Hands-on lab on Javascript

The purpose of this lab is to javascript before you set off doing server side coding with Node JS. This lab presumes that you have completed all the other labs in the course IBM HTML CSS and JS for Web development.

Duration (25 mins)

Objective

After completing this lab you will be able to:

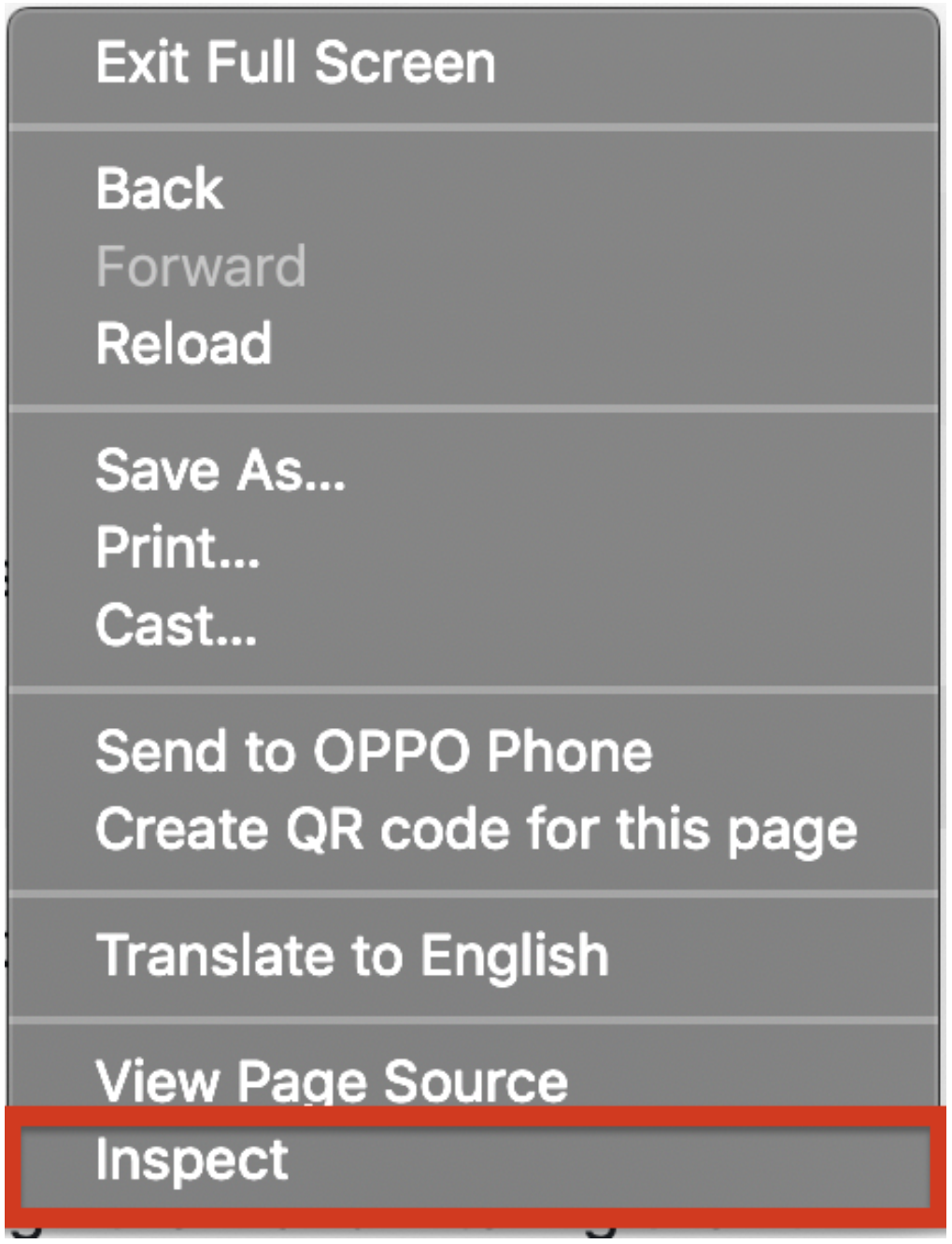
1. Write and run Javascript on the browser console
2. Create variables, work with conditional statements, create loops and define methods in Javascript.

Task 1 - Open the browser console.

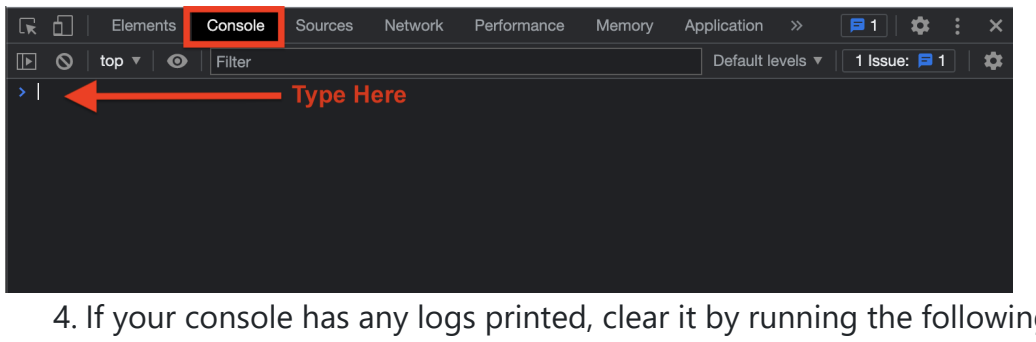
In this task, we are going to run the Javascript code in the browser console. The Chrome browser has a v8, which is Google's open source high-performance JavaScript engine.

1. Open a new blank browser page clicking on Ctrl+T(Windows) or Command+T(Mac) to open a new tab.

2. Right-click anywhere on the new blank browser tab and choose **Inspect** or **Inspect Element** depending on the browser you are using. The image below is for Chrome Broswer.



3. Go to the **Console** tab, as shown below. You will see a command prompt. You can run the rest of the tasks there.



4. If your console has any logs printed, clear it by running the following command. This is not mandatory. It will just help in a fresh start.

```
clear()
```

Task 2 - Running JS commands

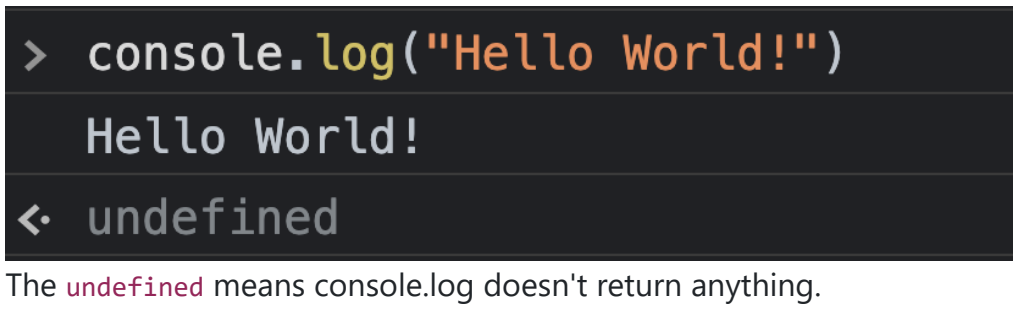
> **NOTE:** At any point of time, when you want to clear the console run the `clear()` command.

To run the commands we will use the command prompt on the browser control. Type or paste the command and press **enter** to run the command.

1. Let's start with a simple code to print **Hello World!** to the console. Run the following command.

```
console.log("Hello World!")
```

The output would be as below.



The `undefined` means console.log doesn't return anything.

2. Let's create some variable and print them. Run the following command.

```
let num = 5
var mystr = "John"
console.log(num)
console.log(mystr)
```

> Both let and var can be used to create variables. `var` is used when you want the variable to have global scope and `let` is used when you want the variable to have scope within the block where it is created.

3. Let's create a constant and print it. Run the following command.

```
const pi_val = 3.147
console.log(pi_val)
```

> Const is used to declare variable whose values can never change

4. Let's create function which prints any value that is input to it.

```
function printMyInput(user_input) {
    console.log("The parameter passed is "+user_input)
}
```

5. Call the function you created in the previous step once with a number and once with a string.

```
printMyInput(9)
printMyInput("John")
```

6. Let's rewrite the function printMyInput according to the ES6 standard. This syntax is also called `arrow functions` and provide a shorthand to write functions.

```
let printMyInputES6 = (user_input)=>{
    console.log(user_input)
}
```

7. Call the function you created in the previous step once with a number and once with a string.

```
printMyInputES6(9)
printMyInputES6("John")
```

> Since the function is passed a single value and the body of the function is a single line, the brackets can be omitted. The code can also be written as below.

```
let printMyInputES6Short = user_input => console.log(user_input)
```

Now when we call it, the output should remain the same.

```
printMyInputES6(9)
printMyInputES6("John")
```

Task3 - Operators, Conditions, Loops

In this task you will be running some javascripts from which you can learn how to use operators, controls and loops.

> Ensure that you understand the code in each file. These are primitive and foundational for your understanding of JS

1. **Arithmetic operators** are operators that we use to perform arithmetic operations.
    - + (plus) operator is used to add

- - (minus) operator is used to subtract)
- * (star or asterisk) operator is used to multiply
- / (slash) operator is used to divide
- ** (double star) operator is used for exponentiation/power
- % (percentage) operator is used for modulus operation

```
console.log("5 + 3 = ",5+3)
console.log("7 - 3 = ",7-3)
console.log("8 * 2 = ",8*2)
console.log("27 / 3 = ",27/3)
console.log("4 power 3 = ",4 ** 3)
console.log("19 mod 4 = ",19%4)
```

2. **Assignment operators** are operators that are used to assign values to variables

- = operator is used to assign value on the right to the variable on left
- += operator is used to increment the value stored in the left operand by the value of the right operand and store it back to the left operand
- -= operator is used to decrement the value stored in the left operand by the value of the right operand and store it back to the left operand
- *= operator is used to multiply the value stored in the left operand by the value of the right operand and store it back to the left operand
- /= operator is used to divide the value stored in the left operand by the value of the right operand and store it back to the left operand
- **= operator is used to raise the value stored in the left operand to the power value of the right operand and store it back to the left operand
- %= operator is used to get modulus of the value stored in the left operand by value of the right operand and store it back to the left operand

```
x = 5
console.log("Old value x ",x)
x += 3
console.log("New value x ",x)

y = 5
console.log("Old value y ",y)
y -= 3
console.log("New value y ",y)
a = 6
console.log("Old value a ",a)
a *= 3
console.log("New value a ",a)

b = 6
console.log("Old value b ",b)
b /= 3
console.log("New value b ",b)

c = 6
console.log("Old value c ",c)
c %= 3
console.log("New value c ",c)

d = 6
console.log("Old value d ",d)
d **= 3
console.log("New value d ",d)
```

3. **Comparison Operators** are used to compare values or variables against values or other variables

- == operator checks if the operand on the left is of equal value to the operand on right
- === operator checks if the operand on the left is of equal value and equal type to the operand on right
- != operator checks if the operand on the left is not of equal value to the operand on right
- > operator checks if the operand on the left is greater than that on the right
- < operator checks if the operand on the left is lesser than that on the right
- >= operator checks if the operand on the left is greater than or equal to that on the right
- <= operator checks if the operand on the left is lesser than or equal to that on the right

```
//Checking equality of 5 number type and 5 string type
console.log("5=='5' ", 5=='5')
console.log("5==='5' ", 5==='5')
console.log("5===5 ", 5===5)

console.log("5 != 5 ", 5 !== 5)
console.log("5 != 6 ", 5 != 6)
console.log("5 != '5' ", 5 !== '5')

console.log("5 > 2 ", 5 > 2)
console.log("5 > 7 ", 5 > 7)
console.log("5 > 5 ", 5 > 5)

console.log("5 < 7 ", 5 < 7)
console.log("5 < 2 ", 5 < 2)
console.log("5 < 5 ", 5 < 5)

console.log("5 >= 5 ", 5 >= 5 )
console.log("5 <= 5 ", 5 <= 5 )
```

4. **Logical Operators** are used to combine more than one conditions.

- && operator checks if the condition on left and right are true. Returns true only of both conditions are true. Else returns false.
- || operators checked if either the condition on the left is true or right is true. Returns true even if one of the two conditions is true.
- ! operator checks if the condition is not met.

Practice exercises for logical operators will be covered along with if-else conditions.

5. **if-else-else if** Conditional statements are very useful to control the flow of your code.

```
//Accept a input from the user. If it is a number print the multiplication table for the number. Else print the input and the
length of the input.

let user_input = prompt('Enter a value');
//check if the user input is not a number
if(isNaN(user_input)) {
    console.log("Your input is ",user_input)
    console.log("The length of your input is ",user_input.length)
} else {
    console.log(user_input, " X 1 = ",user_input*1)
    console.log(user_input, " X 2 = ",user_input*2)
    console.log(user_input, " X 3 = ",user_input*3)
    console.log(user_input, " X 4 = ",user_input*4)
    console.log(user_input, " X 5 = ",user_input*5)
    console.log(user_input, " X 6 = ",user_input*6)
    console.log(user_input, " X 7 = ",user_input*7)
    console.log(user_input, " X 8 = ",user_input*8)
    console.log(user_input, " X 9 = ",user_input*9)
    console.log(user_input, " X 10 = ",user_input*10)
}
```

6. **Loops** can be used when the same block of code is to be executed many times.

for loops have an initial value, condition based on which the loop is executed and an incremental value.

```
//Accept a input from the user. If it is a number print the multiplication table for the number.

let user_input = prompt('Enter a number');
//check if the user input is not a number
if(!isNaN(user_input)) {
    //Using for loop for the repitive statement
    for (let i=0;i<10;i++) {
        console.log(user_input, " X ",i," = ",user_input*i)
    }
}
```

while loops have just a condition based on which a block of code is executed many times.

```
//The code below is to find the length of the words the user is entering. The loop will go on and on until the user chooses not to continue by pressing 'n'

let do_more = true

while(do_more) {
    //Accept a input from the user.
    let user_input = prompt('Enter a word');
    //check if the user input is not a number and then print the length of the input
    if(isNaN(user_input)) {
        console.log("Length of the word you entered is "+user_input.length)
    } else {
        console.log("You entered a number. Only words are allowed")
    }
    let should_continue = prompt("Do you want to continue. Press n to stop")

    if(should_continue === "n") {
        do_more = false
    }
}
```

7. **switch-case** is used to replace multiple if-else if conditions checking the same variable. After one of the conditions is satisfied and the block of code is executed, the control should explicitly **break** out of the switch block.

```
let user_input = prompt('Enter a number between 1 to 7');

//Using logical OR operator to check if the input is a number and it is between 1 to 7
if(isNaN(user_input) || user_input< 1 || user_input>7) {
    console.log("Invalid input")
} else {
    user_input = parseInt(user_input)
    switch(user_input){
        case 1: console.log("Sunday");break;
        case 2: console.log("Monday");break;
        case 3: console.log("Tuesday");break;
        case 4: console.log("Wednesday");break;
        case 5: console.log("Thursday");break;
        case 6: console.log("Friday");break;
        case 7: console.log("Saturday");break;
        default: console.log("Invalid entry");
    }
}
```

Task 4 - Collections

1. Array is an indexed collection. The index positions are from 0. To access the element in first position, we use 0, second position will be 1 and so on. The index of the last position will always be one less than the length of the array.

```
let myArray = ["Jack","Jill",4,5,true,"John"]

console.log(myArray[0]);
console.log(myArray[5]);
```

2. To iterare through arrays there is a special for loop, **forEach**, which executed for each value in

```
let myArray = ["Jack","Jill",4,5,true,"John"]

myArray.forEach(x => {
    console.log(x)
})
```

3. To find the index position and the value, we can use the generic Object.entries method, which can be used with all collection objects. This maps each index position to the value.

```
let myArray = ["Jack","Jill",4,5,true,"John"]
for (const [idx, value] of Object.entries(myArray)) {
  console.log(idx," - ",value);
}
```

4. **Map** object maps a key to a value. The keys have to be unique. The values can be string, int, float or any other valid javascript datatype. An empty Map object can be create with the `new` keyword.

```
let myMap = new Map();

//Here name is key and John is the value.
myMap.set("name","John")

//Here age is the key and 22 is the value.
myMap.set("Age",22)

myMap.forEach((val,key) => {
    console.log(key," - ",val)
})
```

Author(s)

Lavanya

Changelog

Date	Version	Changed by	Change Description
29-Sep-2021	1.0	Lavanya	Created the lab