Lesson 03.01 - FINAL

In this lesson:

- what's a function?
- defining (declaring) a function
- calling a function
- variable scope
  - global variables
  - function scoped variables
  - other block scoped contexts
- function parameters and arguments
- default parameter values
- skipping a paramter
- function return values

What's a function? A function:

- is a block of code that runs only when invoked (called)
- is analogous to a real-world input-output machine, e.g. a coffee maker
- often (but not always) has a name by which to call it
- carries out some action, so it is customary for the name to be, or at least begin, with a verb
- must be be called somehow by the application, such as by a button click
- can take arguments (inputs), which it works with inside the function
- will often return a value--that is produce an "answer"
- can be passed to another function as its input (callback)
- definition is hoisted (lifted) to the top of the script, meaning it can be called from lines above where the function is written

---

Defining (declaring) a function

To declare or define a named function, follow these steps:

1. Start with the keyword: function
2. Give the function a name. Variable naming rules apply.
3. Tack on a pair of parentheses. These are for passing data into the function:
4. Next, add a pair of curly braces.
5. Inside the curly braces goes the code which runs when the function is called. For starters, let's just log "Hello World":

```
function greetWorld() {
    console.log("Hello World");
}
```

Run the script, and check the Console.

Nothing appears, because a function must be called; ll we have done so far is define the function.

---

calling a function

// We do not have a button to click, or other web page element for calling the function, so's just call it directly in the code. // Call a function by writing its name, followed by parentheses

6. Call the function by name, refresh and check the Console. Now we should see 'Hello World'. Also, check the data type--it's function:

```
greetWorld(); // Hello World
console.log('greetWorld datatype:', typeof(greetWorld));
// greetWorld() datatype: function
```

---

variable scope

global variables

- A **var**, **let** or **const** declared *outside* of any curly braces are **global** in scope.
- **global variables** are available *everywhere* in the script.

function scoped variables

- A **var**, **let** or **const** declared *inside* the curly braces of a function is **function scoped**, that is, local to that function. Outisde of the function, the variables do not exist.

other block scoped contexts

A let or const declared inside the curly braces of an "if statement" are also block scoped, but a var so declared is global in scope.

7. Declare two variables inside the curly braces of an "if statement":

```
let meal = "dinner";

if(meal == "dinner") {
    var food = "bison burger";
    let bev = "ginger ale";
    const TIP = 0.15;
    console.log(`INSIDE IF-BLOCK: I'll have a ${food} and ${bev},
please!`);
}
```

8. Try to access 'food' outside of the if-block.

```
console.log(`OUTSIDE IF-BLOCK: I'll have a ${food}, please!`);
// OUTSIDE IF-BLOCK: I'll have a bison burger, please!
```

This works, because 'var' variables are global--that is, NOT confined to the curly braces in which they are declared.

9. Now try logging 'bev' and 'TIP' outside of the function. Both attempts fail, because 'let' and 'const' are "block scoped" (confined to their curlies).

```
console.log(TIP); // Error: TIP is not defined
console.log(bev); // Error: bev is not defined
```

10. Declare a function with variables declared inside the function:

```
function welcomePlayer() {
    var user = "Pika2";
    let score = 500;
    console.log(`Welcome, ${user}! Your score is ${score}!`);
}
```

11. Call the function and check the console. We get the expected output:

```
welcomePlayer(); // Welcome, Pika2! Your score is 500.
```

12. Try to access the function variables (user and score) OUTSIDE the function:

```
// console.log(user); // Error: user is not defined -- even though it is a
'var'
// console.log(score); // ERROR score is not defined
```

The errors result because any variable declared inside a function -- be it a 'var', 'let', 'const' -- is "function scoped", meaning available only inside the curly braces of the function.

---

function parameters and arguments

Parameters (params for short) are:

- variables belonging to a function
- they are declared inside the function parentheses
- parameters are local variables scoped to the function
- when a function is called, the params are assigned values

- arguments are the values assigned to parameters
- arguments are passed to the function when the function is called

-

13. Write a function with a parameter:

```
function greetPlayer(username) {
    console.log(`Greetings, ${username}!`);
}
```

14. Call the function twice, passing it different arguments each time:

```
greetPlayer("Brian123"); // Greetings, Brian123!
greetPlayer("Sally789"); // Greetings, Sally789!
```

If a function expects an argument, but none is provided, the missing value will be 'undefined'.

15. Call the function again, but this time omit the argument:

```
greetPlayer(); // Greetings, undefined!
```

16. Write a function with two parameters:

```
function greetWithScore(username, score) {
    console.log(`Greetings, ${username}! Your score is ${score}!`);
}
```

17. Call the function, passing in both expected arguments:

```
greetWithScore('Dazey12', 4500); // Hey, Dazey12! Your score is 4500.
```

18. Call the function again, but this time omit the score argument:

```
greetWithScore('Ida34'); // Hey, Ida34! Your score is undefined.
greetWithScore(5400); // Hey, 4543! Your score is undefined.
greetWithScore(); // Hey, undefined! Your score is undefined.
```

---

skipping a parameter

passing in 'undefined'

If a function expects two arguments, but only one is provided, it will assign the argument to the first parameter. This is why the username was set to 5400. To keep arguments "lined up" with their correct parameters pass in the argument as a name=value pair. The ignored param will be **undefined**

```
greetWithScore(score=5400); // Hey, undefined! Your score is 5400.
```

---

default parameter values

// To avoid 'undefined' output, assign params default values // When callilng the function you can provide your own arguments, // ignore all the arguments and leave the parens empty when calling the function or assign argument(s) as key value pairs:

19. Define a function with two parameters, each of which has a default value:

```
function greetScore(username="Friend", score="1000") {
    console.log(`Welcome, ${username}! Your score is ${score}!`);
}
```

20. Call the function with both, either and neither parameter provided a value:

```
greetScore("Brian", 12500); // Welcome, Brian! Your score is 12500!
greetScore("Jane"); // Welcome, Jane! Your score is 1000!
greetScore(score=7500); // Welcome, Friend! Your score is 7500!
greetScore(); // Welcome, Friend! Your score is 1000!
```

---

function return values

- so far, our functions have produced only console output.
- logging to the console is great for tesing and debugging, BUT console output that is not saved to a variable just goes "poof"
- once a function running, it is removed from memory; function variables, being block-scoped, "vanish" with the function
- to "preserve the result" of a function, don't just log it--RETURN it.
- a 'return' statement "exports" the function's "answer"
- put the keyword 'return' in front of the value to be "exported"
- to "capture" the return value, set the function call equal to a variable
- a 'return' statement also terminates/exits the function

21. Write a function addNumbers with two parameters.
    Have the function add the numbers together and log the sum:

```
function addNumbers(num1, num2) {
    console.log('logged sum:', num1 + num2);
}
```

22. Call the function and provide the expected numbers to add:

```
addNumbers(86, 79); // 165
```

It works, but the only problem is the 165 result is not available to us anywhere in our application, besides in the function itself and console.

23. With oue change we can fix this. Instead of logging the sum, return it:

```
function addNums(num1, num2) {
    return num1 + num2;
}
```

24. Call the function, but this time set the call equal to a variable. The return value -- the sum of 95 and 55 -- will be stored in the variable:

```
let sum = addNums(95, 55);
console.log('returned sum:', sum);
```

The sum variable is in our global scope and so, the sum value is available throughout the script. This would be useful if we were calling the function multiple times with different inputs and needed to save all the results.

25. Write a function called findArea, and give it two parameters, length and width, which the function multiplies together and returns:

```
function findArea(length, width) {
    return length * width;
}
```

26. Call the function several times, once for each room of a house.

- Each time pass in new length and width values.
- Set the function call equal to a variable each time to capture the return value:

```
let bedroomArea = findArea(18, 14);
console.log('bedroomArea:', bedroomArea); // bedroomArea: 252
```

```
let livingRoomArea = findArea(22, 17);
console.log('livingRoomArea:', livingRoomArea); // livingRoomArea: 374

let kitchenArea = findArea(12, 10);
console.log('kitchenArea:', kitchenArea); // kitchenArea: 120

let bathroomArea = findArea(7, 10);
console.log('bathroomArea:', bathroomArea); // kitchenArea: 70

let apartmentArea = bedroomArea + livingRoomArea + kitchenArea +
bathroomArea;
console.log('apartmentArea:', apartmentArea); // apartmentArea: 826
```

DONE: Lesson 03.01

NEXT: Lab 03.01

THEN: Lesson 03.02

03.01 Lab

- Write functions as instructed
- Assume valid input; if the function expects a number argument, provide number
- No function validation need be peformed (e.g. checking the datatype of inputs)

1. The weight of anything on the moon is one-sixth its weight on earth.

- Make a function called calcMoonWeight() that takes in an earth weight number as its argument
- Function calculates the moon weight equivalent, and returns that number.
- So if the input is 180, the output is 30.

```
function {
    return
}

console.log("Earth weight: 180");
// console.log("Moon weight:", calcMoonWeight(180));
// Earth weight: 180
// Moon weight: 30
```

2. Write a function that:
   - takes an integer as its input (argument)
   - squares that number (multiplies it by itself)
   - returns
   - so, if you input 4, it returns 16.

- run the function three times with different inputs.

```
function {
    return
}

let square;
console.log('4 sq = ', square); // 4 sq = 16

square = squareNum(5);
console.log('5 sq = ', square); // 5 sq = 25

square = squareNum(6);
console.log('6 sq = ', square); // 6 sq = 36
```

3. Write a function that:
    - takes an integer as its input (argument)
    - if the number is even, it squares the number
    - if the number is odd, it cubes the number
    - the function returns the answer
    - if the input 3, the output is 27 (3 x 3 x 3)
    - if the input 7, the output is 49 (7 x 7) Run the function three times with different values.

```
function squa n {
    // if n divided by 2 yields a remainder of 0, n is even
    if
        return // return the even number, squared
    else // else, assuming int, n is odd
        return// return the odd number, cubed

    // ternaray alternative to if-else
    // return n % 2 == 0 ? n ** 2 : n ** 3;
}

console.log(squareOrCubeNum(3)); // 27
console.log(squareOrCubeNum(4)); // 16
console.log(squareOrCubeNum(5)); // 125
console.log(squareOrCubeNum(6)); // 36
```

4. Declare a function introducePet() :
    - has four parameters: petType, petName, age, sound
    - there are no default parameter values;
    - the 4 expected arguments must be passed to the function when it's called
    - return a message that includes all four arguments. Example: Meow! My name is Fluffy! I am a 3-year-old cat!.
    - run the function three times, with different pet inputs each time

```
function intro petType, petName, age, sound
    return My name is ! I am a -year-old !;
}

console.log(introducePet("cat", "Fluffy", 3, "Meow"));
// Meow! My name is Fluffy! I am a 3-year-old cat!

console.log(introducePet("dog", "Fido", 2, "Woof"));
// Woof! My name is Fido! I am a 2-year-old dog!

console.log(introducePet("rabbit", "Bunny", 4, "Crunch"));
// Crunch! My name is Bunny! I am a 4-year-old rabbit!
```

5. Write a function called feetToMeters, which converst feet to meters

   - function takes feet as its input and returns meters
   - round answer to 2 decimal places
   - answer should be a number--not a string

   Conversion formulas: 12 inches = 1 foot 39.37 inches = 1 meter

```
function feetToMe {
    inches; // calc total inches
    meters; // use inches to get meters
    toFixed(2) // round to 2 decimals
    // + sign to convert metersRounded from str back to num
    return
}

console.log(`50 ft = ${feetToMeters(50)} m`); // 5 ft' = 15.2 m
console.log(`408 ft = ${feetToMeters(408)} m`); // 408 ft = 124.36 m
```

6. Write a function called convertDistance(), which is similar to feetToMeters() except that it converts both ways: feet-to-meters and meters-to-feet The logic must tell the function which conversion to do. To do this:
   - add a second parameter, a boolean
   - the boolean value should indicate which conversion to do
   - use if-else logic to check the boolean
   - if the boolean is true, convert one way, else go the other way

```
function dist isFeet
    // if isFeet is true, convert feet to meters
    if
        let meters
        let roundedMeters
        // return +roundedMeters;
        // Or, return all in one line:
```

```
        return
    // if isFeet is false, convert meters to feet
    else
        // let feet = dist * 39.37 / 12;
        // let roundedFeet = feet.toFixed(2);
        // return +roundedFeet;
        // Or, return all in one line:
        return
    }
}

console.log('500 ft =', convertDistance(500, true), 'm'); // 500 ft =
152.4 m
console.log('150 m =', convertDistance(150, false), 'ft'); // 150 m =
492.13 ft
```

7. Refactor the function to return feet in feet-and-inches format, which is a string; use this style: "4ft 8in"
    - Hint: Use the % operator to get inches

```
function dist isFeet
    // if isFeet is true, convert feet to meters
    if
        let meter
        let roundedMeters
        // return +roundedMeters;
        // Or, return all in one line:
        return
    // if isFeet is false, convert meters to feet
    else
        // get the feet only as integer by flooring feet:
        let feet
        // get inches as the int remainder of feet divided by 12:
        let inches
        // return the concatenated answer
        return `${feet}ft ${inches}in`;
    }
}
console.log('60 m =', convertDist(60, false)); // 60 meters = 196ft 4in
console.log('111 m =', convertDist(111, false)); // 111 meters = 364ft 4in
```

8. Write a function that finds the hypotenuse (side C) of a right triangle Function has parameters (a,b) representing side A and side B
    - function uses the the Pythagorean Theorem to find c: $a^2 + b^2 = c^2$
    - Function returns c, the hyotenuse (the square root of c^2)
    - test the function on the 3 classic Pythagorean Triples, that is where all 3 sides are integers 3-4-5, 6-8-10, 5-12-13

```
function a, b
    // the hypotenuse is side C in the formula a² + b² = c²
    let c
    return c;
}

console.log(findHypotenuse(3, 4)); // 5
console.log(findHypotenuse(6, 8)); // 10
console.log(findHypotenuse(5, 12)); // 13
```

9. Write a function called addUpCoins() that:
    - takes in numbers of pennies, nickels, dimes and quarters
    - calculates the total value of the coins
    - returns the total as dollars and cents, to two decimal places as a number without a $

```
function add p, n
    let cents
    return


console.log('$', addUpCoinValue(103, 50, 30, 20)); // $ 11.53
// 103 pennies = $1.03; 50 nickels = $2.50, 30 dimes = $3, 20 quarters =
$5
console.log('$', addUpCoinValue(10537, 5650, 6730, 2034)); // $ 1569.37
```

10. Write a function version of the calculate cafe bill from a previous lesson,

- function has four parameters: foodAmt, bevAmt, taxPct and tipPct,
- assign taxPct and tipPct default values of 8.875 and 18, respectiely
- return the answer as an itemized Guest Check (see example, below)
- include the date and time in the Guest Check

Procedure:

- function adds the food and bev together to obtain the sub-total
- tax equals the sub-total times the tax percent.
- tip equals the sub-total times the tip percent.
- total equals the sub-total plus the tax plus the tip.
- Date class must be used for time and date (refer to previous lesson)

```
function food, bev, 8.875 18
    let subTot
    let tipAmt
    let taxAmt
    let total
    const dt
    let date
    let time
    let guestCheck = `
    * ** JavaSlurp Cafe ** *
    * ** Guest Check ** *
    * ** Date: ** *
    * ** Time: ** *
        Food: $
        Bev: $
        Sub-tot: $
        Tax pct: %
        Tax: $
        Tip pct:%
        Tip: $
        Please Pay: $
    * ** Thank you! ** *`;
    return
}

let cafeBill = calcCafeBill(80, 20, 8.875, 15);
console.log(cafeBill);

/* EXPECTED OUTPUT:
* ** JavaSlurp Cafe ** *
    * ** Guest Check ** *
    * ** Monday, Jan 16, 2024 ** *
    * ** 11:34:23 AM ** *
        Food: $80
        Bev: $20
        Sub-tot: $100
        Tax pct: 8.875%
        Tax: $8.87
        Tip pct: 15%
        Tip: $15
        Please Pay: $123.88
    * ** Thank you! ** *;
*/
```

END Lab 03.01

NEXT: Lesson 03.02

03.01 Lab Solution

- Write functions as instructed
- Assume valid input; if the function expects a number argument, provide number
- No function validation need be peformed (e.g. checking the datatype of inputs)

1. The weight of anything on the moon is one-sixth its weight on earth.

- Make a function called calcMoonWeight() that takes in an earth weight number as its argument

- Function calculates the moon weight equivalent, and returns that number.
- So if the input is 180, the output is 30.

```
function calcMoonWeight(earthWeight) {
    return earthWeight / 6;
}

console.log("Earth weight: 180");
console.log("Moon weight:", calcMoonWeight(180));
// Earth weight: 180
// Moon weight: 30
```

2. Write a function that:
    - takes an integer as its input (argument)
    - squares that number (multiplies it by itself)
    - returns
    - so, if you input 4, it returns 16.
    - run the function three times with different inputs.

```
function squareNum(n) {
    return n ** 2;
}

let square = squareNum(4);
console.log('4 sq = ', square); // 4 sq = 16

square = squareNum(5);
console.log('5 sq = ', square); // 5 sq = 25

square = squareNum(6);
console.log('6 sq = ', square); // 6 sq = 36
```

3. Write a function that:
    - takes an integer as its input (argument)
    - if the number is even, it squares the number
    - if the number is odd, it cubes the number
    - the function returns the answer
    - if the input 3, the output is 27 (3 x 3 x 3)
    - if the input 7, the output is 49 (7 x 7) Run the function three times with different values.

```
function squareOrCubeNum(n) {
    // if n divided by 2 yields a remainder of 0, n is even
    if(n % 2 == 0) {
        return n ** 2; // return the even number, squared
    } else { // else, assuming int, n is odd
        return n ** 3; // return the odd number, cubed
    }
    // ternaray alternative to if-else
    // return n % 2 == 0 ? n ** 2 : n ** 3;
}

console.log(squareOrCubeNum(3)); // 27
console.log(squareOrCubeNum(4)); // 16
console.log(squareOrCubeNum(5)); // 125
console.log(squareOrCubeNum(6)); // 36
```

4. Declare a function introducePet() :
    - has four parameters: petType, petName, age, sound
    - there are no default parameter values;
    - the 4 expected arguments must be passed to the function when it's called
    - return a message that includes all four arguments. Example: Meow! My name is Fluffy! I am a 3-year-old cat!.
    - run the function three times, with different pet inputs each time

```
function introducePet(petType, petName, age, sound) {
    return `${sound}! My name is ${petName}! I am a ${age}-year-old
${petType}!`;
}

console.log(introducePet("cat", "Fluffy", 3, "Meow"));
// Meow! My name is Fluffy! I am a 3-year-old cat!

console.log(introducePet("dog", "Fido", 2, "Woof"));
// Woof! My name is Fido! I am a 2-year-old dog!

console.log(introducePet("rabbit", "Bunny", 4, "Crunch"));
// Crunch! My name is Bunny! I am a 4-year-old rabbit!
```

5. Write a function called feetToMeters, which converst feet to meters

    - function takes feet as its input and returns meters
    - round answer to 2 decimal places
    - answer should be a number--not a string

    Conversion formulas: 12 inches = 1 foot 39.37 inches = 1 meter

```
function feetToMeters(feet) {
    let inches = feet * 12; // calc total inches
    let meters = inches / 39.37; // use inches to get meters
    metersRounded = meters.toFixed(2); // round to 2 decimals
    // + sign to convert metersRounded from str back to num
    return +metersRounded;
}
```

```
console.log(`50 ft = ${feetToMeters(50)} m`); // 5 ft' = 15.2 m
console.log(`408 ft = ${feetToMeters(408)} m`); // 408 ft = 124.36 m
```

6. Write a function called convertDistance(), which is similar to feetToMeters() except that it converts both ways: feet-to-meters and meters-to-feet The logic must tell the function which conversion to do. To do this:
    - add a second parameter, a boolean
    - the boolean value should indicate which conversion to do
    - use if-else logic to check the boolean
    - if the boolean is true, convert one way, else go the other way

```
function convertDistance(dist, isFeet) {
    // if isFeet is true, convert feet to meters
    if(isFeet) {
        let meters = dist * 12 / 39.37;
        let roundedMeters = meters.toFixed(2);
        // return +roundedMeters;
        // Or, return all in one line:
        return +((dist * 12 / 39.37).toFixed(2));
    // if isFeet is false, convert meters to feet
    } else {
        // let feet = dist * 39.37 / 12;
        // let roundedFeet = feet.toFixed(2);
        // return +roundedFeet;
        // Or, return all in one line:
        return +((dist * 39.37 / 12).toFixed(2));
    }
}

console.log('500 ft =', convertDistance(500, true), 'm'); // 500 ft =
152.4 m
console.log('150 m =', convertDistance(150, false), 'ft'); // 150 m =
492.13 ft
```

7. Refactor the function to return feet in feet-and-inches format,
   which is a string; use this style: "4ft 8in"
    - Hint: Use the % operator to get inches

```
function convertDist(dist, isFeet) {
    // if isFeet is true, convert feet to meters
    if(isFeet) {
        let meters = dist * 12 / 39.37;
        let roundedMeters = meters.toFixed(2);
        // return +roundedMeters;
        // Or, return all in one line:
        return +((dist * 12 / 39.37).toFixed(2));
    // if isFeet is false, convert meters to feet
    } else {
        // get the feet only as integer by flooring feet:
        let feet = ~~(dist * 39.37 / 12);
        // get inches as the int remainder of feet divided by 12:
        let inches = ~~(feet % 12);
        // return the concatenated answer
        return `${feet}ft ${inches}in`;
    }
}
console.log('60 m =', convertDist(60, false)); // 60 meters = 196ft 4in
console.log('111 m =', convertDist(111, false)); // 111 meters = 364ft 4in
```

8. Write a function that finds the hypotenuse (side C) of a right triangle Function has parameters (a,b) representing side A and side B
    - function uses the the Pythagorean Theorem to find c: $a^2 + b^2 = c^2$
    - Function returns c, the hyotenuse (the square root of c^2)
    - test the function on the 3 classic Pythagorean Triples, that is where all 3 sides are integers 3-4-5, 6-8-10, 5-12-13

```
function findHypotenuse(a, b) {
    // the hypotenuse is side C in the formula a² + b² = c²
    let c = Math.sqrt(a**2 + b**2);
    return c;
}

console.log(findHypotenuse(3, 4)); // 5
console.log(findHypotenuse(6, 8)); // 10
console.log(findHypotenuse(5, 12)); // 13
```

9. Write a function called addUpCoins() that:
    - takes in numbers of pennies, nickels, dimes and quarters
    - calculates the total value of the coins
    - returns the total as dollars and cents, to two decimal places as a number without a $

```
function addUpCoinValue(p, n, d, q) {
    let cents = p + (n * 5) + (d * 10) + (q * 25);
    return +(cents/100);
}

console.log('$', addUpCoinValue(103, 50, 30, 20)); // $ 11.53
// 103 pennies = $1.03; 50 nickels = $2.50, 30 dimes = $3, 20 quarters =
$5
console.log('$', addUpCoinValue(10537, 5650, 6730, 2034)); // $ 1569.37
```

10. Write a function version of the calculate cafe bill from a previous lesson,

- function has four parameters: foodAmt, bevAmt, taxPct and tipPct,
- assign taxPct and tipPct default values of 8.875 and 18, respectiely

- return the answer as an itemized Guest Check (see example, below)
- include the date and time in the Guest Check

Procedure:

- function adds the food and bev together to obtain the sub-total
- tax equals the sub-total times the tax percent.
- tip equals the sub-total times the tip percent.
- total equals the sub-total plus the tax plus the tip.
- Date class must be used for time and date (refer to previous lesson)

```
function calcCafeBill(food, bev, taxPct=8.875, tipPct=18) {
    let subTot = food + bev;
    let tipAmt = subTot * tipPct / 100;
    let taxAmt = subTot * taxPct / 100;
    let total = subTot + tipAmt + taxAmt;
    const dt = new Date();
    let date = dt.toLocaleDateString('en-US', {weekday:'short',
year:'numeric', month:'short', day:'numeric'});
    let time = dt.toLocaleTimeString('en-US');
    let guestCheck = `
    * ** JavaSlurp Cafe ** *
    * ** Guest Check ** *
    * ** ${date} ** *
    * ** ${time} ** *
        Food: $${food}
        Bev: $${bev}
        Sub-tot: $${subTot}
        Tax pct: ${taxPct}%
        Tax: $${taxAmt.toFixed(2)}
        Tip pct: ${tipPct}%
        Tip: $${tipAmt.toFixed(2)}
        Please Pay: $${total.toFixed(2)}
    * ** Thank you! ** *`;
    return guestCheck;
}

let cafeBill = calcCafeBill(80, 20, 8.875, 15);
console.log(cafeBill);

/* EXPECTED OUTPUT:
* ** JavaSlurp Cafe ** *
    * ** Guest Check ** *
    * ** Monday, Jan 16, 2024 ** *
    * ** 11:34:23 AM ** *
        Food: $80
        Bev: $20
        Sub-tot: $100
        Tax pct: 8.875%
        Tax: $8.87
        Tip pct: 15%
        Tip: $15
        Please Pay: $123.88
    * ** Thank you! ** *;
*/
```

END Lab 03.01

NEXT: Lesson 03.02