Quiz Practice

Unit 1 Review

15

COMP423 / ~~CL14~~

Topics

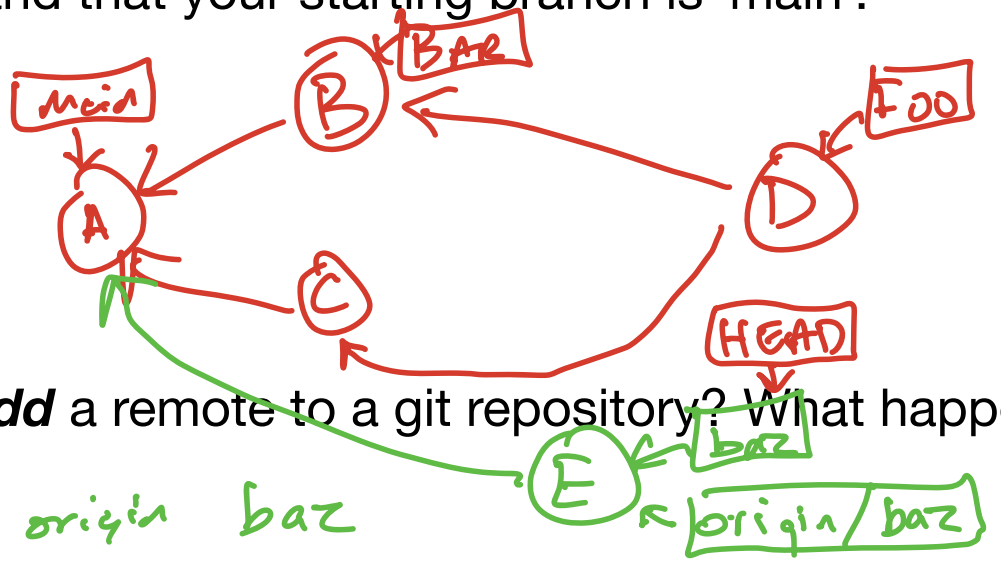
- git Branching, Merging, and Remotes
- Angular Fundamentals (Services, Components)
- Closures, Rest vs. Spread Syntax, Destructuring Assignment
- Dependency Injection
- Metaprogramming with Decorators and @Annotations
- HTTP
- FastAPI Fundamentals from EX03

Git Merging, Branching, and Remotes Review

1. • What is the fundamental difference between a commit and a merge commit?

1 parent vs 2

2. • Assume diverging histories on these branches, but no conflicts! Draw the simplest diagram you can of what happens when you run the following commands (assume neither results in an error) and that your starting branch is `main`:
   - `git switch foo`
   - `git merge bar`



3. • What happens when you ***add*** a remote to a git repository? What happens when you ***fetch*** from it?

git fetch origin bar

Angular Fundamentals

- Justify why Angular distinguishes between Components and Services.
  - Components → View
  - Services → State, Model
    Communication w/ Server
- Delegation
- Abstraction

- What are the three parts of a Component and what are their purposes?
  - Template
    - HTML → Structure of UI — View
    - CSS → Style of UI
  - TS → View Controller

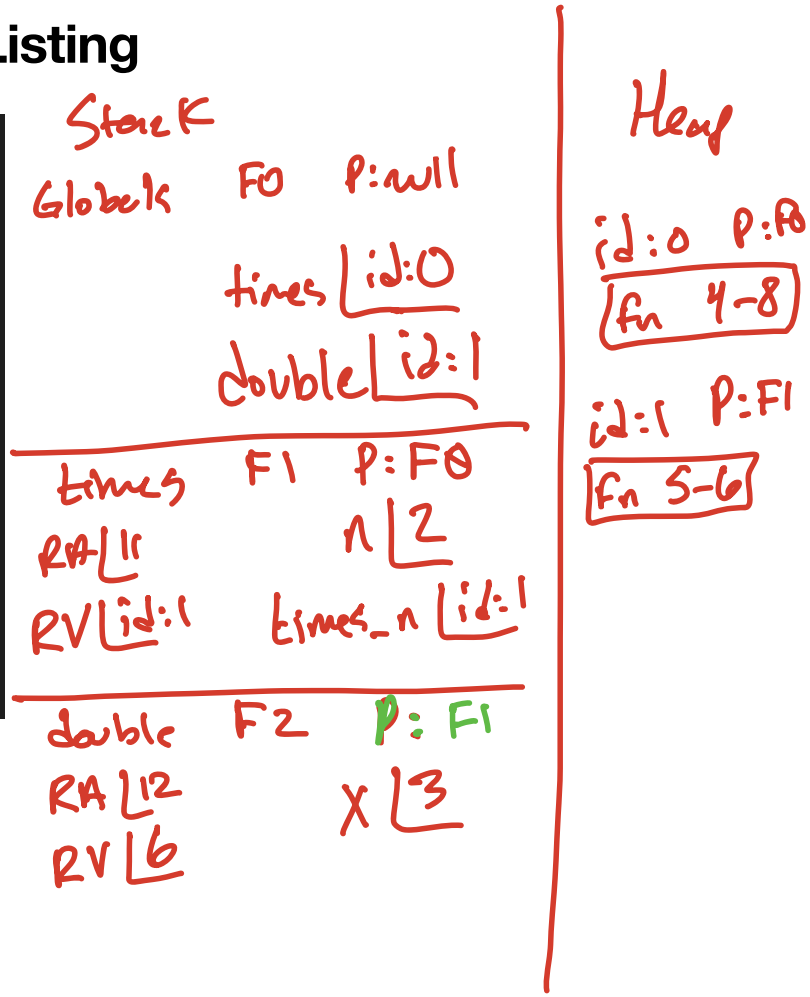
MVC

Closures: Diagram the Following Code Listing

```
1   from typing import Callable
2
3
4   def times(n: int) -> Callable[[int], int]:
5       def times_n(x: int) -> int:
6           return x * n
7
8       return times_n
9
10
11  double = times(2)
12  print(double(3))
```

OUTPUT

6



VarArgs/Rest Syntax vs. Spread Syntax

```
1   const f = (y: number, ...xs: number[]) => {
2     console.log(xs.map((x) => x + y));
3   };
```

Write two function calls to f, the first should *not* use spread syntax, the second *must* use spread syntax. You can define additional variables if useful.

Not using spread

```
f(1, 2, 3)
f(1, 2)
f(1)
```

```
let vals = [3, 4, 5];
f(1, ...vals)
```

Dependency Injection

- Describe the general concept of *dependency injection* at a high level. What benefits does it provide to a system? Give an example of how it is used in Angular or FastAPI.

DI increases modularity and simplifies testing by decoupling the construction of dependencies from the consumers.

Injected dependencies can be easily mocked or doubled for testing. Reduces boilerplate of construction of dependencies.

In angular, DI is seen in the constructors of components where instances of services are passed in automatically.

In FastAPI, seen in route signatures, also

injecting things like services or DB sessions.

Metaprogramming

@Decorators are used for *metaprogramming* in both Angular and FastAPI. Where have you seen them used in each framework? What do these uses have in common?

Angular: Registering @Components w/ Metadata
Also seen on @Input and @Output props of components.
Registering @Injectable dependencies (services)

FastAPI: Registering routes with metadata

Common: Registering classes with the framework and providing metadata / configuration directly in the source code attached to the classes / functions they are

declaratively annotating.

FastAPI and HTTP

- What is the difference between path and query parameters?

Path params are segments of a URL, eg: /user/{user_id}

Query params are key/value pairs of a URL, eg: following the path /search?keywords=foo

- What HTTP method(s) do ***not*** have a request body?

GET!