# Midterm Review - Questions & Answers

## Difference between JavaScript and Node.js

JavaScript is a scripting language that runs in browsers. Node.js is a runtime environment that allows JavaScript to run outside browsers using the V8 engine.

## How to run a JavaScript file in Node.js

Use the terminal:  
```bash  
node filename.js  
```

## What is npm and its purpose?

npm (Node Package Manager) is used to manage JavaScript packages. It helps install, update, and manage dependencies.

## How are variables typed in JavaScript?

JavaScript uses dynamic typing, meaning a variable can hold different data types.

## Operator to determine type

The `typeof` operator:  
```js  
console.log(typeof "hello"); // "string"  
```

## JavaScript Data Types & Their Byte Sizes

- Number (8 bytes)  
- Boolean (4 bytes)  
- Null (special object, minimal memory)  
- String (2 bytes per character)

## String Functions & Properties

- `length` → Gets the number of characters in a string  
- `toUpperCase()` / `toLowerCase()` → Changes case  
- `charAt(index)` → Gets character at index  
- `replace('old', 'new')` → Replaces text  
- `split('separator')` → Splits into an array

## Looping Through a String

```js  
let str = "hello";  
for (let i = 0; i < str.length; i++) {  
 console.log(str[i]);  
}  
```

## Arrays - Functions & Properties

- `push()` → Adds element  
- `pop()` → Removes last element  
- `concat()` → Merges arrays  
- `length` → Returns array size  
- Loop through an array:  
```js  
arr.forEach(item => console.log(item));  
```

## Creating a JSON object

```js  
let person = { "name": "John", "age": 30 };  
```

## Accessing keys in a JSON object

```js  
console.log(person.name);  
```

## Command Line Arguments (`process.argv`)

```js  
console.log(process.argv[2]); // Gets 3rd argument  
```

## Difference Between `==` and `===`

- `==` → Loose equality (checks value, ignores type)  
- `===` → Strict equality (checks value & type)

## Writing a Simple Function

```js  
function add(a, b) { return a + b; }  
```

## Anonymous Functions

```js  
let greet = function(name) { return "Hello, " + name; };  
```

## Passing Anonymous Functions

```js  
setTimeout(function() { console.log("Delayed"); }, 1000);  
```

## forEach Loop

```js  
arr.forEach((item, index, array) => console.log(item));  
```

## How Call Stack Execution Works

- Synchronous functions execute sequentially.  
- Asynchronous functions use the \*\*event loop\*\*.

## Why Callbacks Are Necessary

- Callbacks ensure operations complete before proceeding.

## setTimeout Example

```js  
setTimeout(() => console.log("Hello after 2s"), 2000);  
```

## Importing a Module

```js  
const fs = require('fs');  
```

## Writing a Custom Module

```js  
module.exports = { greet: () => "Hello!" };  
```

## Simple Express App

```js  
const express = require('express');  
const app = express();  
app.get('/', (req, res) => res.send("Hello!"));  
app.listen(3000, () => console.log("Server running"));  
```

## GET vs. POST Requests

- GET retrieves data  
- POST sends data

## Performing a GET Request

```js  
const https = require('https');  
https.get('https://jsonplaceholder.typicode.com/posts/1', res => {  
 res.on('data', d => console.log(JSON.parse(d)));  
});  
```

## Git Basics

```bash  
git add .  
git commit -m "message"  
git push origin main  
```

## Branching in Git

```bash  
git checkout -b new-branch  
```

## MongoDB: Creating a Collection

```js  
db.createCollection("users");  
```

## MongoDB: Inserting Documents

```js  
db.users.insertOne({ name: "Alice", age: 25 });  
db.users.insertMany([{ name: "Bob" }, { name: "Charlie" }]);  
```

## MongoDB: Updating a Document

```js  
db.users.updateOne({ name: "Alice" }, { $set: { age: 30 } });  
```

## MongoDB: Dropping a Collection

```js  
db.users.drop();  
```