

# **Simplify writing code with deliberate commits**

**Front-end London - May 2019**

**Joel Chippindale - @joelchippindale**

**UNMADE**

**Joel Chippindale**  
**CTO at**  
**Unmade**




**“If you can’t explain it  
simply, you don’t  
understand it well  
enough.”**

**- Albert Einstein (probably)**

**[https://en.wikiquote.org/wiki/Albert\\_Einstein](https://en.wikiquote.org/wiki/Albert_Einstein)**

**When I started using git it  
changed the way that I  
developed software**





**Our work requires  
us to make changes  
to complex systems**



**We learn to break down  
complex problems into  
small, simple steps**



A young woman with long dark hair, wearing glasses and a red and black plaid shirt, is smiling and looking to the right. The background is a blurred office or indoor setting.

**Great developers are  
really good at  
breaking problems  
into simple steps**



# This is hard...

**...and it takes discipline  
and willpower...**

**...but we can learn to get  
better**

## **5 Habits**

- 1. Plan your commits**
- 2. Use single purpose branches**
- 3. Make atomic commits**
- 4. Write good commit messages**
- 5. Rewrite your history to tell a story (early and often)**





Habit 1:

Plan your commits



**Make a plan for the  
commmits that you will  
make**

**What if you don't know  
enough yet to make a  
plan?**

# What if the plan changes?

**Plan your commmits ahead  
of time and re-plan when  
you need to**



A long, dimly lit tunnel with a person walking in the distance. The tunnel has a brick or stone wall and a series of lights along the ceiling. The perspective is from the entrance, looking down the length of the tunnel.

**Habit 2:**

**Use single purpose**

**branches**

**Name your branch to  
reflect it's (single)  
purpose**



**Notice when you are  
starting to work on  
something else**

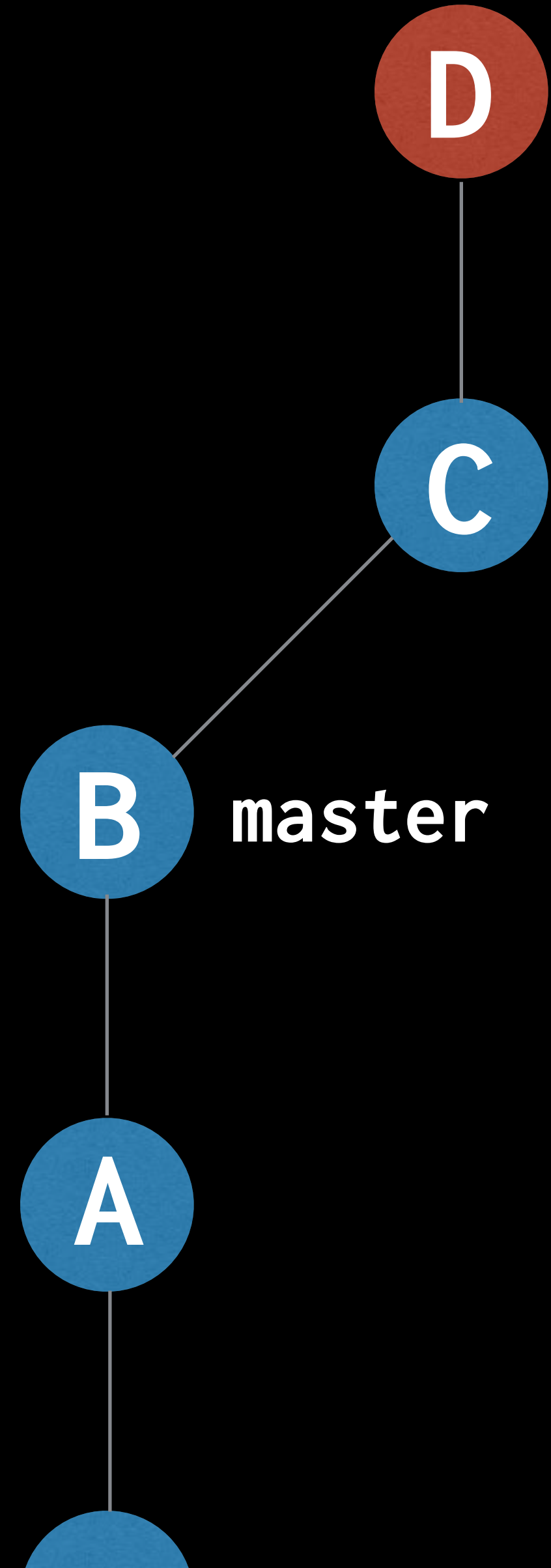
**Notice if a commit has  
value independent of the  
branch**



**...and if it does, then  
'cherry pick' it onto  
master**

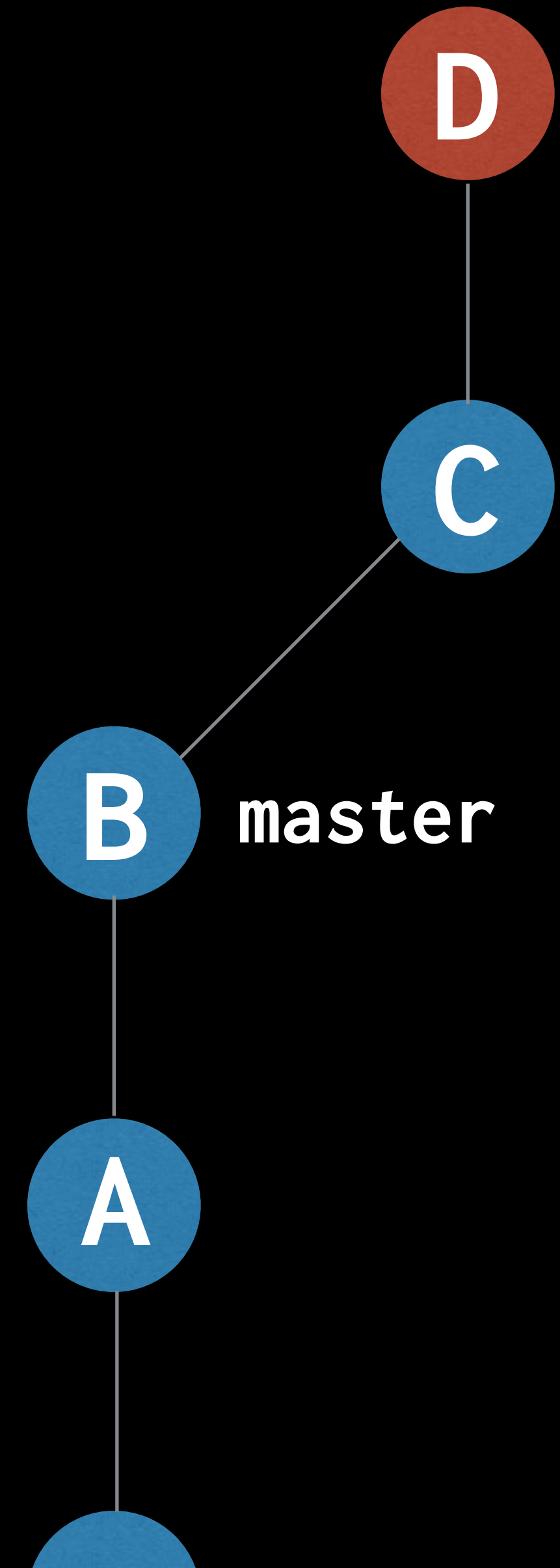
\$

your-dev-branch



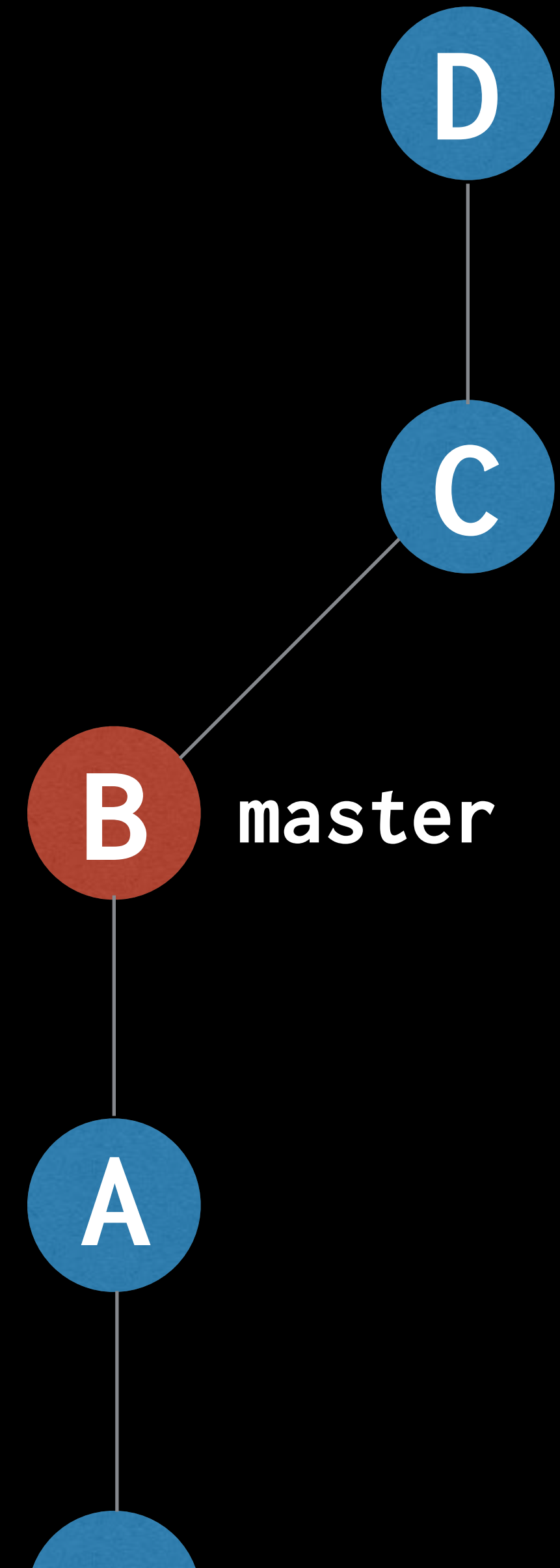
```
$ git checkout master
```

your-dev-branch



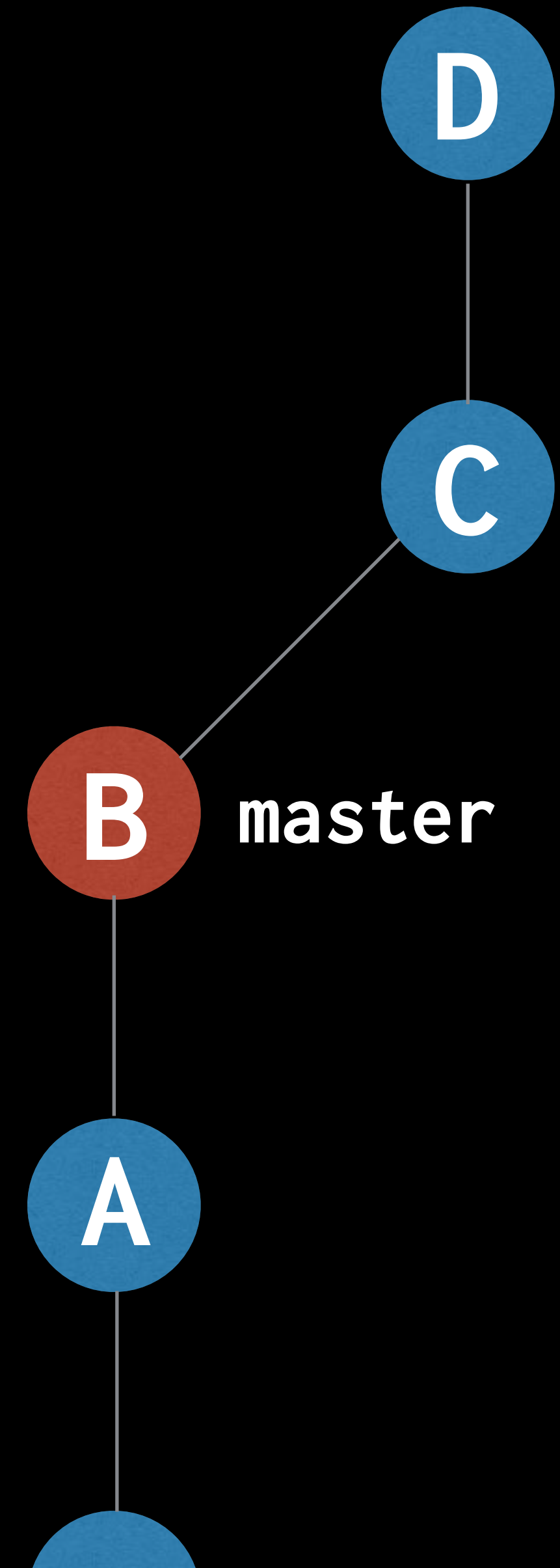
```
$ git checkout master
```

your-dev-branch

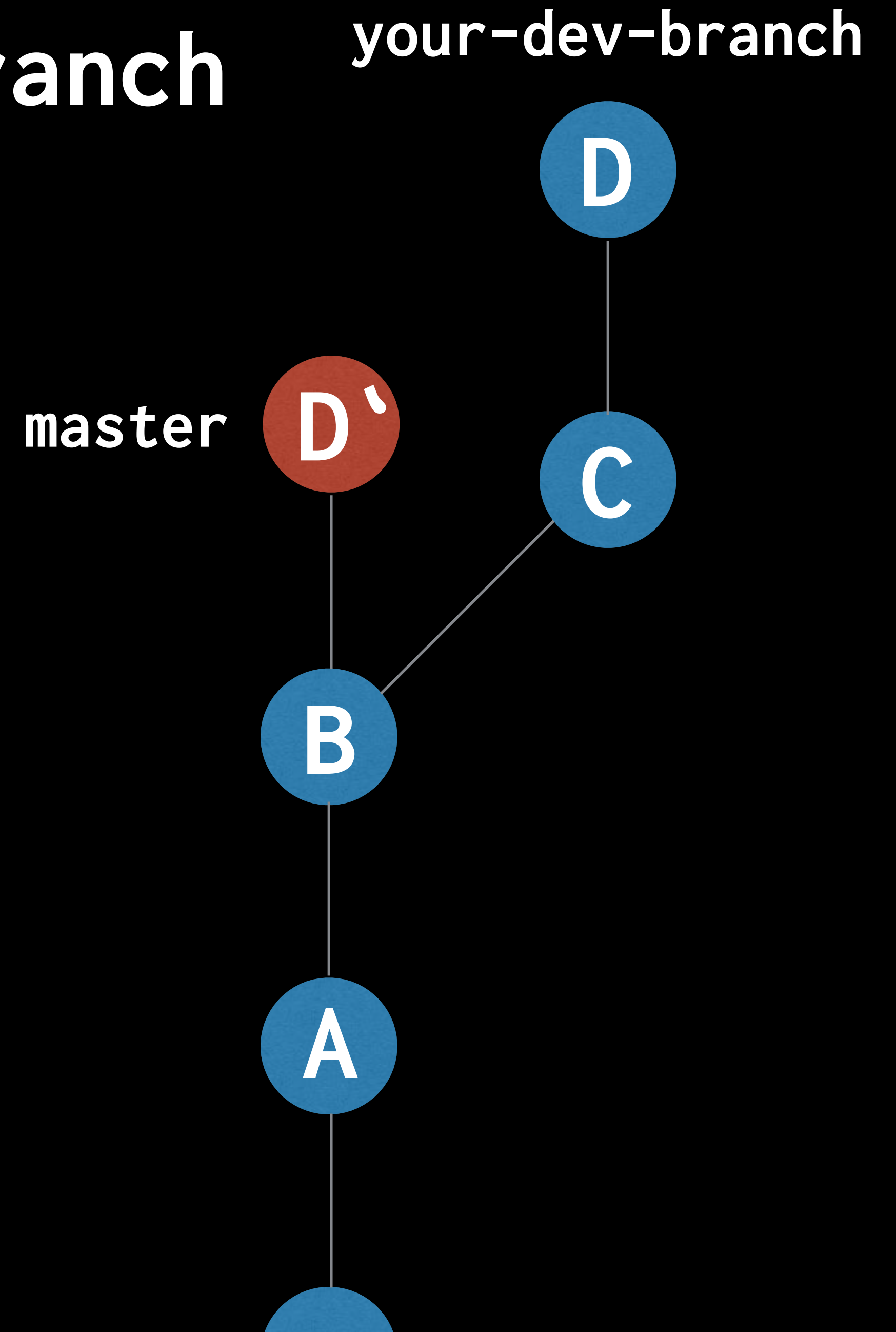


```
$ git cherry-pick your-dev-branch
```

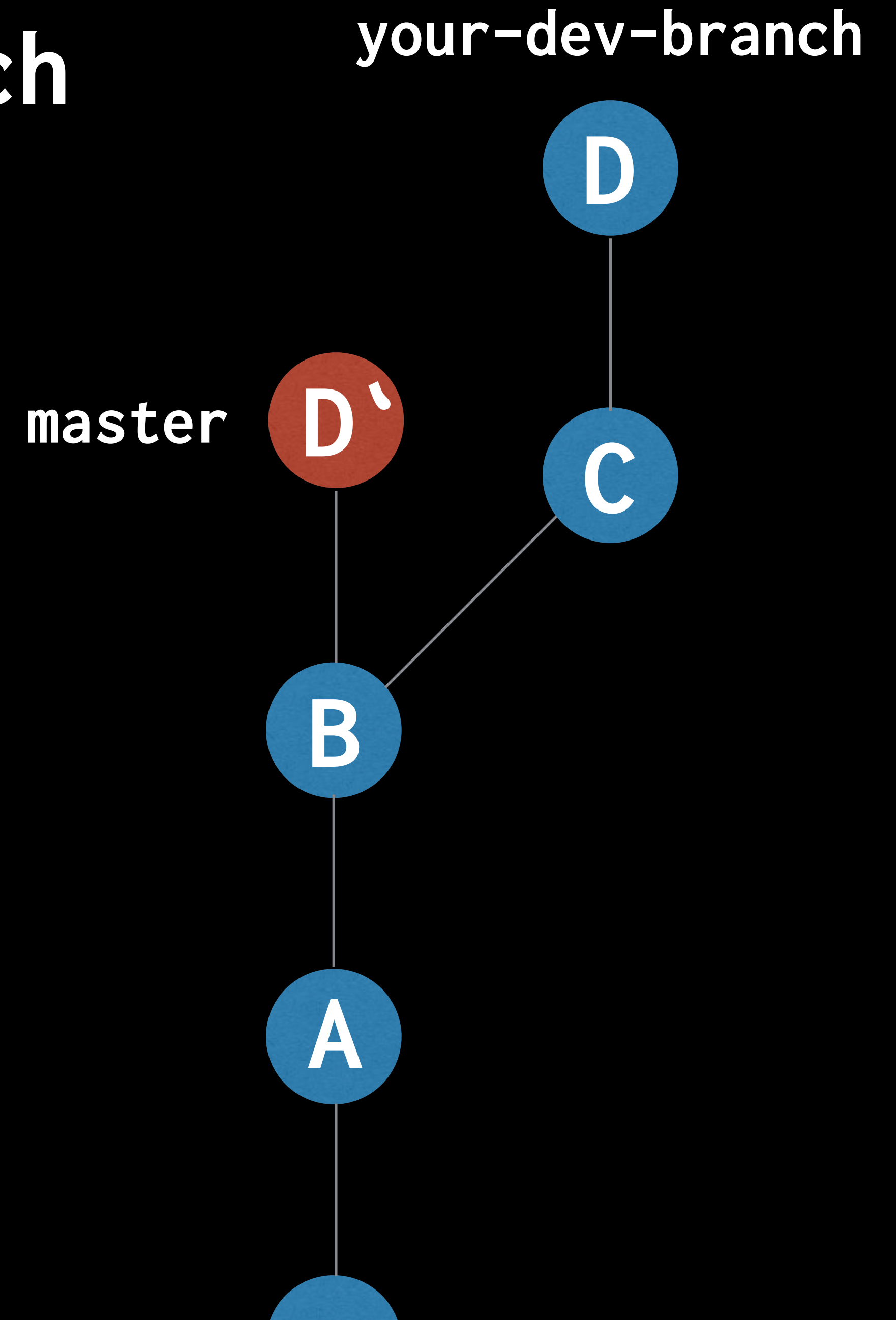
your-dev-branch



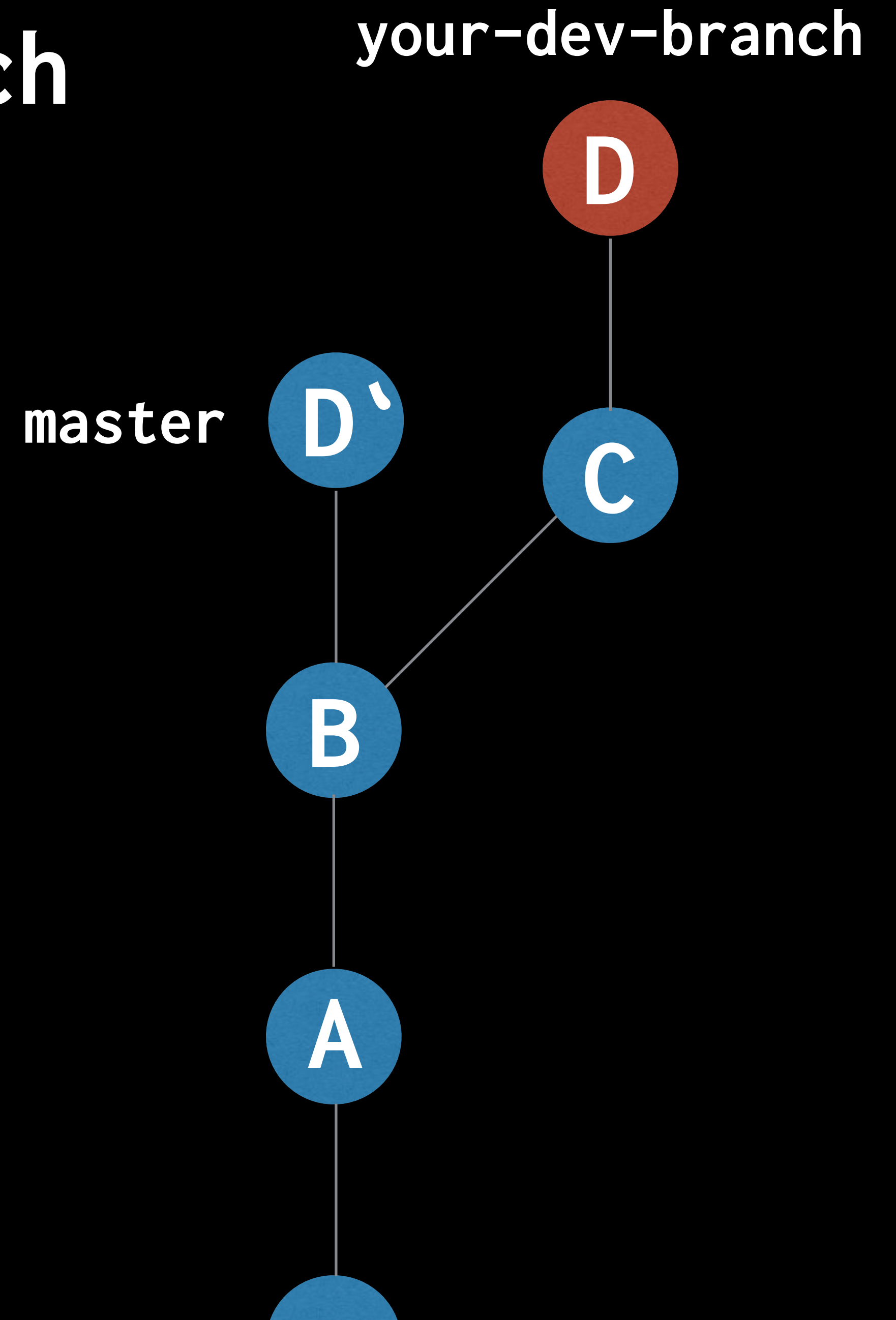
```
$ git cherry-pick your-dev-branch
```



```
$ git checkout your-dev-branch
```

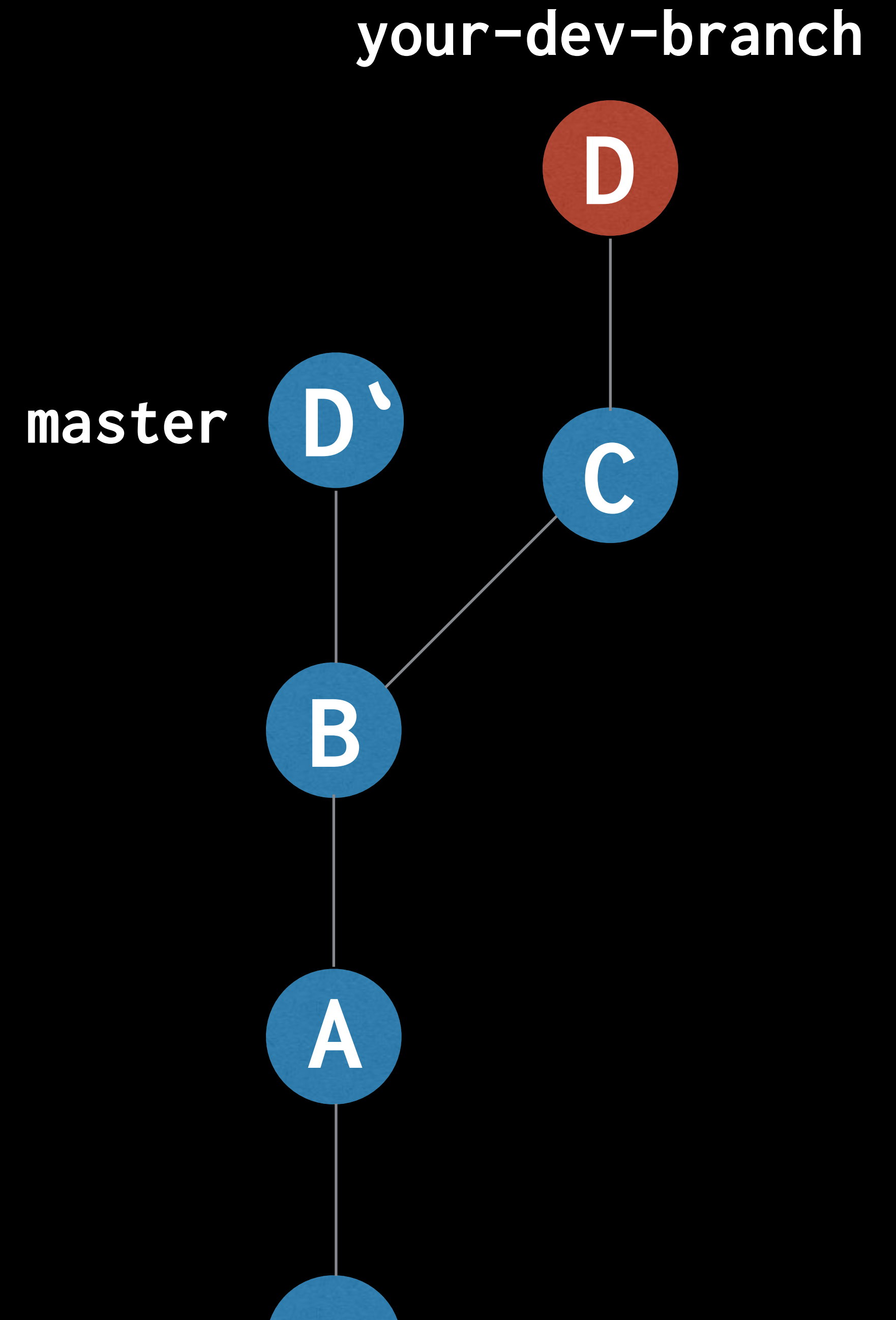


```
$ git checkout your-dev-branch
```

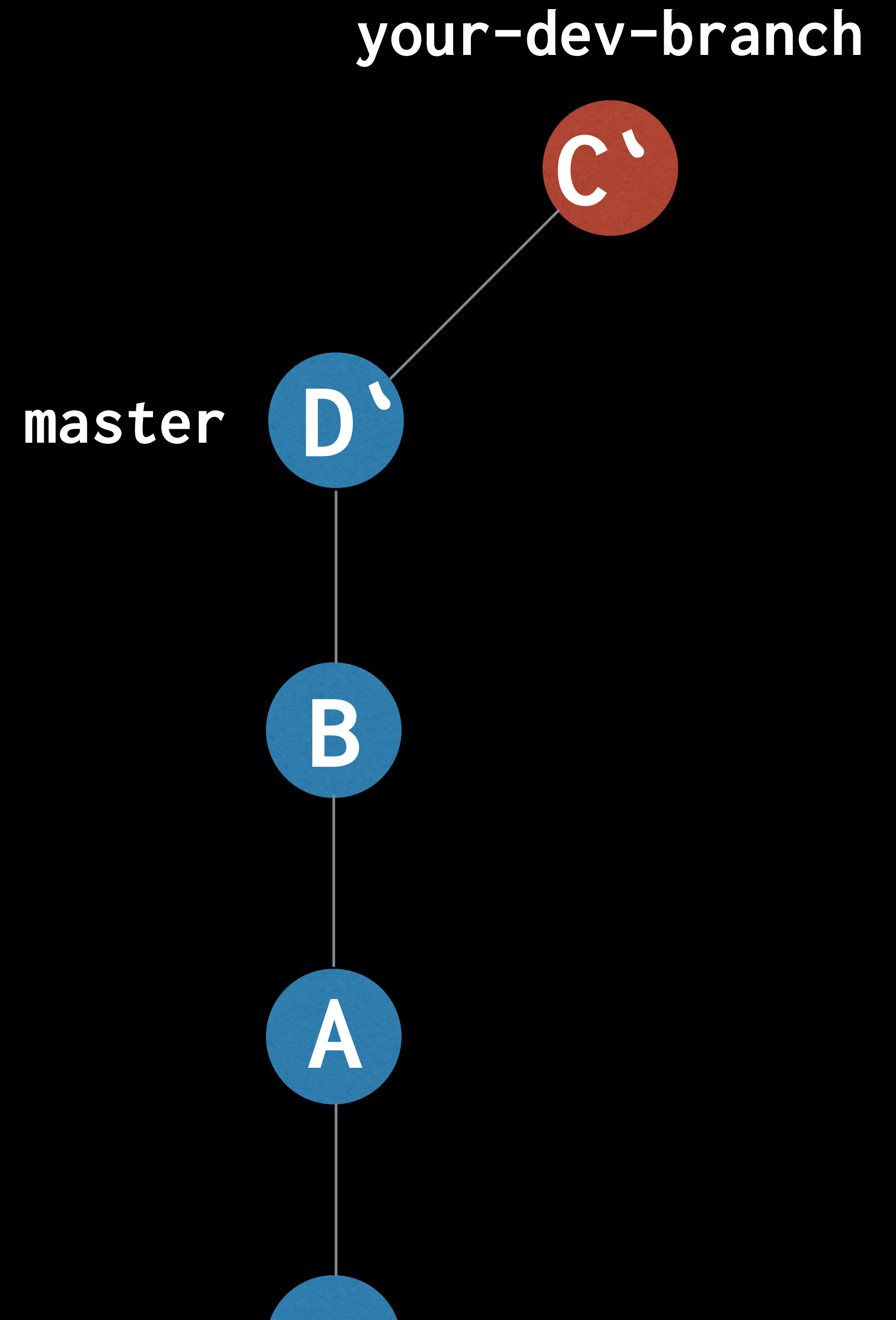




```
$ git rebase master
```



```
$ git rebase master
```



**Keep focussed by making  
each development  
branch single purpose**

A large, glowing, golden-yellow sphere, resembling a sun or moon, is positioned on the right side of the frame against a black background. The sphere has a soft, radial gradient, being brightest in the center and fading towards the edges. It is partially obscured by a semi-transparent grey rectangular box on the left side of the image.

**Habit 3:**

**Make atomic  
commits**

**Decide the one change  
you are going to make  
and commit it**

**How big a change should  
I make?**

# Minimum valuable commit

**Be suspicious of 'and' in  
your commit message**



**Notice when you start  
doing something else and  
stop**

```
$ git add --patch
```

```
$ git add --patch
diff --git a/src/routes.js b/src/routes.js
index b3ab507..f560432 100644
--- a/src/routes.js
+++ b/src/routes.js
@@ -5,10 +5,7 @@ const Chrome = require('@unmade/puppeteer');

  const knitpreview = require('./knitpreview');

-const {
-  createCanvas,
-  renderPattern
-} = require('./editors-v2');
+const { createCanvas, renderPattern } = require('./in_browser');
  const { parseBase64DataURL } = require('./helpers');

  const router = express.Router(); // eslint-disable-line
Stage this hunk [y,n,q,a,d,j,g,/,e,?]?
```

**Make each step simple by  
making atomic commits**





**Habit 4:**

**Write good commit  
messages**



2867d63 Final commit, ready for tagging  
8cecf9e foo  
880f22c WTF  
feb8cd1 More work on this  
a8c9f94 WIP  
46c4aa4 This will definitely work  
79bbf47 This might work  
9ccd522 Trying to fix it again  
6eb4a7f Debug stuff

**What does good look  
like?**

**Short one line title**

Longer description of what the change does (if the title isn't enough).

An explanation of why the change is being made.

Perhaps a discussion of context and/or alternatives that were considered.



Short one line title

Longer description of what the change does  
(if the title isn't enough).

An explanation of why the change is being  
made.

Perhaps a discussion of context and/or  
alternatives that were considered.

Short one line title

Longer description of what the change does  
(if the title isn't enough).

**An explanation of why the change is being  
made.**

Perhaps a discussion of context and/or  
alternatives that were considered.

Short one line title

Longer description of what the change does  
(if the title isn't enough).

An explanation of why the change is being  
made.

**Perhaps a discussion of context and/or  
alternatives that were considered.**

**Short one line title**

**Longer description of what the change does  
(if the title isn't enough).**

**An explanation of why the change is being  
made.**

**Perhaps a discussion of context and/or  
alternatives that were considered.**

Correct the colour of FAQ link in course notice footer

PT: <https://www.pivotaltracker.com/story/show/84753832>

In some email clients the colour of the FAQ link in the course notice footer was being displayed as blue instead of white. The examples given in PT are all different versions of Outlook. Outlook won't implement CSS changes that include `!important` inline[1]. Therefore, since we were using it to define the colour of that link, Outlook wasn't applying that style and thus simply set its default style (blue, like in most browsers). Removing that `!important` should fix the problem.

[1] <https://www.campaignmonitor.com/blog/post/3143/outlook-2007-and-the-inline-important-declaration/>

**Clear space in your head  
by writing good commit  
messages**



Habit 5:

Rewrite your history  
to tell a simple story  
(early and often)



```
$ git rebase --interactive
```



**Remove, reorder, edit,  
merge and split commits**

**343eed2** Fix typo in foo

**ba66794** Add bar

**90328f9** Add foo

```
$ git rebase --interactive master
```

```
1 pick 90328f9 Add foo
2 pick ba66794 Add bar
3 pick 343eed2 Fix typo in foo
4
5 # Rebase c405e59..343eed2 onto c405e59 (3 commands)
6 #
7 # Commands:
8 # p, pick <commit> = use commit
9 # r, reword <commit> = use commit, but edit the commit message
10 # e, edit <commit> = use commit, but stop for amending
11 # s, squash <commit> = use commit, but meld into previous commit
12 # f, fixup <commit> = like "squash", but discard this commit's log message
13 # x, exec <command> = run command (the rest of the line) using shell
14 # b, break = stop here (continue rebase later with 'git rebase --continue')
15 # d, drop <commit> = remove commit
16 # l, label <label> = label current HEAD with a name
17 # t, reset <label> = reset HEAD to a label
```

```
1 pick 90328f9 Add foo
2 pick 343eed2 Fix typo in foo
3 pick ba66794 Add bar
4
5 # Rebase c405e59..343eed2 onto c405e59 (3 commands)
6 #
7 # Commands:
8 # p, pick <commit> = use commit
9 # r, reword <commit> = use commit, but edit the commit message
10 # e, edit <commit> = use commit, but stop for amending
11 # s, squash <commit> = use commit, but meld into previous commit
12 # f, fixup <commit> = like "squash", but discard this commit's log message
13 # x, exec <command> = run command (the rest of the line) using shell
14 # b, break = stop here (continue rebase later with 'git rebase --continue')
15 # d, drop <commit> = remove commit
16 # l, label <label> = label current HEAD with a name
17 # t, reset <label> = reset HEAD to a label
```

```
1 pick 90328f9 Add foo
2 fixup 343eed2 Fix typo in foo
3 pick ba66794 Add bar
4
5 # Rebase c405e59..343eed2 onto c405e59 (3 commands)
6 #
7 # Commands:
8 # p, pick <commit> = use commit
9 # r, reword <commit> = use commit, but edit the commit message
10 # e, edit <commit> = use commit, but stop for amending
11 # s, squash <commit> = use commit, but meld into previous commit
12 # f, fixup <commit> = like "squash", but discard this commit's log message
13 # x, exec <command> = run command (the rest of the line) using shell
14 # b, break = stop here (continue rebase later with 'git rebase --continue')
15 # d, drop <commit> = remove commit
16 # l, label <label> = label current HEAD with a name
17 # t, reset <label> = reset HEAD to a label
```

**4a14d7b** Add bar  
**c296093** Add foo

**Make your progress clear  
by rewriting your history  
to tell a simple story**



## **To recap**

- 1. Plan your commits**
- 2. Use single purpose branches**
- 3. Make atomic commits**
- 4. Write good commit messages**
- 5. Rewrite your history to tell a story (early and often)**

```
$ git cherry-pick
```

```
$ git add --patch
```

```
$ git rebase --interactive
```

**Following these 5 habits  
will free up your brain and  
help you break work into  
small steps**



# As an added bonus...

**It will benefit you and  
your team by providing  
you with documentation  
for all your code**

**“Every line of code is  
always documented”**

- Mislav Marohnić

<https://mislav.net/2014/02/hidden-documentation/>



# Thank you

**Joel Chippindale - CTO at Unmade - @joelchippindale**

**Thanks to my teams at Unmade, FutureLearn and Econsultancy who have all helped develop and refine these habits.**

**We are hiring, so come and work with us at Unmade**  
**<https://www.unmade.com/careers/>**