

.gitignore 사용하기

.gitignore 사용하기 - 1

- .gitignore란?

GitHub에 올리지 말아야 할 파일들을 설정할 수 있는 파일

ex) db 접속 정보, AWS Key 등등

- 사용 방법

현재 작업중인 폴더에 .gitignore 파일을 생성 후 무시할 파일명을 저장



.gitignore 사용하기 - 2

- .gitignore 설정 전

git status 명령어를 입력했을 경우 dbcon.html 파일을 찾는 것을 확인 가능

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dbcon.html

nothing added to commit but untracked files present (use "git add" to track)
```

.gitignore 사용하기 - 3

- .gitignore 설정 후

dbcon.html 파일을 찾지 않음

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

.gitignore 사용하기 - 4

- .gitignore란?

기본적으로 리눅스 환경의 폴더 구조에 익숙해져야 사용하기 편함

* gitignore 파일은 윈도우 경로(₩)가 아닌 리눅스 경로(/)를 사용하기 때문

구분자	설명
filename.html	filename.html 무시
/filename.html	최상위 폴더의 filename.html 무시
*.html	모든 html 확장자 파일을 무시
!filename.html	filename.html만 무시하지 않음
views	views 파일 또는 폴더 무시
views/	views 폴더 안의 내용을 무시
views/*.html	views 폴더 안의 html 확장자 무시

branch 활용하기

branch 기초 활용 (생성, 이동)

브랜치 기초 활용 - 1

- 브랜치란?

하나의 프로젝트에서 여러가지 모습으로 독립적인 작업을 진행하는 개념

*일부 개발자들은 "가지치기"라고 학습하기도 함

브랜치 기초 활용 - 2

• 브랜치 관련된 명령어

1. 브랜치 생성 : git branch [브랜치명]
2. 현재 활성화된 브랜치 및 전체 목록 확인 : git branch

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)  
$ git branch firstBranch
```

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)  
$ git branch  
firstBranch  
kimjaeseop  
* master
```

브랜치 기초 활용 - 3

• 브랜치 관련된 명령어

1. 브랜치 이동 : git switch [브랜치명]

```
user@DESKTOP-T9RHACC MINGW64 /d/gitWorkspace (master)
$ git switch firstBranch
Switched to branch 'firstBranch'
M      index.html
```

```
user@DESKTOP-T9RHACC MINGW64 /d/gitWorkspace (firstBranch)
$ git branch
* firstBranch
  kimjaeseop
  master
```

브랜치 기초 활용 - 3

- 참고

아직 많은 개발자들이 checkout을 사용하기도 하지만, 2.23 버전부터 checkout 대신 switch, restore 명령으로 나뉘져 도입되었음

checkout도 사용이 가능하지만 되도록 switch, restore 명령을 사용 및 학습 하는 것을 권장

브랜치 기초 활용 - 4

- Q1. master 브랜치로 이동 후 firstBranch를 삭제 하세요.

브랜치 삭제 명령어 : `git branch -d [브랜치명]`

- Q2. originalBranch를 생성하고 SecondBranch로 이름을 변경 하세요.

브랜치 이름 변경 명령어 : `git branch -m [원본 브랜치명] [변경할 브랜치명]`

- Q3. firstBranch를 다시 생성하고 firstBranch로 이동 하세요.

브랜치 기초 활용 - 5

- firstBranch에서 board.html 생성

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Board</title>
</head>
<body>
  여기는 게시판 입니다.
  firstBranch
</body>
</html>
```

브랜치 기초 활용 - 6

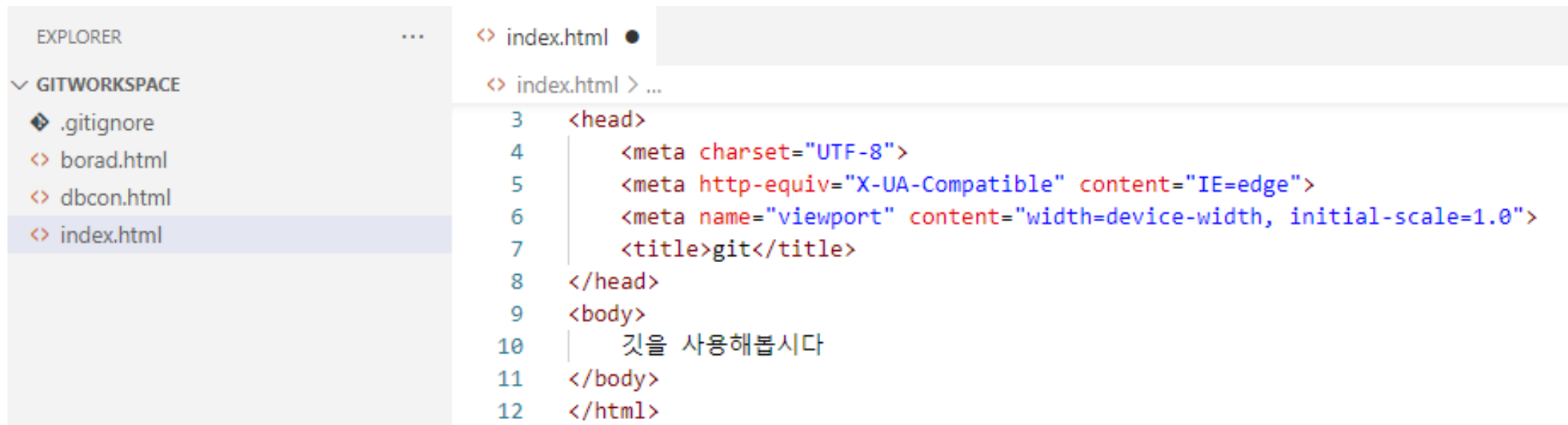
• 작업 공간 확인하기 - 1

secondBranch로 이동하여 firstBranch에서 작업한 내용이 사라지는것을 확인

브랜치 기초 활용 - 7

• 작업 공간 확인하기 - 2

다시 firstBranch로 이동하여 작업하던 내용이 생긴 것을 확인



```
EXPLORER
...
GITWORKSPACE
  .gitignore
  borad.html
  dbcon.html
  index.html

index.html
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>git</title>
8  </head>
9  <body>
10   |   깃을 사용해봅시다
11 </body>
12 </html>
```

브랜치 기초 활용 - 8

- 작업 공간 확인하기 - 3

git log --all --decorate --oneline --graph 명령어 사용해보기

branch 기초 활용 (합치기)

브랜치 기초 활용 - 1

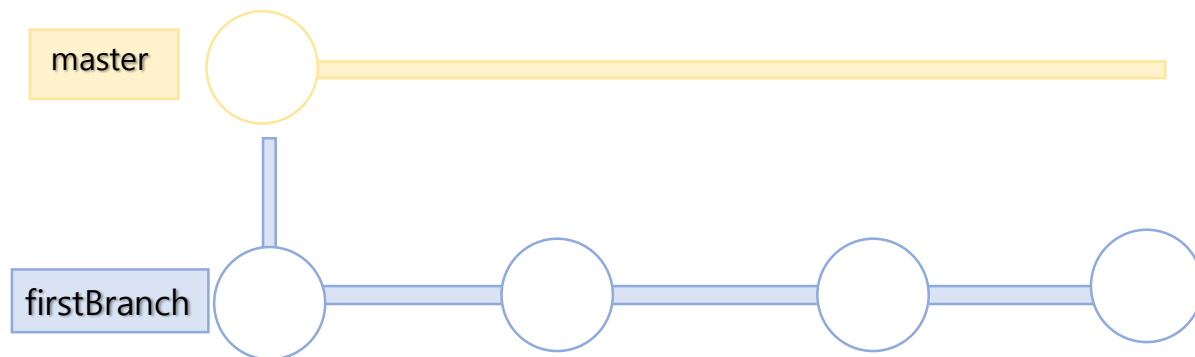
- **merge**

현재의 브랜치와 다른 브랜치를 하나의 커밋에 이어 붙이는 명령
사용 방법 : `git merge` [이어붙일 브랜치]

브랜치 기초 활용 - 2

- 설명

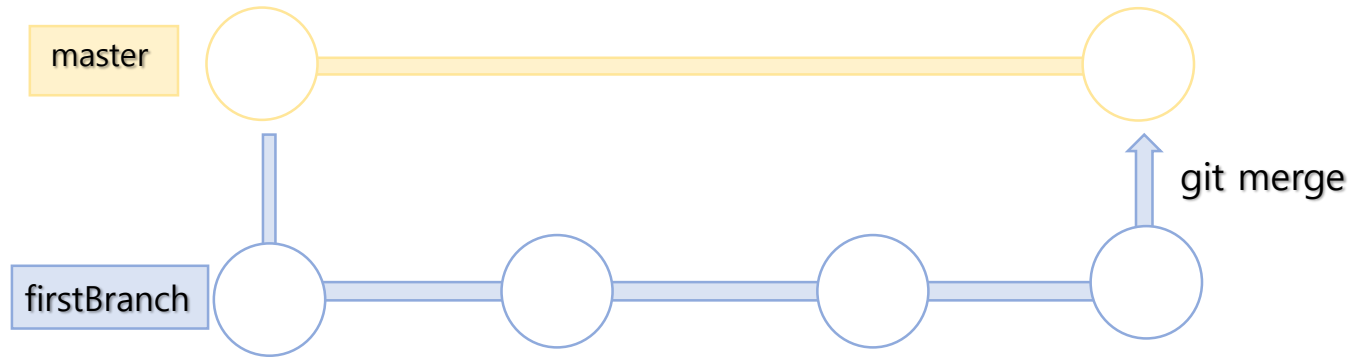
현재는 master와 firstBranch가 서로 다른 작업을 진행하고 commit되어있는 상태



브랜치 기초 활용 - 3

- merge

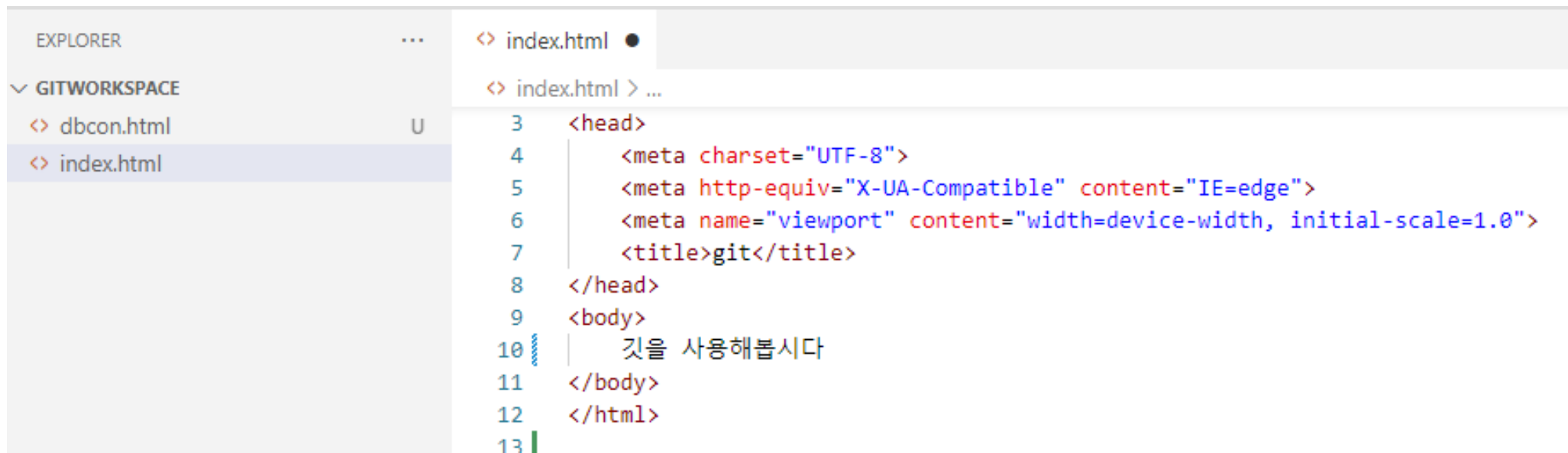
firstBranch에서 commit까지 진행한 작업 내용을 master에 이어붙임



브랜치 기초 활용 - 4

- merge 사용해보기

master로 이동 후 현재 작업 내용을 확인



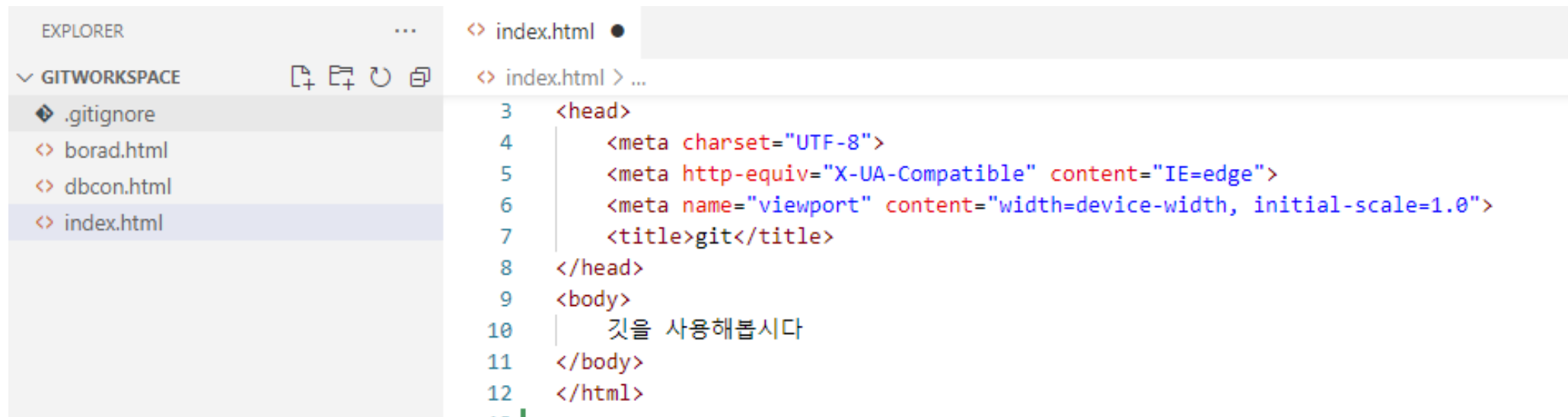
```
EXPLOLERER  ...
v GITWORKSPACE
  <> dbcon.html  U
  <> index.html

<> index.html ●
<> index.html > ...
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>git</title>
8  </head>
9  <body>
10 |   깃을 사용해봅시다
11 </body>
12 </html>
13 |
```

브랜치 기초 활용 - 5

- merge 사용해보기

git merge firstBranch 명령어를 사용하여 작업했던 내용이 생기는 것을 확인



```
<> index.html •
<> index.html > ...
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>git</title>
8  </head>
9  <body>
10     |   깃을 사용해봅시다
11 </body>
12 </html>
--
```

branch 기초 활용 (충돌 상황 해결하기)

브랜치 기초 활용 - 1

- 충돌 상황

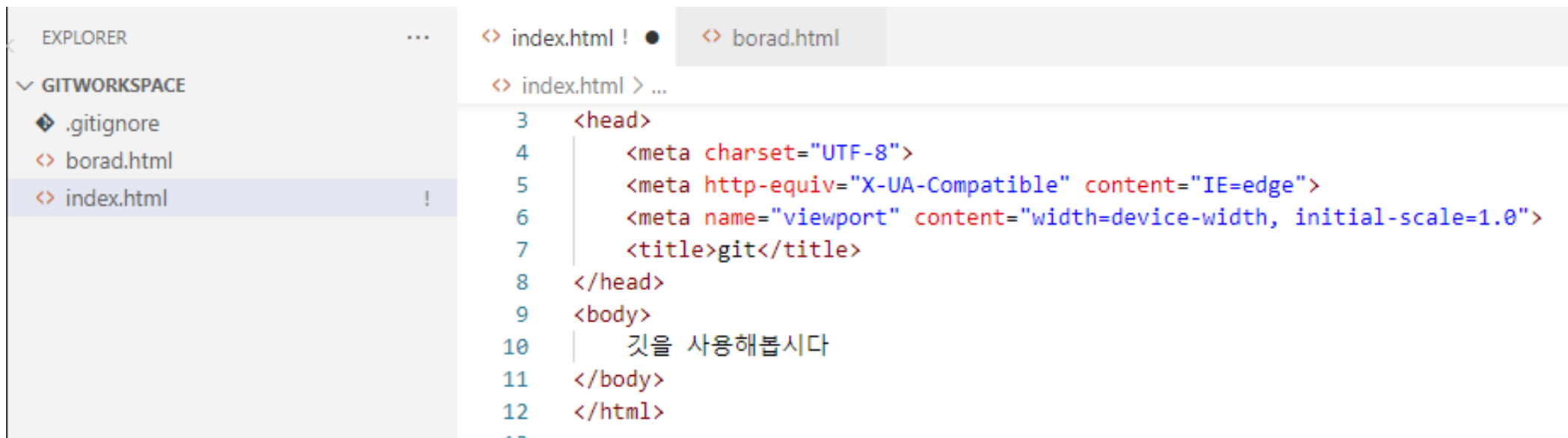
하나 또는 여러 개의 파일에 대해 각각의 브랜치가 서로 다른 내용을 가지고 있을 때 발생

* 충돌이라고도 하지만 "conflict가 발생했다" 라고도 많이 말함

브랜치 기초 활용 - 2

- 충돌 상황 만들어보기

master의 index.html에 내용을 변경 후 add 및 commit



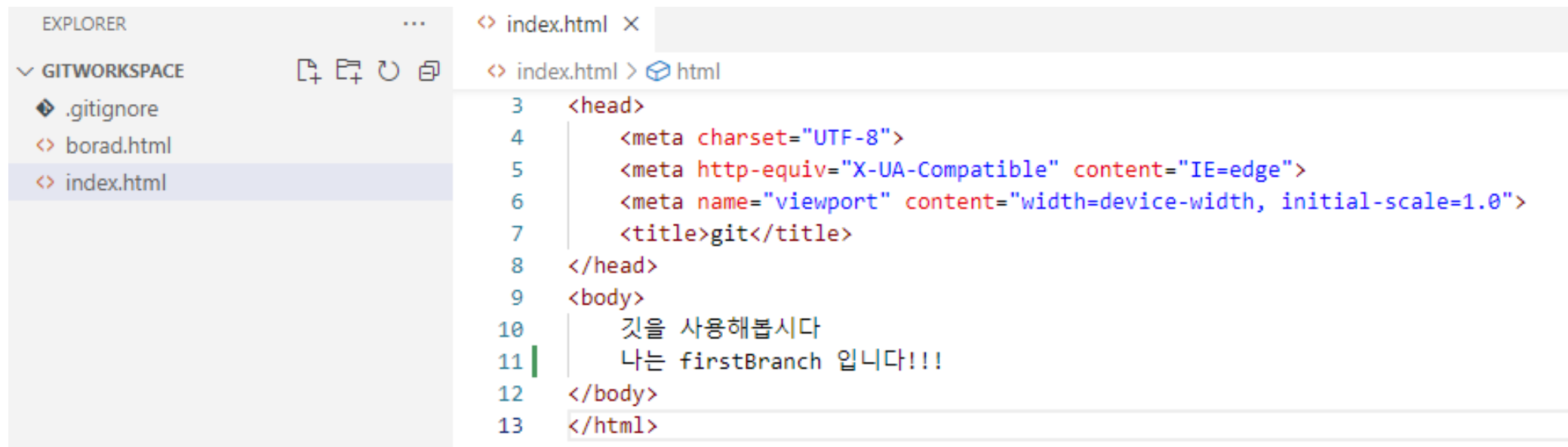
The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORER' sidebar displays the 'GITWORKSPACE' with files: '.gitignore', 'borad.html', and 'index.html'. The 'index.html' file is selected and highlighted. On the right, the editor window shows the content of 'index.html'. The file has a tab labeled 'index.html !' and a code editor with the following HTML code:

```
<? index.html ! ● <? borad.html
<? index.html > ...
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>git</title>
8  </head>
9  <body>
10     깃을 사용해봅시다
11 </body>
12 </html>
13
```

브랜치 기초 활용 - 3

- 충돌 상황 만들어보기

firstBranch의 index.html에 내용을 변경 후 add 및 commit



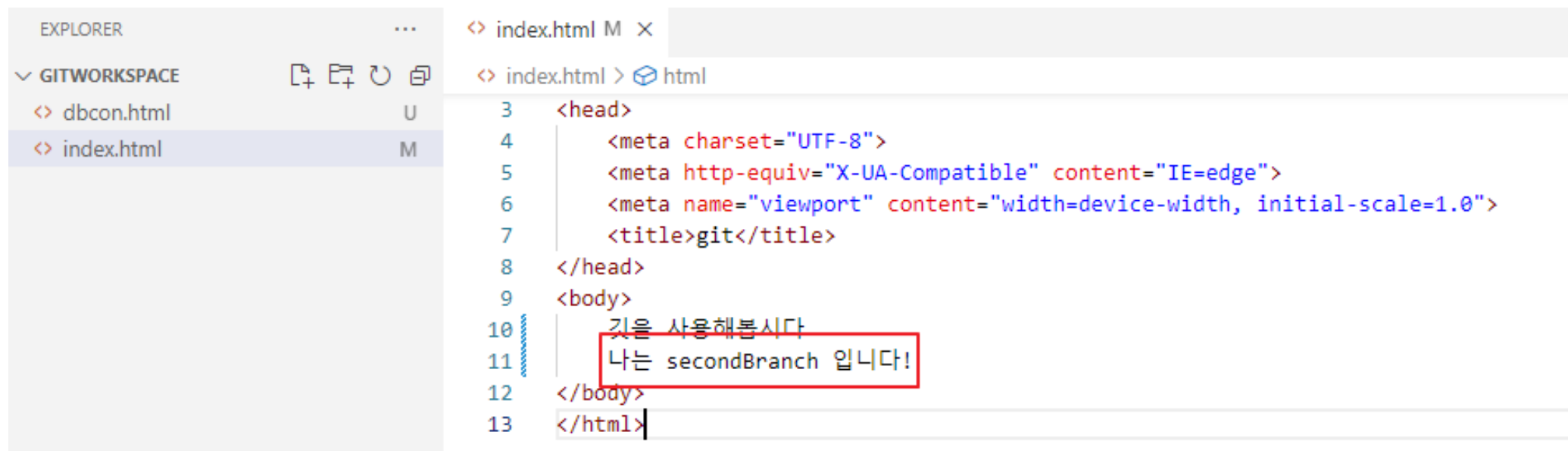
The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORER' sidebar displays the 'GITWORKSPACE' with files: '.gitignore', 'borad.html', and 'index.html'. The 'index.html' file is selected. On the right, the editor window shows the content of 'index.html' with the following code:

```
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>git</title>
8 </head>
9 <body>
10   깃을 사용해봅시다
11   나는 firstBranch 입니다!!!
12 </body>
13 </html>
```

브랜치 기초 활용 - 4

- 충돌 상황 만들어보기

secondBranch의 index.html에 내용을 변경 후 add 및 commit



```
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>git</title>
8 </head>
9 <body>
10  깃을 사용해봅시다
11  나는 secondBranch 입니다!
12 </body>
13 </html>
```

브랜치 기초 활용 - 5

• 충돌 상황 해결

일반적인 상황에서는 개발자가 충돌 상황을 해결하기 위해서는 둘 중 한가지를 선택해야 함

1. merge 작업을 중단(취소)
→ `git merge --abort`
2. 충돌을 해결하고 merge를 완료

```
9  <body>
10  |   깃을 사용해봅시다
    Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
11  <<<<<<< HEAD (Current Change)
12  =====
13  |   나는 firstBranch 입니다!!
14  >>>>>>> firstBranch (Incoming Change)
15  </body>
16  </html>
```

브랜치 기초 활용 - 6

• 충돌 상황 해결

git merge firstBranch를 해서 두 브랜치를 이어붙이려고 시도할 경우 아래와 같이 충돌이 발생함

1. <<<HEAD 부분이 현재 마스터 브랜치의 최신 커밋 내용
2. >>>firstBranch가 merge하려는 브랜치의 최신 커밋 내용

컴퓨터는 서로 다른 내용 중 어떤걸 사용해야할지 모르기 때문에 개발자에게 결정권을 넘김

```
9  <body>
10  |   깃을 사용해봅시다
    | Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
11  <<<<<<< HEAD (Current Change)
12  =====
13  |   나는 firstBranch 입니다!!
14  >>>>>>> firstBranch (Incoming Change)
15  </body>
16  </html>
```