presents

Intermediate source control

Git is an Open Source Distributed Version Control System

- **Control System:** Git is a content tracker. So Git can be used to store content — it is mostly used to store code due to the other features it provides.
- **Version Control System**: The code which is stored in Git keeps changing as more code is added. Also, many developers can add code in parallel. So Version Control System helps in handling this by maintaining a history of what changes have happened.
- **Distributed Version Control System**: Git has a remote repository which is stored in a server and a local repository which is stored in the computer of each developer. This means that the code is not just stored in a central server, but the full copy of the code is present in all the developers' computers. Git is a Distributed Version Control System since the code is present in every developer's computer.

Merge conflicts

-They may seem like a bad thing at first

-Merge conflicts happen when you merge branches that have competing commits, and Git needs your help to decide which changes to incorporate in the final merge.

-Often, merge conflicts happen when people make different changes to the same line of the same file, or when one person edits a file and another person deletes the same file.

-You must resolve all merge conflicts before you can merge a pull request on GitHub.

Resolving merge conflicts

-To resolve a merge conflict caused by competing line changes, you must choose which changes to incorporate from the different branches in a new commit.

-For example, if you and another person both edited the file *index.js* on the same lines of the same Git repository, you'll get a merge conflict error when you try to pull changes. You must resolve this merge conflict with a new commit before you can merge the changes.

Encountering a conflict

Auto-merge failed

Run git status to see which files have merge conflicts

Files with merge conflicts

```
~/OneDrive/Desktop/HumanSociety/HumanSocietyADJ (master)
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 3), reused 4 (delta 3), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/anthonycala/HumanSocietyADJ
   80247fa..92e0d4a  master     -> origin/master
Auto-merging HumaneSociety/Query.cs
CONFLICT (content): Merge conflict in HumaneSociety/Query.cs
Automatic merge failed; fix conflicts and then commit the result.

~/OneDrive/Desktop/HumanSociety/HumanSocietyADJ (master|MERGING)
$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   HumaneSociety/Query.cs

no changes added to commit (use "git add" and/or "git commit -a")

~/OneDrive/Desktop/HumanSociety/HumanSocietyADJ (master|MERGING)
```

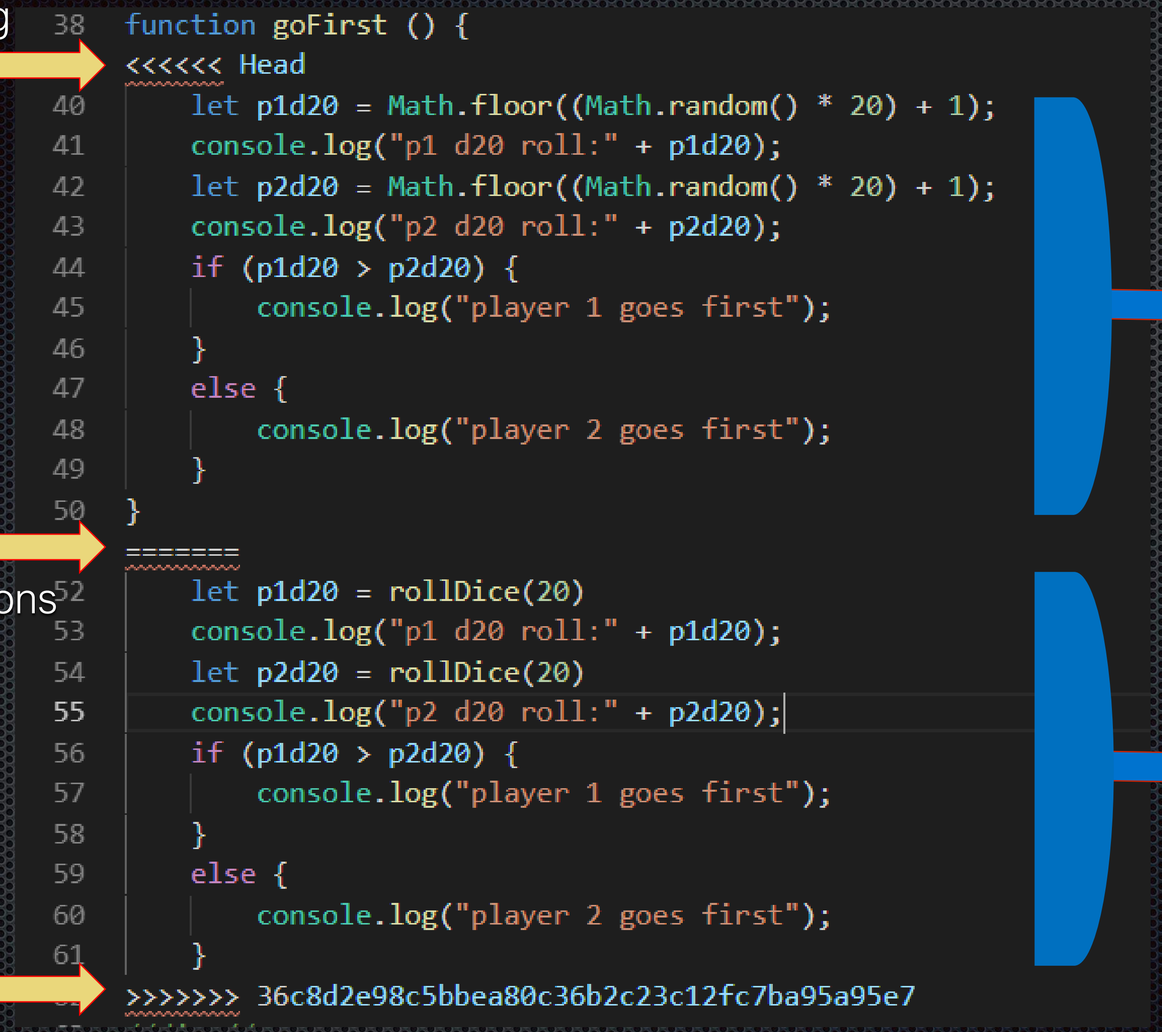
Now in merging

What it looks like

Signifies the beginning of a merge conflict

Separation between Current/Incoming versions

End of merge conflict



Current

Incoming

-Decide which version you want to keep. Current or incoming.

-Delete all the code from change you don't want to keep.

-Delete all merge syntax.

Remove

Delete

```
38  function goFirst () {
<<<<<< Head
40      let p1d20 = Math.floor((Math.random() * 20) + 1);
41      console.log("p1 d20 roll:" + p1d20);
42      let p2d20 = Math.floor((Math.random() * 20) + 1);
43      console.log("p2 d20 roll:" + p2d20);
44      if (p1d20 > p2d20) {
45          console.log("player 1 goes first");
46      }
47      else {
48          console.log("player 2 goes first");
49      }
50  }
=======
52      let p1d20 = rollDice(20)
53      console.log("p1 d20 roll:" + p1d20);
54      let p2d20 = rollDice(20)
55      console.log("p2 d20 roll:" + p2d20);
56      if (p1d20 > p2d20) {
57          console.log("player 1 goes first");
58      }
59      else {
60          console.log("player 2 goes first");
61      }
>>>>>>> 36c8d2e98c5bbea80c36b2c23c12fc7ba95a95e7
```

Resolved conflict

```
38  function goFirst () {
39      let p1d20 = rollDice(20)
40      console.log("p1 d20 roll:" + p1d20);
41      let p2d20 = rollDice(20)
42      console.log("p2 d20 roll:" + p2d20);
43      if (p1d20 > p2d20) {
44          console.log("player 1 goes first");
45      }
46      else {
47          console.log("player 2 goes first");
48      }
49  }
```

Things to remember!

-There maybe more than one file with merge conflicts.

-There maybe more than one merge conflict per file.

-Once all conflicts have been resolved, you must stage the changes and commit them with a message.

-Don’t push merge conflicts to repo!