Convert the following Function ***myFunction*** into an arrow function. Refer to the new function as ***returnSum***.

function myFunction(**a**,**b**) {

return **a**+**b**;

}

// WRITE YOUR CODE HERE

let returnSum = (a,b) => a + b;

Convert the following Function ***myFunction*** into an arrow function. Refer to the new function as ***findSquareRoot***.

**function** **myFunction**(num) {

**return** Math.sqrt(num);

}

// WRITE YOUR CODE HERE

let findSquareRoot = (num) => Math.sqrt(num);

Convert the following Function ***myFunction*** into an arrow function. Refer to the new function as ***pythagoras***.

**function** **myFunction**(num1, num2) {

**return** Math.sqrt((num1 \* num1) + (num2 \* num2));

// WRITE YOUR CODE HERE

let pythagoras = (num1, num2) => Math.sqrt((num1 \* num1) + (num2 \* num2));

1. **Arrow Functions require curly braces if code is on more than one line.**
2. ✅That's Correct! Curly braces are used to define the Function body.

Convert the following Function ***myFunction*** into an arrow function. Refer to the new function as ***yourNumber***.

**function** **myFunction**(num) {

**return** `Your number is ${num}`;

}

// WRITE YOUR CODE HERE

let yourNumber = (num) => `Your number is ${num}`;

* The **function** keyword is replaced with an arrow **=>**.
* Since there's only one parameter **num**, parentheses around the parameter are optional, but you can keep them if you prefer.
* The function name **myFunction** is removed because arrow functions are anonymous by default.
* The function body is simplified to a single expression that returns a string interpolation of "Your number is " followed by the value of **num**.
* The arrow function **yourNumber** is assigned to a constant using **const**.

**Arrow functions can contain either a *block* body (using { } and containing an explicit**return**statement), or a *concise* body (a single expression).**

True

✅That's Correct! Arrow Functions can be one-liners or multiple lines long.

Convert the following Function ***myFunction*** into an arrow function. Refer to the new function as ***monthlyPayment***.

**function** **myFunction**(yearlyPayment) {

**let** monthly = yearlyPayment / 12;

**return** monthly.toFixed(2);

}

// WRITE YOUR CODE HERE

let monthlyPayment = (yearlyPayment) => {

let monthly = yearlyPayment / 12;

return monthly.toFixed(2);

};

Convert the following Function ***myFunction*** into an arrow function. Refer to the new function as ***fancyAlgorithm***.

**function** **myFunction**(num1, num2, num3) {

**let** value = 0;

**for** (**let** index = 0; index < num3; index++) {

value = (value + num2) \* num1;

}

**return** value / (num1 \* (num3 \* 10));

}

// WRITE YOUR CODE HERE

let fancyAlgorithm = (num1, num2, num3) => {

let value = 0;

for(let index = 0; index < num3; index++){

value = (value + num2) \* num1;

}

return value / (num1 \* (num3 \* 10));

};

* The **function** keyword is replaced with an arrow **=>**.
* The function name **myFunction** is removed because arrow functions are anonymous by default.
* The function body, which includes a loop and multiple statements, is wrapped in curly braces **{}**.
* The arrow function **fancyAlgorithm** is assigned to a constant using **const**.
* **Which of the following is the correct syntax when no arguments are passed? Select all that apply.**
* let myFunction = () => { **return** "Test Successful!"; }
* **Why this option is correct:**
* ✅That's Correct! We have used a return here, that implies use of a block body with { }.

let myFunction = () => "Test Successful!";

**Why this option is correct:**

✅That's Correct! Empty parentheses are still required when there is no parameter.

Convert the following Function **myFunction** into an arrow function. Refer to the new function as **multiplyByAdding**.

**function** **myFunction**(a, b) {

**let** result = 0;

**for** (**let** i = 0; i < b; i++) {

result += a;

}

**return** result;

}

// WRITE YOUR CODE HERE

let multiplyByAdding = (a, b) => {

let result = 0;

for (let i = 0; i < b; i++) {

result += a;

}

return result;

}

* The **function** keyword is replaced with an arrow **=>**.
* The function name **myFunction** is removed because arrow functions are anonymous by default.
* The function body, which includes a loop and multiple statements, is wrapped in curly braces **{}**.
* The arrow function **multiplyByAdding** is assigned to a constant using **const**.

**What is the output of the following arrow function?**

let myArrowFunction = (a, b) => a \* b;

console.log(myArrowFunction(4, 5));

✅That's Correct! The numbers 4 and 5 are passed in as arguments to the Function, therefore 4 \* 5 = 20.

Convert the following Function **myFunction** into an arrow function. Refer to the new function as **revert**.

**function** **myFunction**(value){

**return** !value;

}

// WRITE YOUR CODE HERE

let revert = (value) => !value;

* The **function** keyword is replaced with an arrow **=>**.
* Since there's only one parameter **value**, parentheses around the parameter are optional, but you can keep them if you prefer.
* The function name **myFunction** is removed because arrow functions are anonymous by default.
* The function body is simplified to a single expression **!value**, which returns the negation of the input **value**.
* The arrow function **revert** is assigned to a constant using **const**.

Convert the following Function **myFunction** into an arrow function. Refer to the new function as **xor**.

**function** **myFunction**(num1, num2) {

**return** num1 ^ num2;

}

// WRITE YOUR CODE HERE

let xor = (num1, num2) => num1 ^ num2;

* The **function** keyword is replaced with an arrow **=>**.
* The function name **myFunction** is removed because arrow functions are anonymous by default.
* The function body is simplified to a single expression **num1 ^ num2**, which returns the result of the bitwise XOR operation between **num1** and **num2**.
* The arrow function **xor** is assigned to a constant using **const**.

Convert the following Function **myFunction** into an arrow function.

* The challenge here is to get this down to only one line of code
* Refer to the new function as **perfectSquare**

**function** **myFunction**(param) {

**let** total = Math.pow(param, 2);

**return** total;

}

// WRITE YOUR CODE HERE

let perfectSquare = param => Math.pow(param, 2);

* The **function** keyword is replaced with an arrow **=>**.
* Since there's only one parameter **param**, parentheses around the parameter are optional, but you can keep them if you prefer.
* The function name **myFunction** is removed because arrow functions are anonymous by default.
* The function body is simplified to a single expression **Math.pow(param, 2)**, which calculates the square of **param** using the **Math.pow** function.
* The arrow function **perfectSquare** is assigned to a constant using **const**.

1. **What is the output of the following arrow function?**

**let** myArrowFunction = (a, b) => {

**let** sum = a + b;

**let** difference = a - b;

**return** sum \* difference;

}

console.log(myArrowFunction(5, 3));

✅That's Correct! Since 5 and 3 are passed in, we get sum = 5 + 3 = 8, difference = 5 - 3 = 2, then sum \* difference = 8 \* 2 = 16.

**GitHub Cheat Sheet**

**Please Carefully read each instruction before proceeding.**

**1. Adding a new repository**

#Adds a readme file

echo "# Your first commit message!" >> README.md

#initializes the git library

git init

#Adds all project changes

git add .

#adds a commit message

git commit -m "added initial project and readme file"

#creates a branch called main

git branch -M main

#connects your diretory to a github repoistory

git remote add origin GIT\_REPO\_URL\_HERE

#pushes code from your computer to the main branch

git push -u origin main

**2. Run these three commands frequently!**

#adds all current project changes

git add .

#includes a commit message

git commit -m 'I made some code changes!'

#pushes your code from your project to your repository

git push

**3. Errors and Fixes**

#Identifies you as the user

git config --global user.email yourEmail456@email.com

git config --global user.name yourUsername789

#Amends your last commit with the new author

git commit --amend --reset-author

#Incorrect remote origin URL? Allows git to push to the correct repository

git remote set-url origin GIT\_REPO\_URL\_HERE

**4. Useful Commands and tips!**

#Display the current status of your files

git status

#Copies an existing repository to your computer

git clone https://github.com/some\_user/some\_repo

#clone a specific branch from a repository

git clone -b yourBranch https://github.com/some\_user/some\_repo

#Lists all branches local and remote

git branch -a

#Switch to another branch

git checkout some\_branch

#Create a new branch

git checkout -b new\_branch

#Add a specific file ie index.html

git add someFile.fileextension

#Commits to adding your changes to the repository

git commit -m 'new message'

#Updated your last commit with an ammended message

git commit --amend -m 'new message'

#Pushes changes up to your GitHub repository

git push

#Pulls changes from your GitHub repository

git pull

#Shows the history of your commits

git log

#Change a file name and keep the History

git mv old\_file\_name.js new\_file\_name.html

#Delete a file and keep the History

git rm 'file\_name'

#Remove a file from GitHub but not locally

git rm --cached 'file\_name'

#Allows git to push to the correct repository

#Useful if you have a different GitHub account

git remote set-url origin git://new\_url\_here