Git Basics - The Ball Game

Setup

1. Set up an account on GitHub
2. Install ***git*** on your computer.
3. Connect ***git*** to GitHub using the following commands in the terminal (You only need to do this the first time)

```
git config --global user.name "Your Name"
git config --global user.email "me@example.com"
```

4. Install the GitLens VSCode extension.
5. Install the github1s Linker Chrome extension

Create a new empty folder and initialize a ***git*** repository inside it.
When you see this symbol ⇩ - it's time to make a commit.
Make sure to comply with commit message conventions.

- Start with a capital letter
- Use the imperative mood
- Maximum of 50 characters in the message
- No punctuation

Lesson Materials

Link to a Step by Step Guide	Link to the presentation

coding_
academy

Part I

The first part of this exercise was designed to walk you through several common scenarios and procedures which you will probably encounter in your day to day development workflow with ***git***. Follow These steps closely.

1. Add an ***index.html*** file with an `<h1>` heading - The Ball Game.
2. Initialize a local repo and make the initial commit. ⇩
3. Add a `<div class="ball">100</div>`. ⇩
4. Link a ***styles.css*** file and set the page's background color to black and the heading's text color to white. ⇩
5. Style the `<div>` to be a circle with a width & height of 100px and some initial background color.
6. Add a main.js file with a click event handler - `onBallClick()` , which is activated when the `<div>` is clicked and prints a message to the console.
7. Commit the changes to the CSS file only (not the script file). ⇩
8. We forgot to center the ball horizontally in the page. Add margins to accomplish this and use the ***commit amend*** option to redo the last commit. ⇩
9. Commit the new JavaScript file and the changes to the HTML. ⇩
10. Change `onBallClick()` to increment the ball's width & height by 50px and display the updated diameter inside the ball. Remove the `console.log()`. ⇩
11. Add more styling rules: center the text inside the div, set the font size to 22px and the font weight to bold. ⇩
12. Travel through the snapshots in your repository and see how its features were gradually added. Use the diff editor to see the additions and changes to your files. **Don't forget to checkout the main branch again when you are done**!
13. Limit the ball size diameter to 400px. If the diameter grows beyond 400px, reset it to 100px. Add a 1 second transition to make the ball change size smoothly. ⇩
14. Undo the last commit and split it into two separate commits. The first one should only include the changes to the JavaScript ⇩ and the second should only include the changes to the CSS ⇩.
15. Add a ***my-notes.txt*** file and a ***.gitignore*** file to the project. Include the ***my-notes.txt*** file in the ***.gitignore*** file and add any other files which may have

automatically been generated by vscode or your system to it (this happens sometimes and depends on your OS & settings).

16. Publish your work to a remote repository on GitHub. Open the remote repository on GitHub and have a look at the code. If you haven't done so already, install the Open in GitHub1S extension and use it to browse the code in the online version of VSCode.
17. Back in your regular local version of VSCode, you can follow the links to older commits in the remote. Try it out.
18. Clone your remote repository in a new empty folder. Open it and continue your work from there.
19. Add a ***util.js*** file and add a `getRandomInt()` function inside it.
20. Change `onBallClick()` to increment the ball's diameter by a random amount between 20px and 60px.
21. Add a `getRandomColor()` function to the ***util.js*** file and change the ball's color to a random color each time it is clicked.
22. Sync your changes to the remote repo. Navigate to the remote to see that all changes from the clone have been synced to it.
23. Add another ball to the HTML. Give it a different class name and style it with a different initial color. Don't commit the changes yet!
24. Switch to an older commit - you get an error because your working tree isn't clean and ***git*** doesn't want to overwrite it.
25. Stash the changes and now try switching to an older commit again. This should work.
26. Switch back to the main branch and pop the stash.
27. Commit the changes in the working tree.
28. Add a *`maxDiameter`* parameter to `onBallClick()` which will be used as an upper limit to the ball's size. If a ball reaches its diameter limit, its size is reset to 100px. In the HTML file, pass different values as arguments to the function calls of each ball. Commit the changes and sync the remote.
29. Bonus: Open the remote repo on GitHub and set up GitHub pages to run the project.

Part II - Some more features

This part of the exercise, describes some more features which you will need to implement, but here, how to use ***git*** is not rigidly dictated and is left for you to decide upon. In other words, you decide when to commit changes, use the stash, sync with the remote, etc…

Please keep in mind that the main objective of the entire exercise is to familiarize you with ***git***, not with HTML, CSS & Javascript, so try to explore the various tools and operations which ***git*** provides.

- When an opportunity to refactor your code arises, take advantage of it and use ***git*** to take snapshots of the project before, during and after the refactoring. Try to keep a working, stable version of your code committed during the process.
- If you have bugs, try to time travel through your project's history to see when they first appeared. This may help to point you to their cause.
- Above all, remember that ***git*** is a set of tools and the more skilled you become in using them, the more benefit you will gain from them - so try to experiment.

1. Add a third ball. When it is clicked, swap the colors and sizes of the first two balls
2. Add a fourth ball. When it is clicked the diameter of the first two balls is reduced by a random amount between 20px and 60px. Limit their minimum diameter to 100px.
3. Add a fifth ball. When it is clicked, the page's background color changes to a random color.
4. Add a sixth ball. When it is clicked the entire game will reset to its original state.
5. When the sixth ball is hovered over for more than 2 seconds, an interval will start to run the mouse-click handlers of the first four balls every 2 seconds.
6. The interval will be cleared when the mouse cursor leaves the ball or after 10 cycles.
7. Add two buttons at the top of the page for undo and redo which allow the user to travel through the various states of the game (ball sizes, colors, etc…).
8. Make the undo and redo buttons disabled or enabled as required.
9. Add a counter to the page's title with the number of moves made in the game.
10. Add a timer which starts to run as soon as any change occurs in the page.

This should give you plenty of opportunities to try out various ***git*** features and workflows.

Happy Coding :-)

{coding_ academy