HTML5 & CSS3

Week 1

Welcome!

This is a collaborative learning environment. We have a very large class, made up of students with a wide variety of technical experience. I expect you to work together whenever possible to solve problems and learn new things. Great code comes from great teams.

Learning Goals

- Strong understanding of HTML and CSS fundamentals
- Ability to write semantic, valid, and reusable HTML and CSS
- Ability to translate design mockups into working code
- Understanding of web accessibility
- Understanding of Javascript and programming basics
- Understanding of professional development best practices

Agenda

1- Introductions 45 mins
2- Canvas, Adobe Connect 15 mins
3- Class policies, syllabus 10 mins
4- HTML! 30 mins
5- Git & Github 60 mins
6- Homework 10 mins

Introductions

Cheri Allen

Web Developer at Northwest Independent Ruby Development (NIRD)

cherimallen@gmail.com

Introductions

Your turn!

Name

Your goals for this class

Canvas

You should have an email invitation to your @uw.edu address

or https://canvas.uw.edu/courses/954712

Use for:

- Viewing and turning in assignments
- Viewing class notes, slides, vital info
- Participating in class discussion forum

Don’t use for:

- Contacting me- use email instead

HTML!

Introduction, Semantic Tags, and Work Flow

What is HTML?

HyperText Markup Language is the standard markup language used to create web pages.

Created by Tim Berners-Lee, released 1991.

Is the basis of **every site** on the web.

Web Development Tools

Text Editor - write plain text documents (class default is Brackets)

Browsers - view your work in several different ones, they all interpret HTML little differently

Version Control - save your work in a logical and organized way

Host - saves your HTML files on their web servers, so your files can be publicly available

File Structure

We'll all use the same file structure for this class. Place your top level directory anywhere that is not a Git repository and is easy to find.

Top level directory: **foundations**

Directories within foundations: **week1**, **week2**, [etc]

Each **week*** directory will contain files and other directories.

Let’s Write Some Code!

- Within **week1**, create a **duck_example** directory
- Open Brackets (or your text editor of choice)
- Create a new file, save it as **foundations/week1/duck_example/index.html** (once it’s saved with .html extension, editor will offer syntax highlighting)

Minimal Structure for HTML Document

Minimum HTML Structure

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8">
5    <title>Cool Site Title</title>
6  </head>
7  <body>
8
9  </body>
10 </html>
```

Semantic Elements

Semantic elements are those whose tags say something about the kind of content they contain. <h1>, <article>, <nav>, and <footer> are great examples.

Semantic Elements

Using semantic tags allows search engines to understand your site’s content better, and allows screen readers to represent your site’s content more accurately to users. It also makes writing HTML easier for you!

Style Notes

Browsers don’t see whitespace and indentation in HTML, but humans sure do. Keep your code readable and easy to debug by always indenting clearly.

```
<nav class="funfont">
  <ul class="navigation">
    <li><a href="/" title="about">About</a></li>
    <li><a href="cv" title="CV">CV</a></li>
    <li><a href="slides" title="slides">Slides</a></li>
    <li><a href="partytime" title="partytime">Party Time</a></li>
  </ul>
</nav>
```

Two Types of Elements

1. **Standard** have opening and closing tags

     <nav></nav>

   <footer></footer> <h2></h2>

2. **Void** have only a single tag

  <img>

   <input> <meta>

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Duck Info Sheet</title>
</head>
<body>
  <h1>Ducks Are Really Neat</h1>

  <nav>
    <a href="#">Duck Gallery</a>
    <a href="#">Duck Watching Journal</a>
    <a href="#">Interview With a Duck</a>
  </nav>

  <article>
    <p>Ducks are noble and majestic creatures.</p>

    <p>No one knows the origin of ducks, but it is hypothesized that they are the result of lightning striking an eagle.</p>
  </article>

  <footer>
    <p>Coypyright 2014 by Cheri the Duck Expert.</p>
  </footer>
</body>
</html>
```

View File In Browser

Open your browser of choice.

In browser menu, choose File -> Open File, then select your .html file.

See code: right click, choose “view source”.

Git & Github

Free, Awesome Version Control

Version Control

Version control is the management of changes to files or projects. It allows you to easily revert a document back to an earlier state.

Why Use Git?

1-It's distributed, works great for teams

2- It's got tons of documentation and support

3- It's open source

4- It's fast

5- It doesn't require any network connection

What exactly does Git do?

Tracks the changes in files- not the files themselves!

It works a lot like “save points” in a video game.

Keeps full history of a document. Once content is committed, it is there forever. *

What does Github do?

It's a hosting site for software development projects that use Git. It has paid and free memberships. Free ones require that you only have open source (visible to everyone) projects.

It offers a variety of social networking functions, like following and RSS feeds, based around repositories.

Using these Slides

Command line entries:

    indented blue text

Should be replaced with real information:

    [text in brackets]

Output expected in the command line:

    indented green text

Using Git

Download: git-scm.com/downloads

Installing Git

https://help.github.com/articles/set-up-git

Choose your OS (small menu at top), then follow instructions.

Windows: Please choose **default** installation options.

Windows vs *ux

Mac and Linux Terminals use bash, Windows Command Prompt uses cmd. There are slight differences in syntax and functionality between them.

For this class, Windows users will be using the **Git Bash** tool, which uses bash syntax, not Command Prompt.

Line endings are different in two systems. Win users configured Git to commit with Unix style line endings. This saves you a lot of errors when working on a project with Mac or Linux users.

Bash command line basics

.	the directory you are currently in
..	the directory one level up
~	your home directory (only for Mac & Linux)
cd [place]	change directory to [place]
clear	clear terminal screen
help	lists possible commands
man [command]	shows manual for indicated command
ls	lists contents of present directory
pwd	lists present working directory

Using Windows? Work in the GitBash window, not cmd, and you'll use the same syntax as Mac or Linux.

Configure Git

In Terminal or Git Bash:

```
git config --global user.email "[email]"
git config --global user.name "[name]"
```

Email should match your Github account. Name will attach to commits- be professional.

Confirm : `git config --list`

your name

your email

http://git-scm.com/docs/git-config

The Git Work Flow

command	what it does
git init	initializes a repo
do some work	
git add [filename]	stages changes
git commit -m " "	commits changes
git status	what's up, git?

In Terminal or Git Bash, navigate to your **foundations** directory.

```
cd [~/Desktop or wherever your directory is]
cd foundations
```

Navigate into **week1/duck_example**

```
cd week1/duck_example
```

Initialize it as a Git repository

```
git init
```

See what's up with Git

```
git status
```

Add your untracked file

```
git add index.html
```

Save this state

```
git commit -m "Initialize repo”
```

Sweet!

Commit Messages

Every commit must have a message. Make it:

- Precise and polite
- Short, 50 characters or less
- Begin with active, present-tense verb

GOOD:

"Fix the js bug causing login errors"

"Add config info to README"

"Add support for packager to build installers"

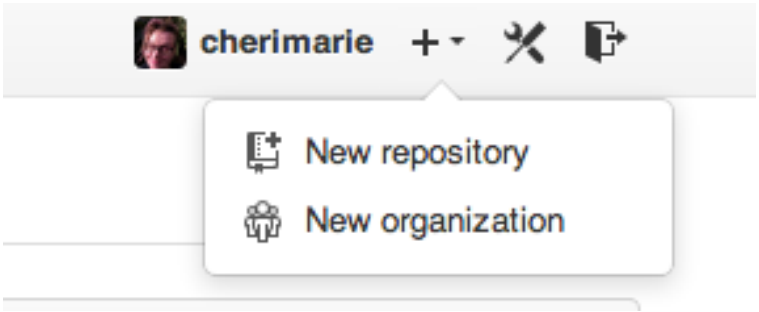
Welcome to Github

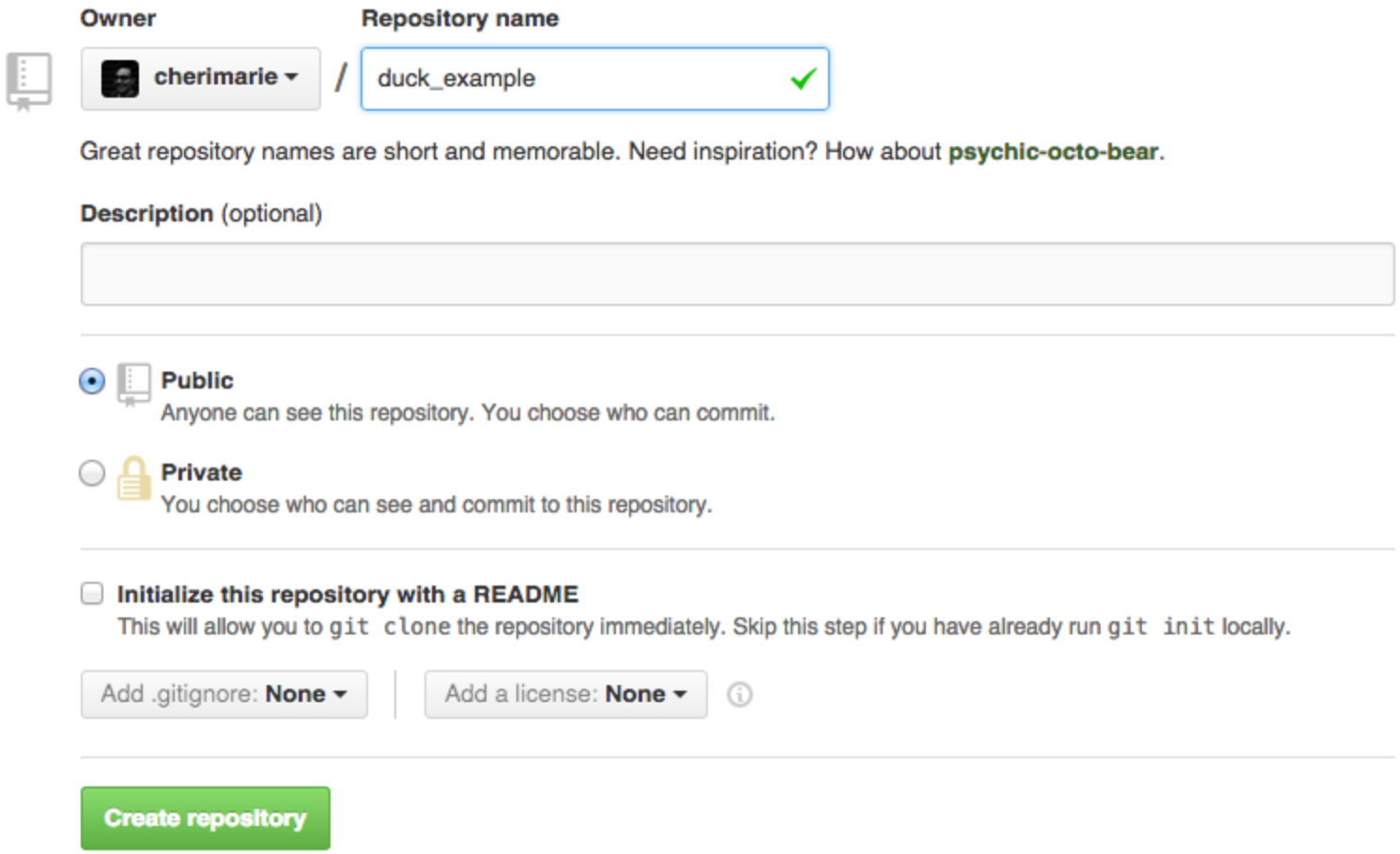
github.com

Octocat, Github's Mascot

Create a Github Repo

1. Log in to Github.com
2. Click the + button on top right of page
3. Choose “New Repository”



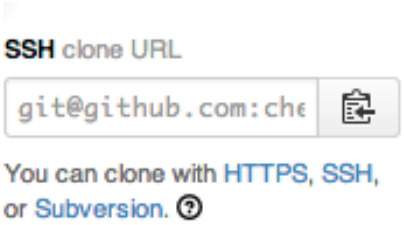

Owner
Repository name
cherimarie
/
duck_example
Great repository names are short and memorable. Need inspiration? How about psychic-octo-bear.
Description (optional)
Public
Anyone can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with a README
This will allow you to git clone the repository immediately. Skip this step if you have already run git init locally.
Add .gitignore: None
Add a license: None
Create repository

Create a Github Repo

4. Give it the same name as your local repo
5. Keep other defaults
6. Click “Create Repository”

Connect Local Repo to Github

1. From the page of your new Github repo, copy “clone URL”, in HTTPS or SSH format, as appropriate



Connect Local Repo to Github

2. In Terminal or Git Bash:

   [navigate to duck_example directory]

   git remote add origin [pasted clone URL]

   git push origin master

origin is what you are naming connection

master is the remote branch you're pushing to

SSH or HTTPS Remote?

HTTPS:

Good for insecure computers. You enter your Github name and password every time you push or pull, unless you have a credential helper. Very easy to set up.

SSH:

Good for secure computers. You generate a SSH key locally that identifies your machine to Github. Follow github tutorial to set up, it's not hard.

Good first class, team.

Homework

- “Assignment 1” in Canvas
- Due 8am next Thursday
- Submit link to a Github repository
- Trouble? Questions? Discuss it with your classmates in chat or the Canvas forums, or attend office hours to discuss it with Cheri.

Office Hours

Office hours are Mondays, 6-8pm, in the Hipchat room. If there is demand, I can also be available in person during those hours.