

Clarusway



Backend Workshop/ Teamwork -1-

Workshop

Subject: OOP and Nodejs

Learning Goals

- Having knowledge about backend and nodejs.
- Understand how real-life entities/objects can be transferred to the computer environment.

Introduction

As developers, we should also be able to express what we know. This study was prepared to support this purpose. Enjoy your work.

Practice Using the IDE in Lesson

We will use the VSCode you are familiar with. At the same time, we need to install Nodejs on our computer. Lets start

1. In the table below specifies the key differences between a front-end and back-end development. Indicate at the beginning of each item whether it is suitable for the front end or the back end.

.....is refers to the client-side of an application.

.....includes everything that attributes to the visual aspects of a web application.

.....refers to the server-side of an application.

..... technologies are HTML, CSS, Bootstrap, jQuery, JavaScript, AngularJS, and React.js.

.....generally includes a web server that communicates with the database to serve the users' requests.

..... some framework examples are AngularJS, React.js, jQuery, Sass, etc.

.....consists of everything that happens behind the scenes and users cannot see and interact with.

.....is the part of a web application where users can see and interact.

.....technologies are Java, PHP, Python, C++, Node.js, etc.

.....some framework examples are Express, Django, Rails, Laravel, Spring, etc.

Answer:

Frontend is refers to the client-side of an application.

Frontend includes everything that attributes to the visual aspects of a web application.

Backend refers to the server-side of an application.

Frontend technologies are HTML, CSS, Bootstrap, jQuery, JavaScript, AngularJS, and React.js.

Backend generally includes a web server that communicates with the database to serve the users' requests.

Frontend some framework examples are AngularJS, React.js, jQuery, Sass, etc.

Backend consists of everything that happens behind the scenes and users cannot see and interact with.

Frontend is the part of a web application where users can see and interact.

Backend technologies are Java, PHP, Python, C++, Node.js, etc.

Backend some framework examples are Express, Django, Rails, Laravel, Spring, etc.

2. What is Nodejs? What can we do with Nodejs? Why use Nodejs?

Answer:

What; Node.js is a versatile runtime environment that allows developers to build fast and scalable network applications.

Why; The answer is simple – it's exceptional for data-intensive real-time applications, APIs, microservices, and streaming applications.

What can we do;

- Weal-time web applications
- Network applications
- Distributed systems

- General purpose applications

3. Lets try to write code axplanid above

- Define a class named "person".
- Let this class have a constructor function that takes a "name" parameter,
- I can learn the name of the object with the function called "sayName".
- Define a class named "BankAccount" derived from the Person class.
- Let it have a private property named "balance".
- Let's have a constructor that takes "name" and "balance" parameters.
- Money can be added to the account with a method called "deposit".
- Money can be withdrawn from the account with a method called "withdraw".
- I can find out the total amount with the function called "balance".

Answer:

```
class Person {
  constructor(name) {
    this.name = name;
  }

  introduceSelf() {
    console.log(`Hello! I'm ${this.name}`);
  }
}

class BankAccount extends Person {
  _balance=0
  constructor(name, balance) {
    super(name)
    this._balance = balance;
  }
  deposit(amount) {
    this._balance += amount;
  }
  withdraw(amount) {
    if (amount > this._balance) {
      console.log("Funds are insufficient");
    } else {
      this._balance -= amount;
    }
  }
  get balance() {
    return this._balance;
  }
}

const account = new BankAccount('tom', '1800');
console.log(account._balance);
account.withdraw(500);
console.log(account._balance);
```

```
account.withdraw(1000);  
console.log(account._balance);
```

Teamwork

Subject: SQL

Learning Goals

- To be able to write SQL statements that will perform the desired query.

Introduction

We use the SQL language when performing operations on relational databases. You can perform many different operations on the DB with SQL, but this work only includes querying.

Lets start

Write SQL statements that produce the desired output.

1. WRITE A QUERY THAT RETURNS TRACK NAME AND ITS COMPOSER FROM TRACKS TABLE

```
SELECT Name, Composer FROM tracks;
```

2. WRITE A QUERY THAT RETURNS ALL COLUMNS FROM TRACKS TABLE

```
SELECT * FROM tracks;
```

3. WRITE A QUERY THAT RETURNS THE UNIQUE NAME OF COMPOSERS OF EACH TRACK

```
SELECT DISTINCT Composer FROM tracks;
```

4. WRITE A QUERY THAT RETURNS UNIQUE ALBUMID, MEDIATYPEID FROM TRACKS TABLE

```
SELECT DISTINCT AlbumId, MediaTypeId FROM tracks;
```

5. WRITE A QUERY THAT RETURNS TRACK NAME AND TRACKID OF 'Jorge Ben'

```
SELECT Name, TrackId
FROM tracks
WHERE Composer = 'Jorge Ben';
```

6. WRITE A QUERY THAT RETURNS ALL INFO OF THE INVOICES OF WHICH TOTAL AMOUNT IS GREATER THAN \$25

```
SELECT *
FROM invoices
WHERE Total > 25;
```

7. WRITE A QUERY THAT RETURNS ALL INFO OF THE INVOICES OF WHICH TOTAL AMOUNT IS LESS THAN \$15. JUST RETURN 5 ROWS

```
SELECT *
FROM invoices
WHERE Total < 15
LIMIT 5;
```

8. WRITE A QUERY THAT RETURNS ALL INFO OF THE INVOICES OF WHICH TOTAL AMOUNT IS GREATER THAN \$10. THEN SORT THE TOTAL AMOUNTS IN DESCENDING ORDER, LASTLY DISPLAY TOP 2 ROWS

```
SELECT *
FROM invoices
WHERE Total > 10
ORDER BY Total DESC
LIMIT 2;
```

9. WRITE A QUERY THAT RETURNS ALL INFO OF THE INVOICES OF WHICH BILLING COUNTRY IS NOT CANADA. THEN SORT THE TOTAL AMOUNTS IN ASCENDING ORDER, LASTLY DISPLAY TOP 10 ROWS

```
SELECT *
FROM invoices
WHERE NOT BillingCountry = 'CANADA'
ORDER BY Total ASC
LIMIT 10;
```

10. WRITE A QUERY THAT RETURNS INVOICEID, CUSTOMERID AND TOTAL DOLLAR AMOUNT FOR EACH INVOICE. THEN SORT THEM FIRST BY CUSTOMERID IN ASCENDING, THEN TOTAL DOLLAR AMOUNT IN DESCENDING ORDER.

```
SELECT InvoiceId, CustomerId, Total
FROM invoices
ORDER BY CustomerId, Total DESC;
```

11. WRITE A QUERY THAT RETURNS ALL TRACK NAMES THAT START WITH 'B' AND END WITH 'S'

```
SELECT Name
FROM tracks
WHERE Name LIKE 'B%' AND Name LIKE '%s';
(ALTERNATIVE -- WHERE name LIKE 'B%s');
```

12. WRITE A QUERY THAT RETURNS THE NEWEST DATE AMONG THE INVOICE DATES BETWEEN 2008 AND 2011

```
SELECT InvoiceDate
FROM invoices
WHERE InvoiceDate BETWEEN '2008-01-01' AND '2012-01-01'
ORDER BY InvoiceDate DESC
LIMIT 1;
```

13. WRITE A QUERY THAT RETURNS THE FIRST AND LAST NAME OF THE CUSTOMERS WHO HAVE ORDERS FROM NORWAY AND BELGIUM

```
SELECT FirstName, LastName
FROM customers
WHERE Country IN ('Belgium', 'Norway')
```

14. WRITE A QUERY THAT RETURNS THE TRACK NAMES OF 'ZAPPA'

```
SELECT Composer, Name
FROM tracks
WHERE Composer LIKE '%Zappa';
```

15. HOW MANY TRACKS AND INVOICES ARE THERE IN THE DIGITAL MUSIC

```
STORE, DISPLAY SEPERATELY
SELECT COUNT(*)
```

```
FROM tracks;  
SELECT COUNT(*)  
FROM invoices;
```

16. HOW MANY COMPOSERS ARE THERE IN THE DIGITAL MUSIC STORE

```
SELECT COUNT(DISTINCT Composer)  
FROM tracks;
```

17. HOW MANY TRACKS DOES EACH ALBUM HAVE, DISPLAY ALBUMID AND NUMBER OF TRACKS SORTED FROM HIGHEST TO LOWEST

```
SELECT AlbumId,  
COUNT(*) AS number_of_tracks  
FROM tracks  
GROUP BY AlbumId  
ORDER BY number_of_tracks DESC;
```

18. WRITE A QUERY THAT RETURNS TRACK NAME HAVING THE MINIMUM AND MAXIMUM DURATION, DISPLAY SEPERATELY

```
SELECT Name, MIN(Milliseconds) AS Min, MAX(Milliseconds) as Max  
FROM tracks;
```

19. WRITE A QUERY THAT RETURNS THE TRACKS HAVING DURATION LESS THAN THE AVERAGE DURATION

```
SELECT *  
FROM tracks  
WHERE Milliseconds < 393599.212103911  
  
SELECT *  
FROM tracks  
WHERE Milliseconds < (  
SELECT AVG(Milliseconds)  
FROM tracks);
```

20. WRITE A QUERY THAT RETURNS THE TOTAL NUMBER OF EACH COMPOSER's TRACK.

```
SELECT Composer, COUNT(*)  
FROM tracks  
GROUP BY Composer;  
  
SELECT Composer, COUNT(Composer)
```

```
FROM tracks
GROUP BY Composer;

SELECT Composer, COUNT(Composer)
FROM tracks
WHERE Composer IS NOT NULL
GROUP BY Composer;
```

21. WRITE A QUERY THAT RETURNS THE GENRE OF EACH TRACK.

```
SELECT tracks.Name, genres.Name
FROM tracks
JOIN genres
ON tracks.GenreId = genres.GenreId;

SELECT t.Name, g.Name
FROM tracks t
JOIN genres g
ON t.GenreId = g.GenreId;
```

22. WRITE A QUERY THAT RETURNS THE ARTIST'S ALBUM INFO.

```
SELECT *
FROM artists
LEFT JOIN albums
ON albums.ArtistId = artists.ArtistId
```

23. WRITE A QUERY THAT RETURNS THE MINIMUM DURATION OF THE TRACK IN EACH ALBUM. DISPLAY ALBUMID, ALBUM TITLE AND DURATION OF THE TRACK. THEN SORT THEM FROM HIGHEST TO LOWEST

```
SELECT tracks.AlbumId, albums.Title,
MIN(tracks.Milliseconds) AS min_duration
FROM tracks
JOIN albums
ON tracks.AlbumId = albums.AlbumId
GROUP BY tracks.AlbumId, albums.Title
ORDER BY min_duration DESC;
```

24. WRITE A QUERY THAT RETURNS ALBUMS WHOSE TOTAL DURATION IS HIGHER THAN 60 MIN. DISPLAY ALBUM TITLE AND THEIR DURATIONS. THEN SORT THE RESULT FROM HIGHEST TO LOWEST

```
SELECT albums.Title, SUM(tracks.Milliseconds) AS total_duration
FROM tracks
```



```
JOIN albums ON tracks.AlbumId = albums.AlbumId
GROUP BY tracks.AlbumId
HAVING total_duration > 3600000
ORDER BY total_duration DESC;
```

25. WRITE A QUERY THAT RETURNS TRACKID, TRACK NAME AND ALBUMID INFO OF THE ALBUM WHOSE TITLE ARE 'Prenda Minha', 'Heart of the Night' AND 'Out Of Exile'.

```
SELECT Trackid, Name, Albumid
FROM tracks
WHERE albumid IN (
  SELECT AlbumId
  FROM albums
  WHERE Title IN ('Prenda Minha', 'Heart of the Night', 'Out Of Exile'));
```

😊 Thanks for Attending 🙌

Clarusway

