Tutorial 1

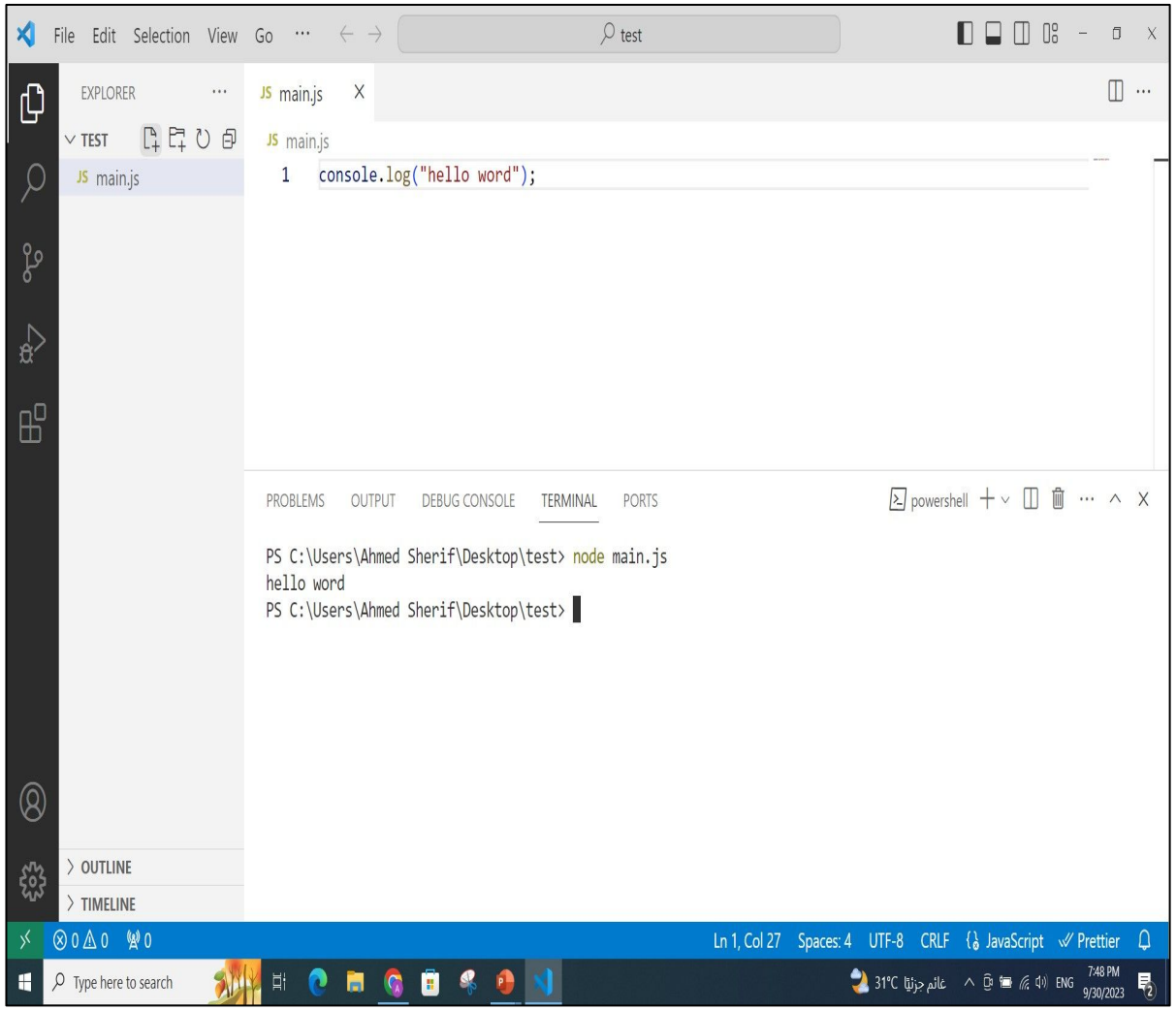
Eng. Ahmed Sherif

Run JavaScript Code

To Run JavaScript Code

- **Download visual studio code**
- **Download and install Node js**

1. **Create new folder Test**
2. **Open visual studio code**
3. **Click on File then open folder**
4. **Select folder Test**
5. **Create new file main.js inside folder Test**
6. **Type console.log("Hello World");**
7. **Open Terminal from view button**
8. **Type node main.js**



JavaScript Data Types

- Javascript is a dynamically typed (no need to explicit write the data type of variable)
- Interpreter assigns variables a type at runtime based on the variable's value.

```
let name = 'Brad'; // string
let fName = "Farid" //string
let mName = `Tawfik`; //string
const age = 37;  // number
const rating = 3.5; // number
const isCool = true; // boolean
var x = null; // null
let y;  // undefined
```

typeof is used to determine the type of variable

```
JS attempt.js > ...
1   let isCool = true;
2   console.log(typeof isCool);

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
boolean
```

number data type includes integers (4)
as well as floating point numbers (3.546)

boolean data type has two values (true or false)

string variable can be declared using backticks (``)
or single quotation ('') or double quotation ("")

undefined data type assigned to variable with no declared value. The value of variable is undefined

Semicolon (;) is optional in JavaScript
You have the freedom to type it or not

let or const or var is used to declare variable
A good practice is to use let or const instead of var

let vs const

const declaration

- **The value of variable can never be changed or updated or even assigned the same value**
- **If the value of variable is reassigned a value then TypeError (Assignment to constant variable) will happen**

```
JS attempt.js > ...
1   const x = 5;
2   x = 6;

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
C:\Users\Ahmed Sherif\Desktop\Tutorials\attempt.js:2
x = 6;
  ^

TypeError: Assignment to constant variable.
```

let declaration

The value of variable can be assigned to any value of same data type or any different data type

```
JS attempt.js > ...
1   let x = 5;
2   x = x + 5;
3   console.log(x);

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
10
```

```
JS attempt.js > ...
1   let x = 5;
2   x = 6;
3   console.log(x);
4   x = "Farid";
5   console.log(x);

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
6
Farid
```

JavaScript Printing Output

console.log() is used to produce output

attempt.js

```
1   console.log("hello world");
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
hello world
PS C:\Users\Ahmed Sherif\Desktop\Tutorials>
```

attempt.js > ...

```
1   const x = "5";
2   console.log("The value of x is" , x);
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The value of x is 5
PS C:\Users\Ahmed Sherif\Desktop\Tutorials>
```

attempt.js

```
1   console.log("hello world" , "welcome Tutorial 13" , "To SE course");
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
hello world welcome Tutorial 13 To SE course
PS C:\Users\Ahmed Sherif\Desktop\Tutorials>
```

attempt.js > x

```
1   const x = "5";
2   console.log("x");
3   console.log(x);
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
x
5
```

Concatenation vs Template literal

console.log() is used to produce output

JS attempt.js > fName

```
1  const fName = "Aly";
2  const age = 45;
3  // Concatenation
4  console.log('My name is ' + fName + ' and I am ' + age);
5  // Template literal (better)
6  console.log(`My name is ${fName} and I am ${age}`);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
My name is Aly and I am 45
My name is Aly and I am 45
```

Template literal syntax : we use backticks (``)
Not single quotation ('') or double quotation ("")
${fName} will add the value of fName (Aly)

JS attempt.js > ...

```
1  const fName = "Aly";
2  console.log(`fname`);
3  console.log(`${fName}`);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
fname
Aly
```

If Condition

Any if condition is evaluated to true or false
If condition is true then it will enter then part

Any if else condition is evaluated to true or false
If condition is true then it will enter then part
If condition is false then it will enter else part

And is represented using &&

attempt.js > ...

```
1  const x = 7;
2  if(x >= 5 && x <= 10){
3      console.log("x is between 5 and 10")
4  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
x is between 5 and 10
```

attempt.js > ...

```
1  const x = 10;
2  if(x > 5){
3      console.log('The value of x is greater than 5');
4  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The value of x is greater than 5
```

attempt.js > ...

```
1  const x = 5;
2  if(x > 5){
3      console.log('The value of x is greater than 5');
4  }else{
5      console.log('The value of x is less than or equal to 5');
6  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The value of x is less than or equal to 5
```

nested if else condition

Any if condition is evaluated to true or false
If condition is true then it will enter then part

Any if else condition is evaluated to true or false
If condition is true then it will enter then part
If condition is false then it will enter else part

|| : or condition

JS attempt.js > ...

```
1   const x = 7;
2   if(x == 7 || x == 8){
3       console.log("x is 7 or 8");
4   }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
x is 7 or 8
```

```
const x = 5;
if(x >= 5 && x <= 10){
    if(x >= 8){
        console.log('The value of x is 8 or 9 or 10');
    }else{
        console.log('The value of x is 5 or 6 or 7');
    }
}else{
    if(x == 3 || x == 4){
        console.log('The value of x is 3 or 4');
    }else{
        console.log('The value of x is less than 3 or greater than 10');
    }
}
```

Switch Statement

We can use switch statement instead of several if else statements

Don't forget to add break keyword

JS attempt.js

```
1  color = 'blue';
2  switch(color) {
3    case 'red': console.log('color is red'); break;
4    case 'blue': console.log('color is blue'); break;
5    case 'green': console.log('color is green'); break;
6    default: console.log('color is not red or green or blue');
7  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
color is blue
```

JS attempt.js

```
1  color = 'blue';
2  if(color == 'red'){
3      console.log('color is red');
4  }else{
5      if(color == 'blue'){
6          console.log('color is blue');
7      }else{
8          if(color == 'green'){
9              console.log('color is green');
10         }else{
11             console.log('color is not red or green or blue');
12         }
13     }
14 }
```

Ternary Operator

We can use it as short hand for if else condition

JS attempt.js > ...

```
1   // Ternary operator / Shorthand if
2   const color = 'red';
3   const z = color === 'red' ? 10 : 20;
4   console.log(z)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
10
```

JS attempt.js > color

```
1   // Ternary operator / Shorthand if
2   const color = 'blue';
3   const z = color === 'red' ? 10 : 20;
4   console.log(z)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
20
```

JS attempt.js > ...

```
1   const color = 'blue';
2   let z;
3   if(color === 'red'){
4       z = 10;
5   }else{
6       z  = 20;
7   }
8   console.log(z);
```

Ternary Operator

const z = color === 'red' ? 50 : 60;

If condition (color === 'red') is true then z is assigned to 50 otherwise 60

Double Equal vs Triple Equal

Double Equal (==) checks the values of variables regardless its data type and return Boolean

```
JS attempt.js > ...
  1    const a = "5"; // string
  2    const b = 5; // number
  3    console.log(a == b);

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
true
```

Triple Equal (===) compares the datatype and values of variables and return Boolean

```
JS attempt.js > ...
  1    const a = "5"; // string
  2    const b = 5; // number
  3    console.log(a === b);

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
false
```

Javascript Truthy Falsy

- If any of the following was used in a boolean expression it would yield a true result:

`'0'` (a string containing a single zero)
`'false'` (a string containing the text “false”)
`[]` (an empty array)
`{}` (an empty object)
`function(){}` (an “empty” function)

Try to run this code and see the output

```
1   const x = '0';
2   if(x){
3       console.log("Then Part");
4   }else{
5       console.log("Else Part");
6   }
```

Javascript Truthy Falsy

- If any of the following was used in a boolean expression it would yield a false result:

`false`
`0` (zero)
`-0` (minus zero)
`''`, `""`, `` `` (empty string)
`null`
`undefined`
`NaN`

Try to run this code and see the output

```
1   const x = '';
2   if(x){
3       console.log("Then Part");
4   }else{
5       console.log("Else Part");
6   }
```

String Operations

Any String can represented as array of characters
Where first character starts at position 0

attempt.js > ...

```
1   const s = 'Hello World';
2   // Get length of string
3   let val = s.length;
4   console.log("The length of string s is",val);
5   // get character at specific index
6   let i = 0;
7   console.log(`The character at index ${i} is ${s[i]}`);
8   i = 4;
9   console.log(`The character at index ${i} is ${s[i]}`);
10  i = s.length - 1;
11  console.log(`The character at index ${i} is ${s[i]}`);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The length of string s is 11
The character at index 0 is H
The character at index 4 is o
The character at index 10 is d
```

substring(start , end) : extract characters starting from start index until end index – 1
substring(0 , 4) : extract characters between index 0 and 3 (inclusive)

attempt.js > ...

```
1   // String methods & properties
2   const s = 'Hello World';
3   // Get sub string
4   let val = s.substring(0 , 4);
5   console.log(val);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
Hell
```

attempt.js > ...

```
1   const s = 'Hello World';
2   let val = s.substring(0 , 2);
3   console.log(val);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
He
```

String Operations

attempt.js > s

```
1  // String methods & properties
2  const s = 'Hello,World,1';
3  // Get sub string
4  let val = s.split(",");
5  console.log(val);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
[ 'Hello', 'World', '1' ]
```

attempt.js > val

```
1  // String methods & properties
2  const s = 'Hello#World#1';
3  // Get sub string
4  let val = s.split("#");
5  console.log(val);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
[ 'Hello', 'World', '1' ]
```

attempt.js > ...

```
1  const s = 'Hello World';
2  // Change case
3  let result = s.toUpperCase();
4  console.log("The uppercase output : " + result);
5  result = s.toLowerCase();
6  console.log(`The lowercase output : ${result}`);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The uppercase output : HELLO WORLD
The lowercase output : hello world
```

attempt.js > ...

```
1  let strExample  = "Hello world 2";
2  const removedCharacter = "2";
3  let result = "";
4  for(let i = 0; i < strExample.length ; i++){
5      if(strExample[i] != removedCharacter){
6          result += strExample[i];
7      }
8  }
9  console.log("The result is",result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The result is Hello world
```

Array

JS attempt.js > ...

```
1   // ARRAYS - Store multiple values in a variable
2   const numbers = [1,2,3,4,5];
3   const fruits = ['apples', 'oranges', 'pears', 'grapes'];
4   console.log("numbers array",numbers);
5   // get length of array
6   console.log("size of numbers array",numbers.length);
7   // Get one value - Arrays start at 0
8   console.log("fruits : ",fruits[1]);
9   // add new element in the array;
10  fruits.push('strawberries');
11  console.log("fruits array",fruits);
12  // Get index
13  console.log(fruits.indexOf('oranges'));
14  console.log(fruits.indexOf('mango'));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
numbers array [ 1, 2, 3, 4, 5 ]
size of numbers array 5
fruits :  oranges
fruits array [ 'apples', 'oranges', 'pears', 'grapes', 'strawberries' ]
1
-1
```

JS attempt.js > ...

```
1   const fruits = ['apples', 'oranges', 'pears', 'grapes'];
2   console.log(fruits.includes('strawberries'));
3   console.log(fruits.includes('oranges'));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
false
true
```

JS attempt.js > ...

```
1   const diffArray = [1 , "mango" , true];
2   console.log(diffArray);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
[ 1, 'mango', true ]
```

Array

This code multiply the value of each element in array by factor of 3

```
JS attempt.js > ...
1   const numbers  = [1,2,3,4]
2   for(let  i = 0; i < numbers.length ; i++){
3       numbers[i] = numbers[i] * 3;
4   }
5   console.log(numbers);
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
[ 3, 6, 9, 12 ]
```

How to update the value of element in a given array

```
JS attempt.js > ...
1   const numbers  = [1,2,3,4]
2   numbers[0] = 10;
3   console.log(numbers);
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
[ 10, 2, 3, 4 ]
```

The first element of Array starts at position 0

Array

This code remove an element from given array and returns new array

attempt.js > ...

```
1   const numbers  = [1,2,3,4]
2   const removeNumber = 3;
3   const result = [];
4   for(let i = 0; i < numbers.length ; i++){
5       if(numbers[i] != removeNumber){
6           result.push(numbers[i]);
7       }
8   }
9   console.log("result array", result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

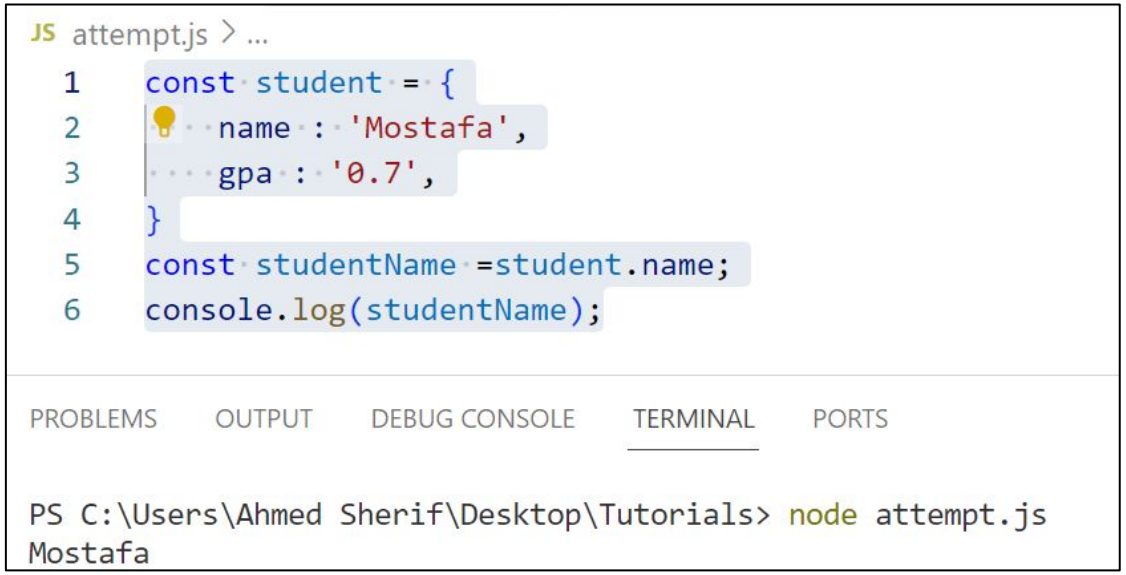
```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
result array [ 1, 2, 4 ]
```

Object

Object is data type where it has properties and values
Object variable can have any unlimited number of properties

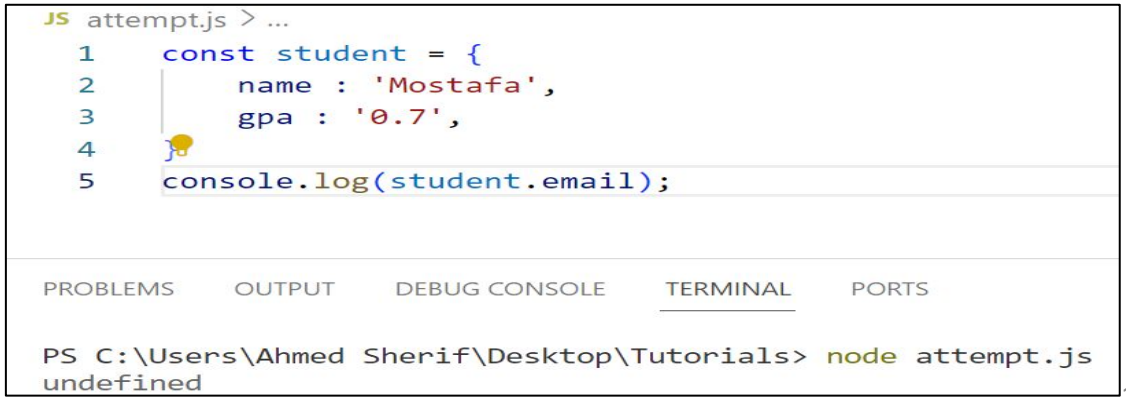
For example

```
const student = {
   name : 'Mostafa',
   gpa : '0.7',
}
```



student variable is an object data type and it has properties such as name and gpa
The name property has value of Mostafa
The gpa property has value of 0.7

If we are trying to access property that doesn't exist in the object then it will give us undefined not an error



Object

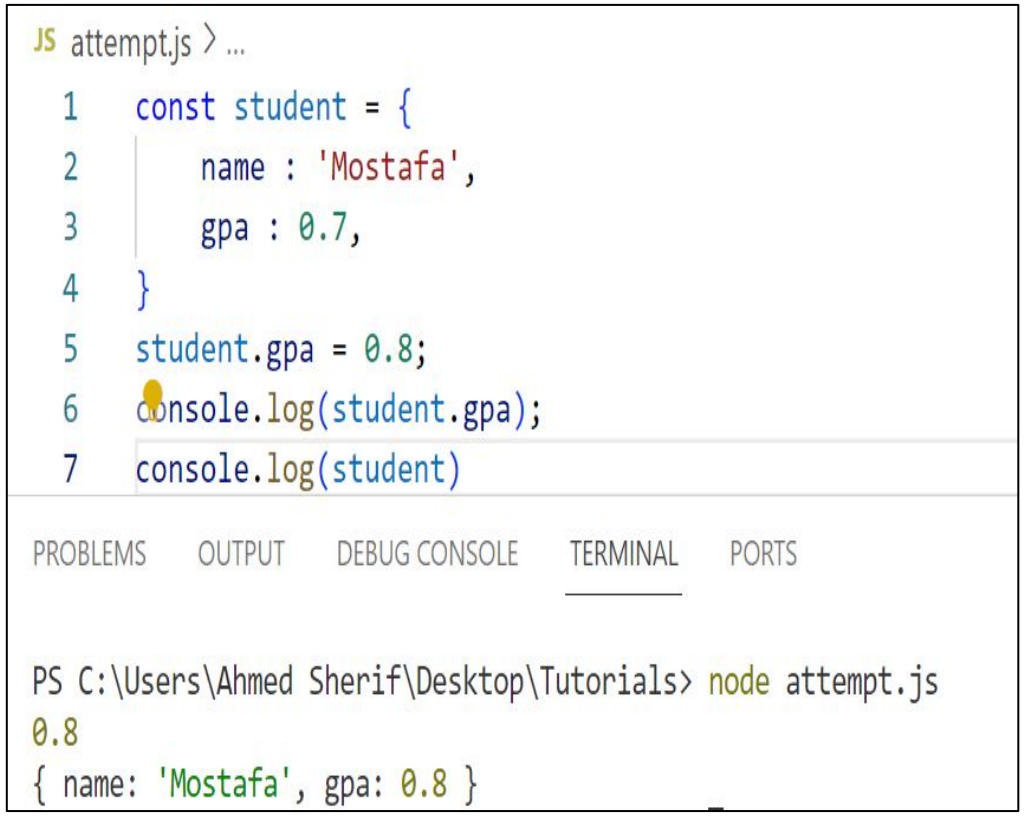
Object is data type where it has properties and values
Object variable can have any unlimited number of properties

For example

```
const student = {
   name : 'Mostafa',
   gpa : 0.7,
}
```

I want to update the gpa of student variable from 0.7 to 0.8

student.gpa = 0.8;



Object

JS attempt.js > ...

```
1   const person = {
2       firstName: 'John',
3       age: 30,
4       hobbies: ['music', 'movies', 'sports'],
5       address: {
6         street: '50 Main st',
7         city: 'Boston',
8         state: 'MA'
9       }
10    }
11  // Get single value
12  console.log(person.firstName)
13  // Get array value
14  console.log(person.hobbies[1]);
15  //Get embedded object
16  console.log(person.address.city);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
John
movies
Boston
```

JS attempt.js > person

```
1   const person = {
2       firstName: 'John',
3       age: 30,
4       hobbies: ['music', 'movies', 'sports'],
5       address: {
6         street: '50 Main st',
7         city: 'Boston',
8         state: 'MA'
9       }
10    }
11  // Add property
12  person.email = 'jdoe@gmail.com';
13  console.log("person object",person)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
person object {
  firstName: 'John',
  age: 30,
  hobbies: [ 'music', 'movies', 'sports' ],
  address: { street: '50 Main st', city: 'Boston', state: 'MA' },
  email: 'jdoe@gmail.com'
}
```

Spread Operator (. . .)

In line 5, spread operator (...) took all properties of student variable and added them to s1 object

attempt.js > ...

```
1   const student = {
2       name : 'Mostafa',
3       gpa : 0.7,
4   }
5   const s1 = {...student};
6   console.log(s1);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
{ name: 'Mostafa', gpa: 0.7 }
```

In line 5, spread operator (...) took all properties of student variable and added them to s1 object and

We add new property courses to s1 object

attempt.js > ...

```
1   const student = {
2       name : 'Mostafa',
3       gpa : 0.7,
4   }
5   const s1 = {...student , courses : ["Se-I" , "SE-II"
6   console.log(s1);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
{ name: 'Mostafa', gpa: 0.7, courses: [ 'Se-I', 'SE-II' ] }
```

Spread Operator (. . .)

JS attempt.js > s1 > gpa

```
1   const student = {
2       name : 'Mostafa',
3       gpa : 0.7,
4   }
5   const s1 = {...student , gpa : 0.8};
6   console.log(s1);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
{ name: 'Mostafa', gpa: 0.8 }
```

s1 = {name : 'Mostafa', gpa : 0.7, gpa : 0.8};
s1 = {name : 'Mostafa', gpa : 0.8};

JS attempt.js > student

```
1   const student = {
2       name : 'Mostafa',
3       gpa : 2.1,
4       gpa : 1.8
5   }
6   console.log(student);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
{ name: 'Mostafa', gpa: 1.8 }
```

Spread Operator (. . .)

```
const course = { university: 'GIU', title: 'Software Engineering };

const newCourse { …course, title: 'Software Engineering II };

console.log(newCourse);
```

Spread Operator (. . .)

```
const course = { university: 'GIU', title: 'Software Engineering };

const newCourse { …course, title: 'Software Engineering II };

console.log(newCourse); // { university: '???', title: '????'}
```

Spread Operator (. . .)

```
const course = { university: 'GIU', title: 'Software Engineering };

const newCourse { ...course, title: 'Software Engineering II };

console.log(numbers); // { university: 'GIU', title: 'Software Engineering II }
```

ECMAScript

- “ECMAScript is a JavaScript standard meant to ensure the interoperability of web pages across different web browsers. It is standardized by Ecma International according to the document ECMA-262”

- ECMAScript 2015 was the second major revision to JavaScript.
- ECMAScript 2015 is also known as ES6 and ECMAScript 6.

- Advantages:
  - Brings new syntax and great features to make your code more modern and readable. It allows the developer to write less code and do more. ES6 introduces many great features like arrow functions, template strings, class destruction, Modules… and more.

https://en.wikipedia.org/wiki/ECMAScript

Object Destructuring

- destructuring syntax provides an alternative way to assign properties of an object to variables:

```
const person = {
    firstName: 'John',
    lastName: 'Doe'
};
```

Before ES6, to assign properties of 'person' to a variable:

```
var fName = person.firstName;
var lName = person.lastName;
```

With ES6, can do:

```
let { firstName: fname, lastName: lname } = person;
```

If variable has same name as object property, it becomes even cleaner:

```
let { firstName, lastName } = person;
console.log(firstName); // 'John'
console.log(lastName); // 'Doe'
```

Object Destructuring

attempt.js

```
1  // Destructuring Objects & Array
2  const person1 = { p_name : "Mostafa", p_age : "25"}
3  const { p_name : fName , p_age : myAge } = person1
4  // instead of const fName = person.p_name
5  // instead of const myAge = person.p_age
6  console.log(fName , myAge);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
Mostafa 25
```

attempt.js

```
2  const person1 = { p_name : "Mostafa", p_age : "25"
3  const { p_name : p_name , p_age : p_age } = person1
4  // instead of const p_name = person.p_name
5  // instead of const p_age = person.p_age
6  console.log(p_name , p_age);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
Mostafa 25
```

attempt.js

```
1  const person1 = { p_name : "Mostafa", p_age : "25"}
2  const { p_name , p_age } = person1
3  // instead of const p_name = person.p_name
4  // instead of const p_age = person.p_age
5  console.log(p_name , p_age);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
Mostafa 25
```

Template Literals

- Template literals are literals delimited with backticks (`), allowing embedded expressions called substitutions.

  Assuming the following:

```
const { firstName, lastName } = person;
```

- To concatenate those two variables, we typically had to do something along these lines:

```
const fullName = firstName + ' ' + lastName;
```

- With Template Literals, we can create a template string using backtick symbols (`) and substituting expressions inside of ${} placeholders

```
const fullName = `${firstName} ${lastName}`;
```

Array Destructuring

- ES6 provides a new feature called destructing assignment that allows you to destructure properties of an object or elements of an array into individual variables:

```
function getScores() {
   return [70, 80, 90];
}
```

```
let scores = getScores();
let x = scores[0],
    y = scores[1],
    z = scores[2];
```

- Using ES6 it becomes much easier:

```
let [x, y, z] = getScores();
```

```
console.log(x); // 70
console.log(y); // 80
console.log(z); // 90
```

Array Destructuring

JS attempt.js > ...

```
1   const dateArray = "06:31:2012".split(":")
2   const [month, day, year] = dateArray;
3   // instead of const month = dateArray[0]
4   // instead of const day = dateArray[1]
5   // instead of const year = dateArray[2]
6   const newDate = `${day}-${month}-${year}`;
7   console.log(newDate);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
31-06-2012
```

Array of Objects

JS attempt.js > todos

```
1   // Array of objects
2   const todos = [
3       {
4         id: 1,
5         text: 'Take out trash',
6         isComplete: false
7       },
8       {
9         id: 2,
10        text: 'Dinner with wife',
11        isComplete: false
12      },
13      {
14        id: 3,
15        text: 'Meeting with boss',
16        isComplete: true
17      }
18  ];
```

JS attempt.js > ...

```
21  for(let i = 0; i < todos.length; i++){
22      let todoElementObj = todos[i]
23      console.log(`${todoElementObj.text}`);
24  }
25
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
Take out trash
Dinner with wife
Meeting with boss
```

JS attempt.js > ...

```
20  for(let todoElementObj of todos) {
21      console.log(`${todoElementObj.text}`);
22  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
Take out trash
Dinner with wife
Meeting with boss
```

For vs While loop

JS attempt.js > ...

```
1  let sum=0;
2  for(let i = 0; i <= 10; i++){
3    //console.log(`For Loop Number: ${i}`);
4    sum=sum+i
5  }
6  console.log("The sum of for loop is",sum)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The sum of for loop is 55
```

JS attempt.js > ...

```
1  let i = 0
2  let sum = 0;
3  while(i <= 10) {
4      sum += i;
5      i++;
6  }
7  console.log("The sum of while loop is",sum);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
The sum of while loop is 55
```

Functions

JS attempt.js > ...

```
1   function sum(x,y){
2       return x+y;
3   }
4   const a = 5;
5   const b = 6;
6   const result = sum(a,b);
7   console.log(result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
11
```

The above left image returns an integer

JS attempt.js > ...

```
1   function sum(x,y){
2       const z = x+y;
3       console.log("result is",z);
4   }
5   const a = 5;
6   const b = 6;
7   sum(a,b);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
result is 11
PS C:\Users\Ahmed Sherif\Desktop\Tutorials>
```

The above Right image doesn't return an output
Just printing the output

Arrow Functions

JS attempt.js > ...

```
1   const sum = (x , y) => {
2       return x+y;
3   }
4   const a = 5;
5   const b = 6;
6   const result = sum(a,b);
7   console.log(result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
11
```

The above left image is an example of arrow function
The function name after const (sum)
After = we got arguments of function (x , y)
After Fat Arrow (=>) , we got the logic of function

JS attempt.js > ...

```
1   function sum(x,y){
2       return x+y;
3   }
4   const a = 5;
5   const b = 6;
6   const result = sum(a,b);
7   console.log(result);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Ahmed Sherif\Desktop\Tutorials> node attempt.js
11
```

The above Right image is an example of conventional function

Run JavaScript Code

To Run JavaScript Code

- **Download visual studio code**
- **Download and install Node js**

1. **Create new folder Test**
2. **Open visual studio code**
3. **Click on File then open folder**
4. **Select folder Test**
5. **Create new file main.js inside folder Test**
6. **Type console.log("Hello World");**
7. **Open Terminal from view button**
8. **Type node main.js**