Assignment Code: FSD-AG-004

Version Control & Basics of JavaScript | Assignment

Question 1 : Explain the purpose of version control systems like Git. Why are they essential in modern software development?

Answer:

What Is a Version Control System?

A **version control system (VCS)** tracks changes to files over time. It lets developers:

- Save snapshots of their codebase
- Revert to earlier versions if something breaks
- Collaborate without overwriting each other's work

Git is the most widely used VCS today, and for good reason.

Why Git Is Essential in Modern Development

1. Collaboration Without Chaos

- Multiple developers can work on the same project simultaneously.
- Git merges changes intelligently, reducing conflicts and duplication.

2. Change Tracking & History

- Every change is recorded with a timestamp and author.
- You can trace bugs or regressions back to the exact commit.

3. Safe Experimentation

- Branching allows you to test new features or fixes without affecting the main code.
- If something fails, you can discard the branch and start fresh.

4. Rollback & Recovery

- Accidentally deleted a file or introduced a bug? Git lets you roll back to a stable version.

5. Integration with DevOps & CI/CD

- Git integrates seamlessly with automated testing, deployment pipelines, and issue tracking tools.

6. Open Source & Community Power

- Most open-source projects use Git (especially GitHub), making it easier to contribute and learn from others.

Why Even Solo Developers Should Use Git

Even if you're working alone on an Arduino project or a web layout:

- Git acts as your personal safety net.
- It helps you organize versions and document your progress clearly

Question 2: List the typical steps you would follow when working with Git to make changes to a project and share them on GitHub.

Answer:

Typical Git Workflow to Share Changes on GitHub

Step	Command	Purpose
1️⃣ Clone the repository	git clone <repo-url>	Copies the GitHub repo to your local machine
2️⃣ Create a new branch	git checkout -b <branch-name>	Keeps your changes isolated from the main code
3️⃣ Make changes	*(Edit your files)*	Modify code, add features, fix bugs
4️⃣ Stage changes	git add <filename> or git add .	Prepares files to be committed
5️⃣ Commit changes	git commit -m "Descriptive message"	Saves a snapshot of your changes
6️⃣ Push to GitHub	git push origin <branch-name>	Uploads your branch to GitHub
7️⃣ Create a Pull Request	*(On GitHub UI)*	Proposes your changes to be merged into the main branch
8️⃣ Review & Merge	*(Team reviews, then merges)*	Final step to integrate your work

Question 3: What is the purpose of a .gitignore file in a Git repository? Provide an example of how to ignore all .log files

Answer:
The .gitignore file tells Git which files or directories to **ignore**—meaning they won't be tracked or committed to the repository. This is especially useful for:

- **Log files** and temporary outputs
- **Build artifacts** like .class, .exe, or .o files
- **Sensitive data** like API keys or config files
- **Editor/system clutter** like .DS_Store or Thumbs.db

By ignoring these, you keep your repo focused on the actual source code and avoid accidentally sharing private or irrelevant files.

Example: Ignore All .log Files

To ignore every file ending in .log, just add this line to your .gitignore file:

Code

*.log

This pattern matches any .log file in any directory of your project.

Bonus Tip

If you already committed a .log file before adding it to .gitignore, Git will still track it. To stop tracking it:

bash

git rm --cached filename.log

Then commit the change. Want help setting up a .gitignore for one of your Arduino or web projects? I can tailor it to your setup.

Question 4: You are currently on the main branch and want to switch to a branch named feature-login. Write the Git command to do this

Answer:

Code:
git checkout feature-login

Question 5: Declare a variable named age using let and assign it the value 25. Then, update its value to 30.

Answer:

```
let age = 25;  // Declare and initialize
age = 30;      // Update the value
```

Question 6: Declare a constant named PI with the value 3.14159. Attempting to reassign it should result in an error. Demonstrate this behavior by writing out the outcome.

Answer:

```
const PI = 3.14159;
PI = 3.14; // ❌ This will throw an error
```

Outcome:
TypeError: Assignment to constant variable.

The const keyword creates a **read-only reference** to a value. Once assigned, it **cannot be reassigned**—perfect for values like PI that should never change.

Question 7: You are given the following array: const colors = ["red", "green", "blue", "yellow", "purple"];
1. Access and print the first element of the array.
2. Access and print the last element of the array.
3. Access and print the third element of the array.

Answer:

```
const colors = ["red", "green", "blue", "yellow", "purple"];

// 1. First element
console.log(colors[0]); // Output: "red"

// 2. Last element
console.log(colors[colors.length - 1]); // Output: "purple"

// 3. Third element
console.log(colors[2]); // Output: "blue"
```

Question 8: Given x = 10 and y = 5, write expressions to:

1. Add x and y
2. Check if x is greater than y
3. Check if x is equal to y

Answer:

```
let x = 10;
let y = 5;

// 1. Add x and y
let sum = x + y;          // sum = 15

// 2. Check if x is greater than y
let isGreater = x > y;     // isGreater = true

// 3. Check if x is equal to y
let isEqual = x === y;     // isEqual = false
```

Question 9: Identify the data types of the following values:
1. "Hello"
2. 42
3. true
4. Undefined

Answer:

1 "Hello" → **String**
- A sequence of characters enclosed in quotes.

2 42 → **Number**
- Represents numeric values, whether integers or floating-point.

3 true → **Boolean**
- Represents logical values: true or false.

4 undefined → **Undefined**
- A special type indicating a variable has been declared but not assigned a value.

-:Finished:-