**Getting Started Exercises with TypeScript and Node.js**

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?”

let personName = "Eric";

let msg=`Hello ${personName} would you like to learn some Python today ?`;

console.log(msg);

2. Name Cases: Store a person’s name in a variable, and then print that person’s name in lowercase, uppercase, and title case.

let personName = "AbRaHaM LiNcOlN";

function toTitleCase(str:string) {

    return str.toLowerCase().replace(/(?:^|\s)\w/g, function (match) {

        return match.toUpperCase();

    });

}

console.log(personName.toLowerCase());

console.log("<br>", personName.toUpperCase());

console.log("<br>", toTitleCase(personName));

3. 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:

console.log('Albert Einstein once said, "A person who never made a mistake never tried anything new."')

4. 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.

let famous\_person = "Albert Einstein";

let message = ' once said, "A person who never made a mistake never tried anything new."';

console.log(famous\_person, message);

5. 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.

let famous\_person = "\t\t\tAlbert\t\t\t\n\t\t\tEinstein\t\t\t";

console.log(famous\_person);

let updatedPerson = famous\_person.replace(/\t|\n/g, "");

console.log(updatedPerson);

6. 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.

let addition = 4 + 4;

console.log("addition: 4 + 4 = " + addition);

let Subtaction = 10 - 2;

console.log("Subtaction: 10 - 2 = " + Subtaction);

let Multiplication = 2 \* 4;

console.log("Multiplication: 2 × 4 = " + Multiplication);

let Division = 80 / 10;

console.log("Division: 80 ÷ 10 = " + Division);

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

console.log("80 + 80");

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

console.log(4 + 4);

console.log(10 - 2);

console.log(2 \* 4);

console.log(80 / 10);

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

let my\_fav\_num = "My favourite number is :" + 2;

console.log(my\_fav\_num);

9. 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.

//By Muhammad Bilawal

//Date: 7/8/2023

let my\_fav\_num = "My favourite number is :" + 2;

console.log(my\_fav\_num);

/\*By Muhammad Bilawal

Date: 7/8/2023\*/

let famous\_person = "\t\t\tAlbert\t\t\t\n\t\t\tEinstein\t\t\t";

console.log(famous\_person);

let updatedPerson = famous\_person.replace(/\t|\n/g, "");

console.log(updatedPerson);

10. 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.

let names = ["Eric", "Gregory", "Sam", "Ellie"];

names.map(word => console.log(word));

11. 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.

let names = ["Eric", "Gregory", "Sam", "Ellie"];

names.map(function (word) { return console.log("To " + word + ": Hi how are you today?"); });

12. 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.”

let my\_fav\_items = ["I would like to own a BMW 250 Heavy Bike", "I would like to own a Lamborghini Sian car.", "I would like to own a Thunder speed golden plated jet."];

my\_fav\_items.map(function (items) { return console.log(items); });

13. 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.

let invited = ["Mom","Dad","Cousin"];

invited.map(function (items) { return console.log(`Hi ${items} would you like to come in todays dinner party?`); });

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

let invited = ["Mom", "Dad", "Cousin"];

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

console.log("Oh no! Cousin said that they can not make it to the party!");

invited[2] = "Aunt"

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

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

let invited = ["Mom", "Dad", "Cousin"];

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

console.log("Oh no! Cousin said that they can not make it to the party!");

invited[2] = "Aunt";

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

console.log("Everyone i found a bigger table. I am now going to invite more people!");

invited.unshift("Grandpa");

invited.splice(3, 0, "Grandma");

invited.push("Uncle");

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

16. 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.

let invited = ["Mom", "Dad", "Cousin"];

invited.map(function (names) { return console.log(`Hi + names + would you like to come in todays dinner party?`); });

console.log("Oh no! Cousin said that they can not make it to the party!");

invited[2] = "Aunt";

invited.map(function (names) { return console.log("Hi " + names + " would you like to come in todays dinner party?"); });

console.log("Everyone i found a bigger table. I am now going to invite more people!");

invited.unshift("Grandpa");

invited.splice(3, 0, "Grandma");

invited.push("Uncle");

invited.map(function (names) { return console.log("Hi " + names + " would you like to come in todays dinner party?"); });

console.log("Oh no the dinner table wont arrive in time i will have to remove some people from the guest list!")

invited.pop()

console.log("Sorry Uncle but you cant come now!")

invited.pop()

console.log("Sorry Aunt but you cant come now!")

invited.pop()

console.log("Sorry Grandma but you cant come now!")

invited.pop()

console.log("Sorry Dad but you cant come now!")

console.log("Mom and Grandpa you are still invited!")

invited.pop()

invited.pop()

console.log(invited);

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

let placesToVisit= ["Tokyo", "Paris", "Bora Bora", "New York", "Sydney"];

console.log("Original Order:", placesToVisit);

console.log("Alphabetical Order:", [...placesToVisit].sort());

console.log("Original Order (Unchanged):", placesToVisit);

console.log("Reverse Alphabetical Order:", [...placesToVisit].sort().reverse());

console.log("Original Order (Unchanged):", placesToVisit);

placesToVisit.reverse();

console.log("Reversed Order:", placesToVisit);

placesToVisit.reverse();

console.log("Original Order (Unchanged):", placesToVisit);

placesToVisit.sort();

console.log("Alphabetical Order:", placesToVisit);

placesToVisit.reverse();

console.log("Reverse Alphabetical Order:", placesToVisit);

18. 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.

// question no. 16

let invited = ["Mom", "Dad", "Cousin"];

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

console.log("Oh no! Cousin said that they can not make it to the party!");

invited[2] = "Aunt";

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

console.log("Everyone! i found a bigger table. I am now going to invite more people!");

invited.unshift("Grandpa");

invited.splice(3, 0, "Grandma");

invited.push("Uncle");

invited.map(function (items) { return console.log("Hi " + items + " would you like to come in todays dinner party?"); });

console.log(invited.length);

19. 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.

const countries= [];

countries.push("USA");

countries.push("Canada");

countries.push("Australia");

countries.push("Germany");

countries.push("Japan");

console.log(countries);

20. Think of something you could store in a TypeScript Object. Write a program that creates Objects containing these items.

interface Book {

    title: string

    author: string

    pages: number

}

let wimpykid: Book = {

    title: "Wimpy Kid",

    author: "Jeff kiney",

    pages: 175

}

21. 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.

22. 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'):

let num2 = 100;

console.log('is num2 >= "4"? i predict true');

console.log(num2 >= 4);

23. 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:

let num = 5;

console.log('is num <= "4"? i predict false');

console.log(num <= 4);

let num2 = 100;

console.log('is num2 >= "4"? i predict true');

console.log(num2 >= 4);

24. 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':

let alien\_color: string = "Green";

if (alien\_color.toLowerCase() == "green") {

    console.log("Player Just earned 5 points!")

}

else {

    // No output.

}

let alien\_color: string = "rainbow";

if (alien\_color.toLowerCase() == "green") {

    console.log("Player Just earned 5 points!")

}

else {

    // No output.

}

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

let alien\_color: string = "green";

if (alien\_color.toLowerCase() == "green") {

    console.log("Player Just earned 5 points!")

}

else {

   console.log("Player Just earned 10 points!")

}

let alien\_color: string = "yellow";

if (alien\_color.toLowerCase() == "green") {

    console.log("Player Just earned 5 points!")

}

else {

   console.log("Player Just earned 10 points!")

}

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

let alien\_color: string = "green";

if (alien\_color.toLowerCase() == "green") {

    console.log("Player Just earned 5 points!")

}

else if(alien\_color.toLowerCase() == "yellow"){

    console.log("Player Just earned 10 points!");

}

else if(alien\_color.toLowerCase() == "red"){

    console.log("Player Just earned 15 points!");

}

else {

    // No output.

}

let alien\_color: string = "yellow";

if (alien\_color.toLowerCase() == "green") {

    console.log("Player Just earned 5 points!")

}

else if(alien\_color.toLowerCase() == "yellow"){

    console.log("Player Just earned 10 points!");

}

else if(alien\_color.toLowerCase() == "red"){

    console.log("Player Just earned 15 points!");

}

else {

    // No output.

}

let alien\_color: string = "red";

if (alien\_color.toLowerCase() == "green") {

    console.log("Player Just earned 5 points!")

}

else if(alien\_color.toLowerCase() == "yellow"){

    console.log("Player Just earned 10 points!");

}

else if(alien\_color.toLowerCase() == "red"){

    console.log("Player Just earned 15 points!");

}

else {

    // No output.

}

27. 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:

let age: number = 66;

if (age < 2) {

    console.log("Person is a Baby");

}

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

    console.log("Person is a Toodler");

}

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

    console.log("Person is a Kid");

}

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

    console.log("Person is a Teenager");

}

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

    console.log("Person is an Adult");

}

if (age >= 65) {

    console.log("Person is an Elder");

}

28. 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.

let favorite\_fruits: string[] = ["Leechi","Mango", "Cherry", "Banana", "Orange","Pineapple"];

if(favorite\_fruits.toString().toLowerCase().indexOf("chERRY".toLowerCase())>-1){

    console.log("I love Cherry!")

}

if (favorite\_fruits.toString().toLowerCase().indexOf("Mango".toLowerCase())>-1) {

    console.log("I love Mangoes!")

}

if(favorite\_fruits.toString().toLowerCase().indexOf("Banana".toLowerCase())>-1){

    console.log("I love Bananas!")

}

if(favorite\_fruits.toString().toLowerCase().indexOf("Orange".toLowerCase())>-1){

    console.log("I love Oranges!")

}

if(favorite\_fruits.toString().toLowerCase().indexOf("Pineapple".toLowerCase())>-1){

    console.log("I love Pineapples!")

}

29. 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:

let names: string[] = ["George", "Admin", "Ali", "Eric", "Alia"];

names.map(greet => greet == "Admin" ? console.log("Hello " + greet + ", would you like to see a status report?") : console.log("Hello " + greet + ", thank you for logging in again"));

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

let names: string[] = ["George", "Admin", "Ali", "Eric", "Alia"];

names.length == 0 ?

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

:

names.map(greet =>

    greet == "Admin" ?

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

    :console.log("Hello " + greet + ", thank you for logging in again!"))

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

let Current\_users: string[] = ["bilal", "George", "ali", "eric", "alia"];

let new\_users: string[] = ["conor", "Ali", "imran", "john", "george"];

new\_users.map(x => Current\_users.toString().toLowerCase().indexOf(x.toLowerCase()) > -1 ?

    console.log("Already taken need to enter a new username")

    :console.log(`${x} username is available`))

32. 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.

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

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

    let element = num[i];

    element == 1 ? console.log(`${element}st`)

    :element == 2 ? console.log(`${element}nd`)

    :element == 3? console.log(`${element}rd`)

    :console.log(`${element}th`)

}

33. 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.

let fav\_pizza\_flavors:string[]=["Chicken Tikka","Chicken Fajita","Smoked Chicken"]

for (let i = 0; i < fav\_pizza\_flavors.length; i++) {

    const element = fav\_pizza\_flavors[i];

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

}

console.log("I love pizza so much as each slice brings me sheer joy and satisfaction.The aroma of freshly baked dough and melted cheese fills my senses, igniting a passion that only pizza can fulfill")

34. 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.

let animals:string[]=["dog","cat","duck"]

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

    const element = animals[i];

   return console.log(`I really like ${element}s`)

}

console.log("Any of these animals would make a great pet!")

35. 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 make\_shirt(msg:string="I love Typescript",size:"S"|"M"|"L" = "L"){

   return console.log(`The size of the shirt is ${size}. With the message ${msg}.`)

}

make\_shirt();

36. 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:"S"|"M"|"L"|"Free" = "L",msg:string="I love Typescript"){

   return console.log(`The size of the shirt is ${size}. With the message ${msg}.`)

}

make\_shirt();

function make\_shirt(size:"S"|"M"|"L"|"Free" = "L",msg:string="I love Typescript"){

   return console.log(`The size of the shirt is ${size}. With the message ${msg}.`)

}

make\_shirt("M");

function make\_shirt(msg:string="I love Typescript",size:"S"|"M"|"L"|"Free" = "L"){

   return console.log(`The size of the shirt is ${size}. With the message ${msg}.`)

}

make\_shirt("I Love JS","Free");

37. 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="Pakistan") {

   return console.log(`${City} is in ${Country}`);

}

describe\_city("Karachi");

function describe\_city(City:string,Country:string="Pakistan") {

   return console.log(`${City} is in ${Country}`);

}

describe\_city("Islamabad");

function describe\_city(City:string,Country:string="Pakistan") {

   return console.log(`${City} is in ${Country}`);

}

describe\_city("New York","United States of America");

38. 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:

function city\_country(City:string,Country:string) {

    return console.log(`${City}, ${Country}`);

}

city\_country("New York","United States of America");

function city\_country(City:string,Country:string) {

    return console.log(`${City}, ${Country}`);

}

city\_country("Islamabad","Pakistan");

function city\_country(City:string,Country:string) {

    return console.log(`${City}, ${Country}`);

}

city\_country("Chongqing","China");

39. 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.

type musicAlbum = {

    artistName: string,

    albumTitle: string,

    songs:number | null

}

function make\_album(artist:string,title:string, Tracks?: number): musicAlbum {

    let album = {

        artistName: artist,

        albumTitle: title,

        songs:Tracks=== undefined ? null : Tracks

    }

    return album;

}

let Music = {BU:{

                FD: make\_album("BoyWithUke", "Fever Dreams"),

                MD: make\_album("BoyWithUke", "Melatonin Dreams",13),

                SD: make\_album("BoyWithUke", "Serotonin Dreams",11),

    },

    AM: {

        CH: make\_album("Aleemrk", "Cold Hours",3),

        LG: make\_album("Aleemrk", "Life Goes on"),

        LC: make\_album("Aleemrk", "Life in the city",7),

    },

    Ruth:{

        MS: make\_album("Ruth.B", "Mixed Signals",5),

        UR:make\_album("Ruth.B", "UnRightnous"),

        Dan:make\_album("Ruth.B", "Dandelions",4),

    }

}

for (const key of Object.keys(Music)) {

    for (const key2 of Object.keys(Music[key])) {

        let tr = Music[key][key2].songs === null ? "" : `, ${Music[key][key2].songs}`

        console.log("Artist: ",Music[key][key2].artistName,", AlbumTitle: ",Music[key][key2].albumTitle, tr)

    }

}

40. 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.

let magicians: string[] = ["Eric", "Elsa", "Ruby", "Ali"];

function show\_magicians() {

    magicians.map(function (names) { return console.log("Magician "+names); });

}

show\_magicians();

41. 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.

let magicians: string[] = ["Eric", "Elsa", "Ruby"];

function make\_great() {

    magicians = magicians.map((names) => names = names + " The Great ");

}

make\_great();

function show\_magicians() {

    console.log(magicians.toString());

}

show\_magicians();

42. 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.

let Orignal\_magicians: string[] = ["Eric", "Elsa", "Ruby"];

let modified\_magicians: string[] = [];

function make\_great() {

    modified\_magicians= Orignal\_magicians.map((names) => names = names + " The Great ");

}

make\_great();

function show\_magicians(arr1:string[]) {

    console.log(arr1.toString());

}

console.log("Modified array: ")

show\_magicians(modified\_magicians);

console.log("Orginal array: ")

show\_magicians(Orignal\_magicians)

43. Sandwiches: Write a function that accepts an 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 Sandwich\_things(Things:string[]) {

    console.log("The ordered sandwich has the following things: "+Things.toString())

}

Sandwich\_things(["Bread", "Tomato", "Onions","Lettuce","Mayonise","Ketchup","Fried chicken"])

44. Cars: Write a function that stores information about a car in an 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.

interface Car  {

    manufacturer: string,

    model\_name: string

}

function car\_making(manufacturer: string, model\_name: string, ...features: any[]):Car {

    let build\_car = {

        manufacturer: manufacturer,

        model\_name: model\_name,

        ...Object.assign({},...features)

    }

    return build\_car;

}

console.log(car\_making("BMW", "M2", { Color: "red", price: "200000", optional\_feature:"SunRoof"}));