Getting Started Exercises with TypeScript and Node.js

Note: Try these short programs to get some firsthand experience with TypeScript and Node.js. You might want to create a new folder for each exercise to keep them organized. Create a single Github repository to commit the code for these exercises and once finished submit the URL of the repo.

1. Install Node.js, TypeScript and VS Code on your computer.

2. 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?"

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

4. 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."

5. 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.

6. 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.

7. Number Eight: Write addition, subtraction, multiplication, and division operations that each result in the number 8. Be sure to enclose your operations in print statements to see the results.

8. 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.

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

10. 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.

11. 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.

12. 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.

13. 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."

14. 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.

15. 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.

16. 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.

17. 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.

• Start with your program from Exercise 16. Add a new line that prints a message saying that you can invite only two people for dinner.

• Remove guests from your list one at a time until only two names remain in your list. Each time you pop a name from your list, print a message to that person letting them know you're sorry you can't invite them to dinner.

• Print a message to each of the two people still on your list, letting them know they're still invited.

• Remove the last two names from your list, so you have an empty list. Print your list to make sure you actually have an empty list at the end of your program.

18. 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.

19. 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.

20. 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.

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

22. 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.

23. 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.

24. 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

25. 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.)

26. 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.

27. 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.

28. 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.

29. 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!

30. 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.

31. 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.

32. 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.

33. 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.

34. 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!

35. 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!

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.

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.

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.

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.

40. 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.

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.

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.

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.

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.

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.