Web Programming (CS-A)
CS 406

Lecture 3

Shamsa Abid

Agenda

- HTML
- Git
- HTML Assignment

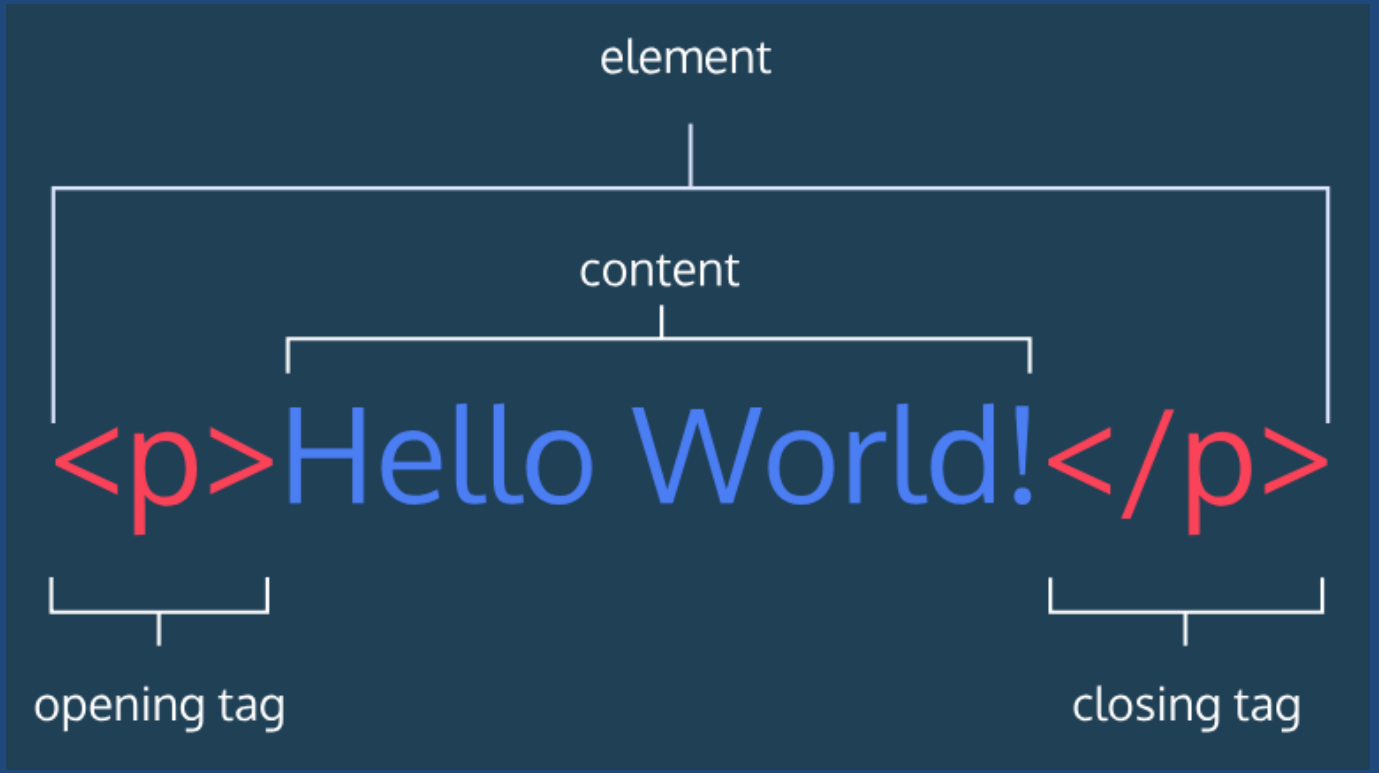
HTML

- HyperText Markup Language:
  - A markup language is a computer language that defines the structure and presentation of raw text.
  - In HTML, the computer can interpret raw text that is wrapped in HTML elements.
  - HyperText is text displayed on a computer or device that provides access to other text through links, also known as hyperlinks.

Try it

```
<h1>shamsa</h1>
```

HTML Anatomy



HTML Tags

- **The Body**
  - One of the key HTML elements we use to build a webpage is the *body* element. Only content inside the opening and closing body tags can be displayed to the screen.

```
<body>
  <p>"Life is very short and what we have to do must be done in the now." - Audre Lorde
  </p>
</body>
```

HTML Structure

- HTML is organized as a collection of family tree relationships.
- we placed <p> tags within <body> tags.
- When an element is contained inside another element, it is considered the *child* of that element.
- The child element is said to be *nested* inside of the *parent* element.

HTML Heirarchy

```
<body>
  <div>
    <h1>Sibling to p, but also grandchild of body</h1>
    <p>Sibling to h1, but also grandchild of body</p>
  </div>
</body>
```

HTML Heirarchy

Understanding HTML hierarchy is important because child elements can inherit behavior and styling from their parent element.

Headings

- In HTML, there are six different *headings*, or *heading elements*.
- The following is the list of heading elements available in HTML. They are ordered from largest to smallest in size.
  - <h1> — used for main headings. All other smaller headings are used for subheadings.
  - <h2>
  - <h3>
  - <h4>
  - <h5>
  - <h6>

Headings

- <body>
- <h1>The Brown Bear</h1>
- <h2>About Brown Bears</h2>
- <h3>Species</h3>
- <h3>Features</h3>
- <h2> Habitat </h2>
- </body>

Divs

- is short for "division" or a container that divides the page into sections.
- These sections are very useful for grouping elements in your HTML together.

Divs

- s can contain any text or other HTML elements, such as links, images, or videos.
- Remember to always add two spaces of indentation when you nest elements inside of s for better readability.

Divs

```
<body>
 <h1>The Brown Bear</h1>
 <div>
  <h2>About Brown Bears</h2>
  <h3>Species</h3>
  <h3>Features</h3>
 </div>
 <div>
  <h2>Habitat</h2>
  <h3>Countries with Large Brown Bear Populations</h3>
  <h3>Countries with Small Brown Bear Populations</h3>
 </div>
 <div>
        <h2>Media</h2>
 </div>
</body>
```

Attributes

- If we want to expand an element's tag, we can do so using an *attribute*.
- Attributes are content added to the opening tag of an element and can be used in several different ways, from providing information to changing styling.
- Attributes are made up of the following two parts:
  - The *name* of the attribute
  - The *value* of the attribute
-

The id attribute

- We can use the id attribute to specify different content (such as s) and is really helpful when you use an element more than once.
- When we add an id to a , we place it in the opening tag:

```
<div id="intro"> <h1>Introduction</h1> </div>
```

```
<body>
  <h1>The Brown Bear</h1>
  <div id = "introduction">
    <h2>About Brown Bears</h2>
    <h3>Species</h3>
    <h3>Features</h3>
  </div>
  <div id = "habitat">
    <h2>Habitat</h2>
    <h3>Countries with Large Brown Bear Populations</h3>
    <h3>Countries with Small Brown Bear Populations</h3>
  </div>
  <div id = "media">
    <h2>Media</h2>
  </div>
</body>
```

Displaying Text

- use a *paragraph* or *span*:
  - *Paragraphs* (<p>) contain a block of plain text.
  - contains short pieces of text or other HTML. They are used to separate small pieces of content that are on the same line as other content.

-
- <h1>Technology</h1>
-
-
- Self-driving cars are anticipated to replace up to 2 million jobs over the next two decades.
-
-

Styling Text

- The tag emphasizes text, while the
- The tag will generally render as *italic* emphasis.
- The will generally render as **bold** emphasis.

```
<p>
<strong>The Nile River</strong> is the
    <em>longest</em> river in the world measuring over
    6,850 kilometers long (approximately 4,260 miles).
</p>
```

Line Breaks

-
- The line break element is unique because it is only composed of a starting tag. You can use it anywhere within your HTML code and a line break will be shown in the browser.

Unordered Lists

- *unordered list* tag (<ul>)
  - to create a list of items in no particular order.
  - An unordered list outlines individual *list items* with a bullet point.
  - The <ul> element should not hold raw text
    - Individual list items must be added to the unordered list using the tag. The <li> or list item tag is used to describe an item in a list.

Unordered Lists

```
<ul>
    <li>Limes</li>
    <li>Tortillas</li>
    <li>Chicken</li>
</ul>
```

Ordered Lists

- each list item is numbered.

```
<ol>
<li>Preheat the oven to 350 degrees.</li>
<li>Mix whole wheat flour, baking soda, and salt.</li>
<li>Cream the butter, sugar in separate bowl.</li>
<li>Add eggs and vanilla extract to bowl.</li>
</ol>
```

Images

- <img> tag is a *self-closing tag*.
- Note that the end of the <img> tag has a forward slash /.
- Self-closing tags may include or omit the final slash — both will render properly.

```
<img src="image-location.jpg" />
```

the value of src must be the *uniform resource locator* (URL) of the image. A URL is the web address or local address where a file is stored.

Image Alts

- what happens when assistive technologies such as screen readers come across image tags?
- The alt attribute means alternative text
- The alt attribute can be added to the image tag just like the src attribute.
- The value of alt should be a description of the image.
- `<img src="#" alt="A field of yellow sunflowers" />`

Videos

- <video src="myVideo.mp4" width="320" height="240" controls> Video not supported </video>
- The controls attribute instructs the browser to include basic video controls: pause, play and skip.
- The text, "Video not supported", between the opening and closing video tags will only be displayed if the browser is unable to load the video.

HTML DOCUMENT STANDARDS

PREPARING FOR HTML

Preparing for HTML

- HTML files require certain elements to set up the document properly.
- You can let web browsers know that you are using HTML by starting your document with a *document type declaration*.
- <!DOCTYPE html>
- This declaration is an instruction, and it must be the first line of code in your HTML document.
- It tells the browser what type of document to expect, along with what version of HTML is being used in the document.
- For now, the browser will correctly assume that the html in <!DOCTYPE html> is referring to HTML5.

The <html> tag

- <!DOCTYPE html> <html> </html>
- Anything between the opening <html> and closing </html> tags will be interpreted as HTML code.

The Head

- The <head> element contains the *metadata* for a web page.
- Metadata is information about the page that isn't displayed directly on the web page.

Page Titles

- A browser's tab displays the title specified in the <title> tag. The <title> tag is always inside of the <head>.

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Coding Journal</title>
  </head>
</html>
```

Linking to Other Web Pages

- *anchor* element <a>
  - You can add links to a web page by adding this and including the text of the link in between the opening and closing tags.
- href attribute
  - This attribute stands for *hyperlink reference* and is used to link to a *path*, or the address to where a file is located

```
<a href="https://www.wikipedia.org/">This Is A Link To Wikipedia</a>
```

Opening Links in a New Window

- The target attribute specifies how a link should open
- For a link to open in a new window/tab, the target attribute requires a value of _blank

```
<a href="https://en.wikipedia.org/wiki/Brown_bear" target="_blank">The Brown Bear</a>
```

Linking to Relative Page

- If the files are stored in the same folder, we can link web pages together using a *relative path*.
- Contact
- The ./ in ./index.html tells the browser to look for the file in the current folder.

Linking At Will

- HTML allows you to turn nearly any element into a link by wrapping that element with an anchor element.
- With this technique, it's possible to turn images into links by simply wrapping the <img> element with an <a>element.

-
-
-

Linking to Same Page

- When users visit our site, we want them to be able to click a link and have the page automatically scroll to a specific section.
- In order to link to a *target* on the same page, we must give the target an *id*, like this:
- This is the top of the page! <h1 id="bottom">This is the bottom! </h1>
- The target link is a string containing the # character and the target element's id

```
<ol>
<li><a href="#top">Top</a></li>
<li><a href="#bottom">Bottom</a></li>
</ol>
```

Indentation

- The World Wide Web Consortium, or W3C, is responsible for maintaining the style standards of HTML.
- At the time of writing, the W3C recommends 2 spaces of indentation when writing HTML code.

Comments

- <!-- This is a comment that the browser will not display. -->

Tables(rows,headings,data)

```
<table>
    <tr>
        <th></th>
         <th scope="col">Saturday</th>
        <th scope="col">Sunday</th>
    </tr>
    <tr>
        <th scope="row">Temperature</th>
        <td>73</td>
        <td>81</td>
    </tr>
</table>
```

Table Borders

- <table border="1"> <tr> <td>73</td> <td>81</td> </tr> </table>
- Deprecated, use this instead

```
table, td {
    border: 1px solid black;
    }
```

Spanning Columns

- Data can span columns using the colspan attribute. The attributes accepts an integer (greater than or equal to 1) to denote the number of columns it spans across.

```
<table>
    <tr>
        <th>Monday</th>
        <th>Tuesday</th>
        <th>Wednesday</th>
    </tr>
    <tr>
        <td colspan="2">Out of Town</td>
         <td>Back in Town</td>
    </tr>
</table>
```

Spanning Rows

- The rowspan attribute is used for data that spans multiple rows

```
<table>
  <tr> <!-- Row 1 -->
    <th></th>
    <th>Saturday</th>
    <th>Sunday</th>
  </tr>
  <tr> <!-- Row 2 -->
    <th>Morning</th>
    <td rowspan="2">Work</td>
    <td rowspan="3">Relax</td>
  </tr>
  <tr> <!-- Row 3 -->
    <th>Afternoon</th>
  </tr>
  <tr> <!-- Row 4 -->
    <th>Evening</th>
    <td>Dinner</td>
  </tr>
</table>
```

Table Body and Table Head

- Long tables can be sectioned off using the *table body* element: <tbody>.
- The <tbody> element should contain all of the table's data, excluding the table headings
- We section off the table's headings using the <thead> element.

Table Footer

- The bottom part of a long table can also be sectioned off using the <tfoot> element.
- ….

```
</tbody>
    <tfoot>
        <tr>
            <th>Total</th>
            <td>$22M</td>
            <td>$12.5M</td>
        </tr>
    </tfoot>
</table>
```

Table css

```
table, th, td {
border: 1px solid black;
font-family: Arial, sans-serif;
text-align: center;
}
```

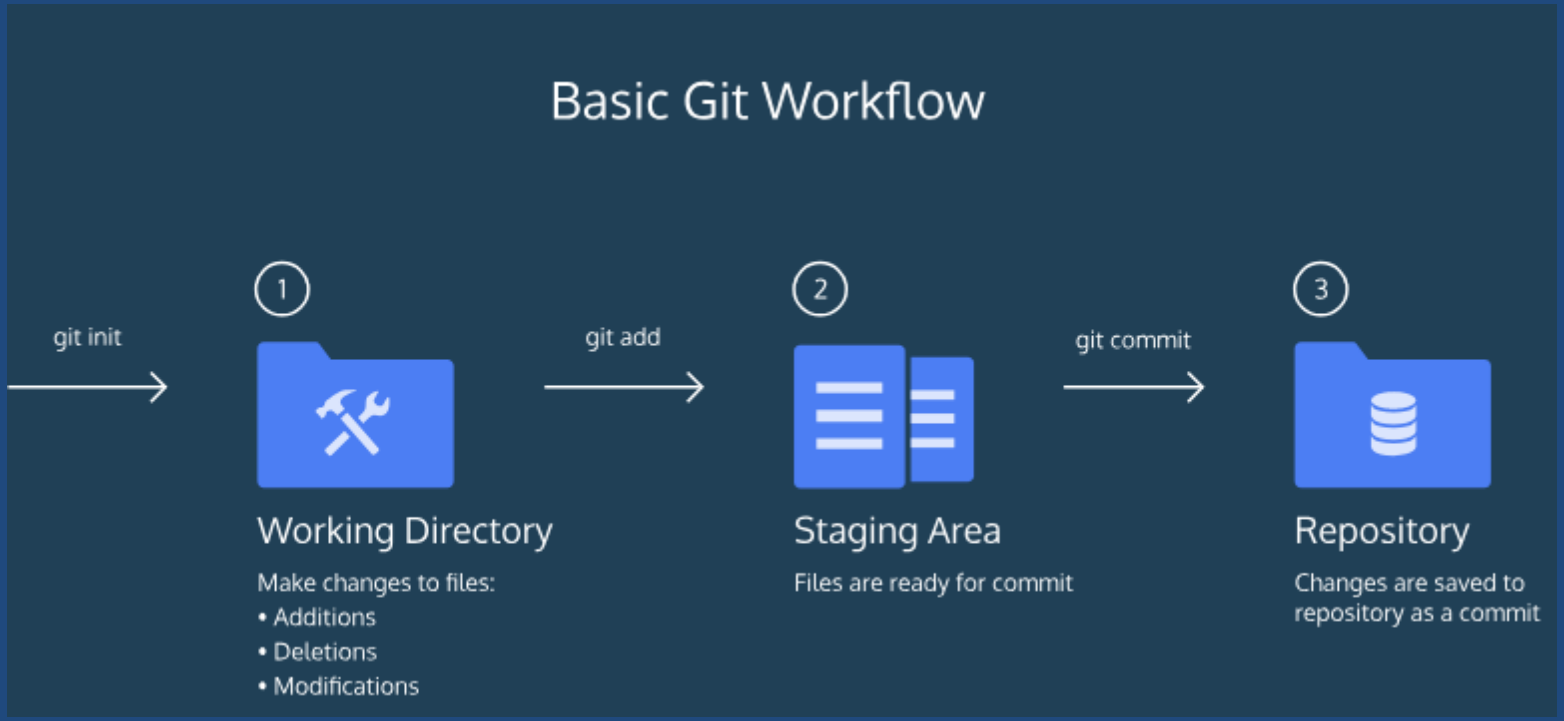
Assignment 2: Deadline ?? Wednesday 11:59 pm

- Make a personal page using the HTML you've learned so far and call it index.html
- Go to https://pages.github.com/
- Follow the instructions to create your own site using that page
- Your personal site (CV based) will be up at the url https://*yourusername*.github.io/
- I will send you an invitation to accept Assignment 2
- Next follow the instructions to create a project site
- Choose your Assignment 2 repo as the repo for which you want to build a page. Choose a theme and edit the content.
- In the first line of the content (the readme) add the two clickable URLS
  - Link to personal site: <your url>
  - Link to project site: <project site url>
- Add a few lines about the purpose of the Assignment, the difficulty/enjoyment level, some images of web development jokes etc\
- Grading Rubric: The more usage of various html elements, the better.

Git

GIT

Git Workflow



Git Commands

- Git is the industry-standard version control system for web developers
- Use Git commands to help keep track of changes made to a project:
  - git init creates a new Git repository
  - git status inspects the contents of the working directory and staging area
  - git add adds files from the working directory to the staging area
  - git diff shows the difference between the working directory and the staging area
  - git commit permanently stores file changes from the staging area in the repository
  - git log shows a list of all previous commits

Backtracking Intro

- When working on a Git project, sometimes we make changes that we want to get rid of.
- Git offers a few eraser-like features that allow us to undo mistakes during project creation.

head commit

- In Git, the commit you are currently on is known as the HEAD commit. In many cases, the most recently made commit is the HEAD commit.
- git show HEAD
  - The output of this command will display everything the git log command displays for the HEAD commit, plus all the file changes that were committed.

git checkout

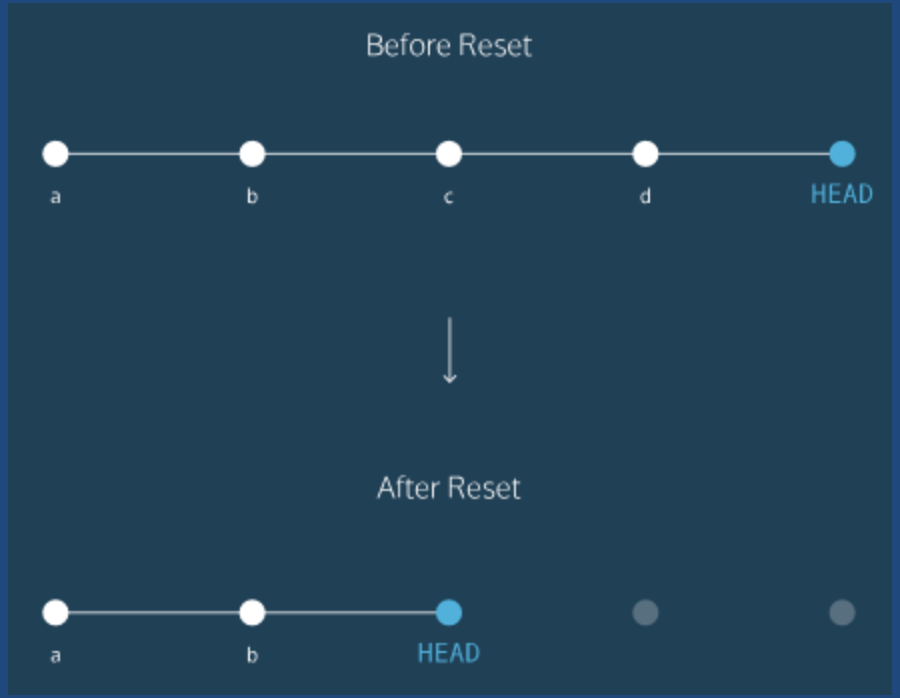
- git checkout HEAD filename
  - will restore the file in your working directory to look exactly as it did when you last made a commit.

git reset

- We can *unstage* a file from the staging area using
- git reset HEAD filename
  - This command *resets* the file in the staging area to be the same as the HEAD commit. It does not discard file changes from the working directory, it just removes them from the staging area.

git reset

- Creating a project is like hiking in a forest. Sometimes you take a wrong turn and find yourself lost.
- Git enables you to rewind to the part before you made the wrong turn.
  - git reset commit_SHA
- This command works by using the first 7 characters of the SHA of a previous commit. For example, if the SHA of the previous commit is 5d692065cf51a2f50ea8e7b19b5a7ae512f633ba, use:
- git reset 5d69206
  - HEAD is now set to that previous commit.


Before Reset
a
b
c
d
HEAD
After Reset
a
b
HEAD

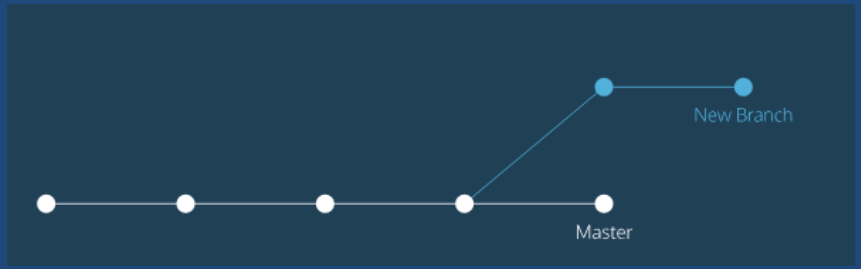
- git checkout HEAD filename: Discards changes in the working directory.
- git reset HEAD filename: Unstages file changes in the staging area.
- git reset commit_SHA: Resets to a previous commit in your commit history.

GIT

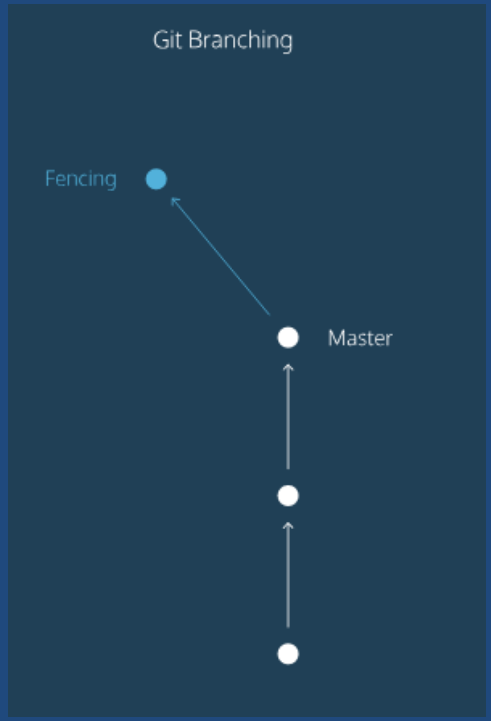
BRANCHING

git branch

- Up to this point, you've worked in a single Git branch called master.
- Git allows us to create *branches* to experiment with versions of a project.
- >git branch
  - Displays the name of the current branch



- To create a new branch, use:
  - git branch new_branch
- You can switch to the new branch with
  - git checkout branch_name



git merge

- What if you wanted include all the changes made to the new branch on he master branch?
  - git merge branch_name
  - if I wanted to merge the skills branch to master, I would enter
    - git merge skills
    - But first do git checkout master and then execute the merge command

merge conflict

- What would happen if you made a commit on master *before* you merged the two branches?
- What if the commit you made on master altered the same exact text you worked on in the new branch?
- When you switch back to master and ask Git to merge the two branches, Git doesn't know which changes you want to keep.
  - This is called a *merge conflict*.

delete branch

- the end goal of a new branch is to merge that feature into the master branch.
- After the branch has been integrated into master, it has served its purpose and can be deleted.
- The command
  - git branch -d branch_name
  - will delete the specified branch from your Git project.

Git branch workflow summary

- The following commands are useful in the Git branch workflow.
  - git branch: Lists all a Git project's branches.
  - git branch branch_name: Creates a new branch.
  - git checkout branch_name: Used to switch from one branch to another.
  - git merge branch_name: Used to join file changes from one branch to another.
  - git branch -d branch_name: Deletes the branch specified.

git

TEAMWORK

- A remote
  - a shared Git repository that allows multiple collaborators to work on the same Git project from different locations. Collaborators work on the project independently, and merge changes together when they are ready to do so.
  - git clone remote_location clone_name
    - clone_name is the name you give to the directory in which Git will clone the repository.

git fetch

- What if your clone is no longer up-to-date.
- An easy way to see if changes have been made to the remote and bring the changes down to your local copy is with:
  - git fetch

git merge

- Even though masters new commits have been fetched to your local copy of the Git project, those commits are on the origin/master branch. Your *local* master branch has not been updated yet, so you can't view or make changes to any of the work added.
- we'll use the git merge command to integrate origin/master into your local master branch. The command:
  - git merge origin/master

- The workflow for Git collaborations typically follows this order:
  1. Fetch and merge changes from the remote
  2. Create a branch to work on a new project feature
  3. Develop the feature on your branch and commit your work
  4. Fetch and merge from the remote again (in case new commits were made while you were working)
  5. *Push* your branch up to the remote for review
- Steps 1 and 4 are a safeguard against *merge conflicts*

git push

- The command:
  - git push origin your_branch_name
- will push your branch up to the remote, origin

review

- git clone: Creates a local copy of a remote.
- git remote -v: Lists a Git project's remotes.
- git fetch: Fetches work from the remote into the local copy.
- git merge origin/master:
  Merges origin/master into your local branch.
- git push origin <branch_name>: Pushes a local branch to the origin remote.

Questions?

One Devloper Army
@OneDevloperArmy

If you are a new programmer I just want you to know

me and all of my colleagues with years of experience

Google the most basic things _daily_

12:18 PM · 13 Aug 18

View Tweet activity

Next

- CSS

References