version for red alien

if (alien\_color === 'green') {

  console.log("The player earned 5 points!");

} else if (alien\_color === 'yellow') {

  console.log("The player earned 10 points!");

} else if (alien\_color === 'red') {

  console.log("The player earned 15 points!");

}

[LOG]: "The player earned 10 points!"

[LOG]: "The player earned 10 points!"

[LOG]: "The player earned 15 points!"

1. Stages of Life: Write an if-else chain that determines a person’s stage of life. Set a value for the variable age, and then:

• If the person is less than 2 years old, print a message that the person is a baby.

• If the person is at least 2 years old but less than 4, print a message that the person is a toddler.

• If the person is at least 4 years old but less than 13, print a message that the person is a kid.

• If the person is at least 13 years old but less than 20, print a message that the person is a teenager.

• If the person is at least 20 years old but less than 65, print a message that the person is an adult.

• If the person is age 65 or older, print a message that the person is an elder.

let age: number = 25;

if (age < 2) {

  console.log("The person is a baby.");

} else if (age >= 2 && age < 4) {

  console.log("The person is a toddler.");

} else if (age >= 4 && age < 13) {

  console.log("The person is a kid.");

} else if (age >= 13 && age < 20) {

  console.log("The person is a teenager.");

} else if (age >= 20 && age < 65) {

  console.log("The person is an adult.");

} else {

  console.log("The person is an elder.");

}

[LOG]: "The person is an adult."

1. Favorite Fruit: Make a array of your favorite fruits, and then write a series of independent if statements that check for certain fruits in your array.

• Make a array of your three favorite fruits and call it favorite\_fruits.

• Write five if statements. Each should check whether a certain kind of fruit is in your array. If the fruit is in your array, the if block should print a statement, such as You really like bananas!

let favorite\_fruits: string[] = ["banana", "apple", "mango"];

if (favorite\_fruits.includes("banana")) {

  console.log("You really like bananas!");

}

if (favorite\_fruits.includes("apple")) {

  console.log("You really like apples!");

}

if (favorite\_fruits.includes("mango")) {

  console.log("You really like mangoes!");

}

if (favorite\_fruits.includes("orange")) {

  console.log("You really like oranges!");

}

if (favorite\_fruits.includes("strawberry")) {

  console.log("You really like strawberries!");

}

[LOG]: "You really like bananas!"

[LOG]: "You really like apples!"

[LOG]: "You really like mangoes!"

1. Hello Admin: Make a array of five or more usernames, including the name 'admin'. Imagine you are writing code that will print a greeting to each user after they log in to a website. Loop through the array, and print a greeting to each user:

• If the username is 'admin', print a special greeting, such as Hello admin, would you like to see a status report?

• Otherwise, print a generic greeting, such as Hello Eric, thank you for logging in again.

let usernames: string[] = ["admin", "Eric", "John", "Alice", "Mike"];

for (let i = 0; i < usernames.length; i++) {

  if (usernames[i] === "admin") {

    console.log("Hello admin, would you like to see a status report?");

  } else {

    console.log(`Hello ${usernames[i]}, thank you for logging in again.`);

  }

}

[LOG]: "Hello admin, would you like to see a status report?"

[LOG]: "Hello Eric, thank you for logging in again."

[LOG]: "Hello John, thank you for logging in again."

[LOG]: "Hello Alice, thank you for logging in again."

[LOG]: "Hello Mike, thank you for logging in again."

1. No Users: Add an if test to Exercise 28 to make sure the list of users is not empty.

• If the list is empty, print the message We need to find some users!

• Remove all of the usernames from your array, and make sure the correct message is printed.

let users: string[] = []; // Empty array of users

// Check if the list of users is empty

if (users.length === 0) {

  console.log("We need to find some users!");

} else {

  // Remove all usernames from the array

  users = [];

  console.log("All usernames have been removed.");

}

[LOG]: "We need to find some users!"

1. Checking Usernames: Do the following to create a program that simulates how websites ensure that everyone has a unique username.

• Make a list of five or more usernames called current\_users.

• Make another list of five usernames called new\_users. Make sure one or two of the new usernames are also in the current\_users list.

• Loop through the new\_users list to see if each new username has already been used. If it has, print a message that the person will need to enter a new username. If a username has not been used, print a message saying that the username is available.

• Make sure your comparison is case insensitive. If 'John' has been used, 'JOHN' should not be accepted.

let current\_users: string[] = ["John", "Jane", "Alex", "Sarah", "Mike"]; // List of current users

let new\_users: string[] = ["John", "Alice", "Robert", "Sarah", "Emily"]; // List of new users

for (let i = 0; i < new\_users.length; i++) {

  let new\_username = new\_users[i];

  let username\_taken = false;

  for (let j = 0; j < current\_users.length; j++) {

    if (new\_username.toLowerCase() === current\_users[j].toLowerCase()) {

      username\_taken = true;

      break;

    }

  }

  if (username\_taken) {

    console.log(`The username '${new\_username}' is not available. Please enter a new username.`);

  } else {

    console.log(`The username '${new\_username}' is available.`);

  }

}

[LOG]: "The username 'John' is not available. Please enter a new username."

[LOG]: "The username 'Alice' is available."

[LOG]: "The username 'Robert' is available."

[LOG]: "The username 'Sarah' is not available. Please enter a new username."

[LOG]: "The username 'Emily' is available."

1. Ordinal Numbers: Ordinal numbers indicate their position in a array, such as 1st or 2nd. Most ordinal numbers end in th, except 1, 2, and 3.

• Store the numbers 1 through 9 in a array.

• Loop through the array.

• Use an if-else chain inside the loop to print the proper ordinal ending for each number. Your output should read "1st 2nd 3rd 4th 5th 6th 7th 8th 9th", and each result should be on a separate line.

const number: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];

for (let i = 0; i < numbers.length; i++) {

  const number = numbers[i];

  let ordinalEnding: string;

  if (number === 1) {

    ordinalEnding = "st";

  } else if (number === 2) {

    ordinalEnding = "nd";

  } else if (number === 3) {

    ordinalEnding = "rd";

  } else {

    ordinalEnding = "th";

  }

  console.log(`${number}${ordinalEnding}`);

}

[LOG]: "1st"

[LOG]: "2nd"

[LOG]: "3rd"

[LOG]: "4th"

[LOG]: "5th

1. Pizzas: Think of at least three kinds of your favorite pizza. Store these pizza names in a array, and then use a for loop to print the name of each pizza.

• Modify your for loop to print a sentence using the name of the pizza instead of printing just the name of the pizza. For each pizza you should have one line of output containing a simple statement like I like pepperoni pizza.

• Add a line at the end of your program, outside the for loop, that states how much you like pizza. The output should consist of three or more lines about the kinds of pizza you like and then an additional sentence, such as I really love pizza!

const favoritePizzas: string[] = ["Pepperoni", "Margherita", "BBQ Chicken"];

for (let i = 0; i < favoritePizzas.length; i++) {

  const pizza = favoritePizzas[i];

  console.log(`I like ${pizza} pizza.`);

}

console.log("I really love pizza!");

[LOG]: "I like Pepperoni pizza."

[LOG]: "I like Margherita pizza."

[LOG]: "I like BBQ Chicken pizza."

[LOG]: "I really love pizza!"

1. Animals: Think of at least three different animals that have a common characteristic. Store the names of these animals in a list, and then use a for loop to print out the name of each animal. • Modify your program to print a statement about each animal, such as A dog would make a great pet. • Add a line at the end of your program stating what these animals have in common. You could print a sentence such as Any of these animals would make a great pet!
2. const animals: string[] = ["Dog", "Cat", "Rabbit"];
3. for (let i = 0; i < animals.length; i++) {
4. const animal = animals[i];
5. console.log(`A ${animal.toLowerCase()} would make a great pet.`);
6. }
7. console.log("Any of these animals would make a great pet!");
8. [LOG]: "A dog would make a great pet."
9. [LOG]: "A cat would make a great pet."
10. [LOG]: "A rabbit would make a great pet."
11. [LOG]: "Any of these animals would make a great pet!"

36.T-Shirt: Write a function called make\_shirt() that accepts a size and the text of a message that should be printed on the shirt. The function should print a sentence summarizing the size of the shirt and the message printed on it. Call the function.

function makeShirt(size: string, message: string): void {

  console.log(`Printing a ${size}-sized shirt with the message: "${message}"`);

}

// Calling the function

makeShirt('M', 'Hello, World!');

[LOG]: "Printing a M-sized shirt with the message: "Hello, World!""

37.Large Shirts: Modify the make\_shirt() function so that shirts are large by default with a message that reads I love TypeScript. Make a large shirt and a medium shirt with the default message, and a shirt of any size with a different message.

function make\_shirt(size: string = 'large', message: string = 'I love TypeScript') {

  console.log(`Creating a ${size} shirt with the message: ${message}`);

}

// Creating a large shirt with the default message

make\_shirt();

// Creating a medium shirt with the default message

make\_shirt('medium');

// Creating a small shirt with a different message

make\_shirt('small', 'I love programming');

[LOG]: "Creating a large shirt with the message: I love TypeScript"

[LOG]: "Creating a medium shirt with the message: I love TypeScript"

[LOG]: "Creating a small shirt with the message: I love programming"

38. Cities: Write a function called describe\_city() that accepts the name of a city and its country. The function should print a simple sentence, such as Karachi is in Pakistan. Give the parameter for the country a default value. Call your function for three different cities, at least one of which is not in the default country.

function describe\_city(city: string, country: string = 'default country') {

  console.log(`${city} is in ${country}.`);

}

// Calling the function for three different cities

describe\_city('Karachi', 'Pakistan');

describe\_city('Paris', 'France');

describe\_city('New York', 'United States');

[LOG]: "Karachi is in Pakistan."

[LOG]: "Paris is in France."

[LOG]: "New York is in United States."

39.City Names: Write a function called city\_country() that takes in the name of a city and its country. The function should return a string formatted like this:

"Lahore, Pakistan"

Call your function with at least three city-country pairs, and print the value that’s returned.

function city\_country(city: string, country: string): string {

  return `${city}, ${country}`;

}

// Calling the function with city-country pairs

const location1 = city\_country('Lahore', 'Pakistan');

const location2 = city\_country('London', 'United Kingdom');

const location3 = city\_country('Tokyo', 'Japan');

// Printing the returned values

console.log(location1);

console.log(location2);

console.log(location3);

[LOG]: "Lahore, Pakistan"

[LOG]: "London, United Kingdom"

[LOG]: "Tokyo, Japan"

1. Album: Write a function called make\_album() that builds a Object describing a music album. The function should take in an artist name and an album title, and it should return a Object containing these two pieces of information. Use the function to make three dictionaries representing different albums. Print each return value to show that Objects are storing the album information correctly. Add an optional parameter to make\_album() that allows you to store the number of tracks on an album. If the calling line includes a value for the number of tracks, add that value to the album’s Object. Make at least one new function call that includes the number of tracks on an album.
2. function make\_album(artist: string, title: string, tracks?: number): object {
3. const album: any = {
4. artist: artist,
5. title: title
6. };
7. if (tracks) {
8. album.tracks = tracks;
9. }
10. return album;
11. }
12. // Calling the function to create album objects
13. const album1 = make\_album('Artist 1', 'Album 1');
14. const album2 = make\_album('Artist 2', 'Album 2', 12);
15. const album3 = make\_album('Artist 3', 'Album 3', 8);

// Printing the album objects

1. console.log(album1);
2. console.log(album2);
3. console.log(album3);
4. [LOG]: { "artist": "Artist 1", "title": "Album 1" }
5. [LOG]: { "artist": "Artist 2", "title": "Album 2", "tracks": 12 }
6. [LOG]: { "artist": "Artist 3", "title": "Album 3", "tracks": 8 }

41.Magicians: Make a array of magician’s names. Pass the array to a function called show\_magicians(), which prints the name of each magician in the array.

function show\_magicians(magicians: string[]): void {

  for (let magician of magicians) {

    console.log(magician);

  }

}

// Creating an array of magician names

const magicianNames: string[] = ['Harry Houdini', 'David Copperfield', 'Penn Jillette', 'Teller'];

// Calling the show\_magicians() function with the magicianNames array

show\_magicians(magicianNames);

[LOG]: "Harry Houdini"

[LOG]: "David Copperfield"

[LOG]: "Penn Jillette"

[LOG]: "Teller"

42.Great Magicians: Start with a copy of your program from Exercise 39. Write a function called make\_great() that modifies the array of magicians by adding the phrase the Great to each magician’s name. Call show\_magicians() to see that the list has actually been modified.

function show\_magicians(magicians: string[]): void {

  for (let magician of magicians) {

    console.log(magician);

  }

}

function make\_great(magicians: string[]): string[] {

  const greatMagicians: string[] = [];

  for (let magician of magicians) {

    const greatMagician = magician + ' the Great';

    greatMagicians.push(greatMagician);

  }

  return greatMagicians;

}

// Creating an array of magician names

const magicianNames: string[] = ['Harry Houdini', 'David Copperfield', 'Penn Jillette', 'Teller'];

// Calling the make\_great() function to modify the magician names

const greatMagicianNames = make\_great(magicianNames);

// Calling the show\_magicians() function to display the modified magician names

show\_magicians(greatMagicianNames);

[LOG]: "Harry Houdini the Great"

[LOG]: "David Copperfield the Great"

[LOG]: "Penn Jillette the Great"

[LOG]: "Teller the Great"

43.Unchanged Magicians: Start with your work from Exercise 40. Call the function make\_great() with a copy of the array of magicians’ names. Because the original array will be unchanged, return the new array and store it in a separate array. Call show\_magicians() with each array to show that you have one array of the original names and one array with the Great added to each magician’s name.

function show\_magicians(magicians: string[]): void {

  for (let magician of magicians) {

    console.log(magician);

  }

}

function make\_great(magicians: string[]): string[] {

  const greatMagicians: string[] = [];

  for (let magician of magicians) {

    const greatMagician = magician + ' the Great';

    greatMagicians.push(greatMagician);

  }

  return greatMagicians;

}

// Creating an array of magician names

const magicianNames: string[] = ['Harry Houdini', 'David Copperfield', 'Penn Jillette', 'Teller'];

// Calling the make\_great() function to modify the magician names

const greatMagicianNames = make\_great(magicianNames);

// Calling the show\_magicians() function to display the modified magician names

show\_magicians(greatMagicianNames);

[LOG]: "Harry Houdini the Great"

[LOG]: "David Copperfield the Great"

[LOG]: "Penn Jillette the Great"

[LOG]: "Teller the Great"

44.Sandwiches: Write a function that accepts a array of items a person wants on a sandwich. The function should have one parameter that collects as many items as the function call provides, and it should print a summary of the sandwich that is being ordered. Call the function three times, using a different number of arguments each time.

function make\_sandwich(...items: string[]): void {

  console.log("Sandwich Order Summary:");

  console.log("Bread");

  for (let item of items) {

    console.log(item);

  }

  console.log("Bread");

  console.log("------------------------");

}

// Calling the function with different number of arguments

make\_sandwich("Cheese", "Tomato");

make\_sandwich("Ham", "Lettuce", "Mayonnaise", "Mustard");

make\_sandwich("Turkey", "Avocado", "Bacon", "Swiss Cheese", "Pickles");

[LOG]: "Sandwich Order Summary:"

[LOG]: "Bread"

[LOG]: "Cheese"

[LOG]: "Tomato"

[LOG]: "Bread"

[LOG]: "------------------------"

[LOG]: "Sandwich Order Summary:"

[LOG]: "Bread"

[LOG]: "Ham"

[LOG]: "Lettuce"

[LOG]: "Mayonnaise"

[LOG]: "Mustard"

[LOG]: "Bread"

[LOG]: "------------------------"

[LOG]: "Sandwich Order Summary:"

[LOG]: "Bread"

[LOG]: "Turkey"

[LOG]: "Avocado"

[LOG]: "Bacon"

[LOG]: "Swiss Cheese"

[LOG]: "Pickles"

[LOG]: "Bread"

[LOG]: "------------------------"

45.Cars: Write a function that stores information about a car in a Object. The function should always receive a manufacturer and a model name. It should then accept an arbitrary number of keyword arguments. Call the function with the required information and two other name-value pairs, such as a color or an optional feature. Print the Object that’s returned to make sure all the information was stored correctly

function store\_car\_info(manufacturer: string, model: string, ...options: { [key: string]: any }[]): object {

  const car: any = {

    manufacturer: manufacturer,

    model: model,

    ...options

  };

  return car;

}

// Calling the function to store car information

const carInfo = store\_car\_info('Toyota', 'Camry', {color: 'Silver', features: ['GPS', 'Sunroof']});

// Printing the returned object to verify the stored information

console.log(carInfo);

[LOG]: { "0": { "color": "Silver", "features": [ "GPS", "Sunroof" ] }, "manufacturer": "Toyota", "model": "Camry" }