JavaScript You Don’t Know

Mari Good

CS 233JS

Topics

- JavaScript You Don’t Know
  - Advanced JS That We Never Showed You
  - JS Features that were added in the “most significant upgrade” of JS or ECMAScript, ES 2015 or ES 6.
- Git and github for source code version control
- Version 2 of StopWatch, Concentration and TicTacToe

CodePen – A JS Playground

- There are several JavaScript “playgrounds” on the internet. Each gives you a place to experiment with html, css and JS code in an online development environment
- Here’s an article that talks about some of some of the good ones https://linuxhint.com/top-five-javascript-playgrounds/
- I created a couple of “pens” on CodePen that you can use to experiment with some of the things in the articles on JS that I assigned for this week
  - Some of you are already familiar with replit.com. It’s a “playground” that can be used with all kinds of programming languages. If any of you choose to recreate the playgrounds that I’ve created in replit.com, please let the rest of us know and share the playgrounds with us.

Advanced JS That We Never Showed You

- https://codepen.io/goodmari/pen/QmbwJq
- Function Stuff – read this but don't be surprised if it doesn't make a ton of sense
    - Function expressions
    - Anonymous functions

Objects – this is important stuff. You'll use it all of the time.

- Object literals and object literal syntax
- Adding properties
- Accessing properties using [] operator
- Arrays of objects
- json

Newer JS Features You Should Know

- https://codepen.io/goodmari/pen/dmoPow
  - Variable scope
    - Function vs block scope
    - Var, let and const
  - Arrow functions (see the next slide for definition of functional programming)
  - The keyword this
  - Destructuring
  - Template literals
  - Classes
- http://kangax.github.io/compat-table/es6/ gives detailed information about browser support for newer features

Functional Programming – New in ES6

- Functional programming is a programming paradigm or style of programming that treats computation as the evaluation of expressions and avoids changing state or mutating data. It is a declarative paradigm, which means programming is done with expressions instead of statements. In functional code, the output value of a function depends only on the arguments that are passed to the function, so calling a function *f* twice with the same value for an argument *x* produces the same result *f(x)* each time; this is in contrast to void functions that depend on or mutate global state, which may produce different results at different times when called with the same arguments but a different program state. Eliminating these side effects is one of the key motivations for the development of functional programming.

Use ES6 Classes in Lab 2

- Lab 2 asks you to encapsulate all 3 of the applications from the first lab, StopWatch, Concentration and TTT, using ES6 style classes.
- I'll talk about concepts and syntax as we do the first problem together.
- I have also given you an article on JavaScript classes in moodle.
- The starting files are in a git repository rather than in a zip file. You'll be using git and github to manage your work.
  - I've given you a version 1 solution.
  - I've given you TONS of comments in the V2.js file that are intended to walk you through the process of doing this conversion.

Keeping Track of Your Work

- Professional programmers use version control software to manage source code and documents during the software development process.
- Git is currently the most popular version control system
  - Popularized by GitHub, largest cloud based source code host with 50+ million repositories
- Stores files and change information in a repository
  - Records changes to files
  - Allows specific versions to be recalled on demand

Before We Get Started

- If you don’t already have a github account, please create one now. Use the email you use for all LCC communication.
- If you haven’t already installed the git command line tools, please do that now.
  - We’re going to start with the command line tools because it’s easier for me to control the process if you type commands at the command line. AND because employers will want you to know the command line tools.
  - There are lots of GUI client tools and most IDEs have git support built in. Once you know the basics you can start to work with GUI tools.

Creating the Remote Repo(sitory) on github

- I’m using a tool called GitHub Classroom to help manage your git repositories. Every time I assign a lab, I’ll create a “template” repository that includes the starting files.
- Accept the GitHub Classroom assignment in moodle.
  - This will create a private GitHub repository for you and will take you to the repository page in GitHub.
    - You’ll work on your code on your local repository and PUSH it to the GitHub repository regularly.
    - You’ll give me the url for your GitHub repo when you want me to look at your code for grading or answering a question.
  - Click clone on the GitHub page. That will copy the url of your repository to the clipboard.

Creating a Local Repo(sitory) on Your Machine

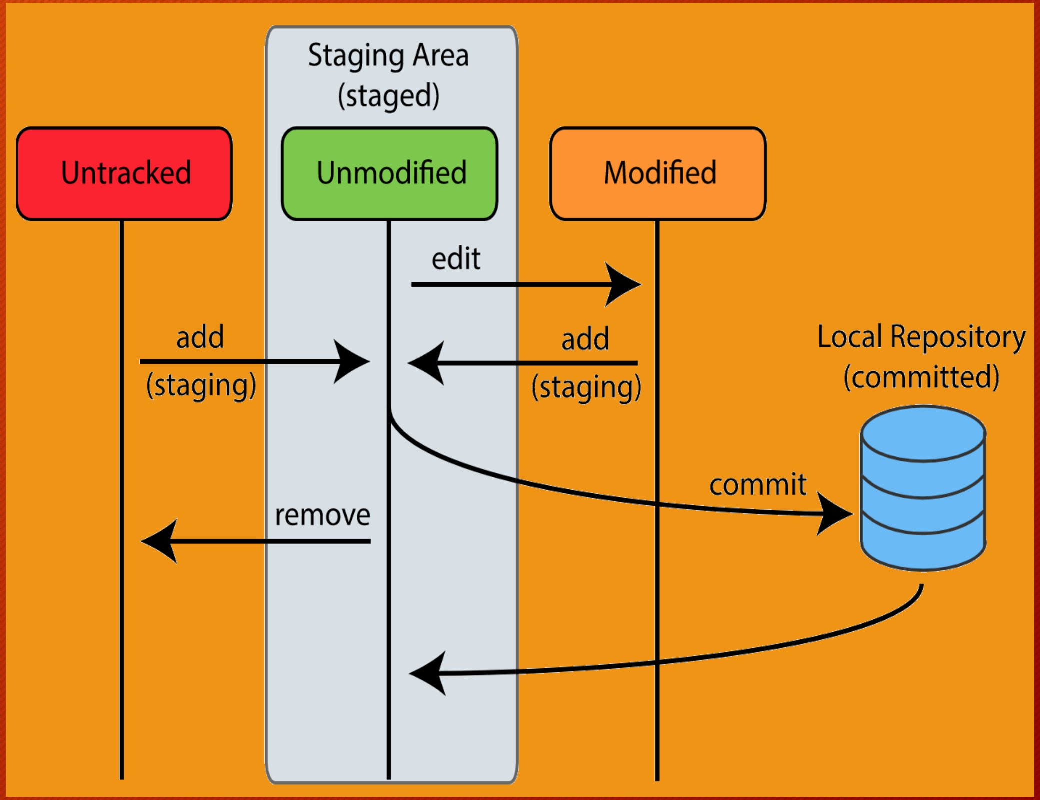
- Locate the folder on your machine where you store your files.
- Select git bash here from the context sensitive menu.
  - This will open a command window that you'll use to execute Git commands.
- Type git clone and paste the url from the clipboard. Press enter.
  - A folder will be created on your machine. The folder and all of the files in it will be managed by Git. This folder will contain your local repository for lab 2.
    - I often rename this folder before I do anything else.
  - Type cd followed by the name of the folder. This will take you into the folder.

Change Something in One of the Files

- Open the folder that contains all of your labs in VSCode
- Add some code (even if it’s just a comment) in at least 2 of the files.
- Save all.

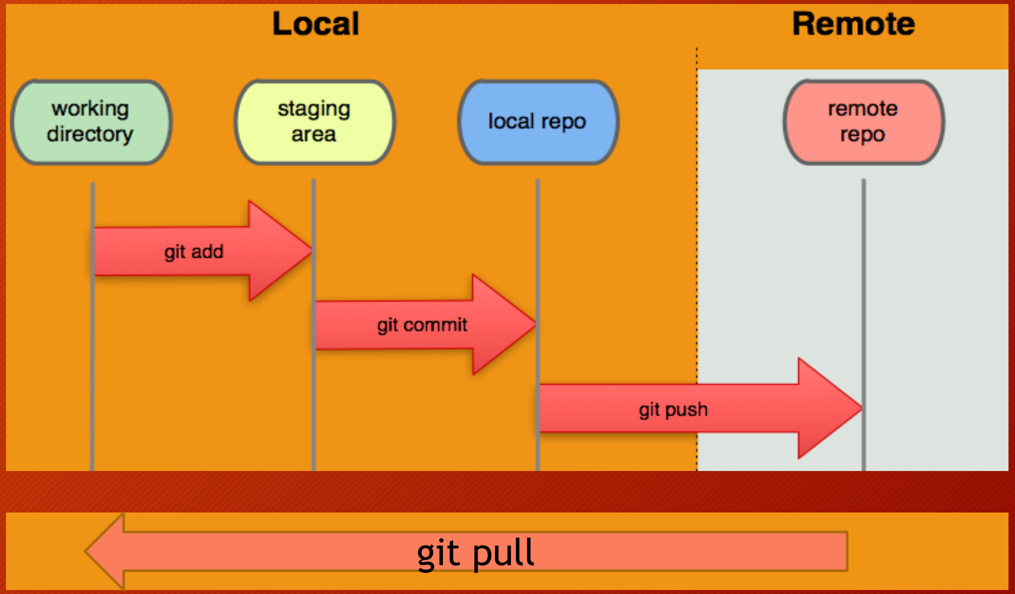
Git's Staging Area

- Git's main states for the files it manages are
  - Untracked - File exists but not yet added to repository
  - Modified - File changed but not committed
  - Staged - File marked as changed and ready for next commit
  - Committed - File stored safely in local repository database
- Type git status to see the status of your files



“Sync”ing your Local and Remote Repos

- Commands for working with remote repository
  - Push - Copy all local commits to remote repository
  - Pull - Copy remote commits to local repository and merge changes with local files. You shouldn’t need this because you’re the only one working on your lab.



Saving Your Changes to your Local Repo

- In the same command window type
  - git status
    - Git tells you that some of your files have changed
  - git add –A
    - Adds all of the files to the staging area
  - git commit –m “Added some comments”
    - Saves the changes to the repo database
  - git push
    - Saves the changes to the remote repo on GitHub. Check it out!
    - The first time you do this you might get an error. Type the 2 commands exactly as they appear in the error message. This will associate your local repo with the remote repo on GitHub.

Turning StopWatch into a Class

- I’ll use the comments in version 2 of the starting files to guide our discussion of both the syntax and the semantics of creating a class.
- There’s a link in moodle to an article about ES6 classes.
- The slide on the next page contains some vocabulary that you’ll learn as we’re working on the StopWatch application.

Vocabulary You Should Now Know

- Here’s a list of vocabulary that you should know as a result of our work on the ES6 version of the StopWatch application
  - Object oriented programming
  - Object
  - Class
  - Encapsulation
  - Instance variable
  - Constructor
  - Method
  - The new operator
  - The keyword this
  - The bind method

Working on Your Lab

- As you finish each part of each problem
  - git status
  - git add –A
  - git commit –m “some message”
  - git push
  - Check your files on GitHub
- When you ask a question, make sure that your most recent code is pushed to GitHub and give me the url for your GitHub repo.
- When you submit your lab you’ll give me that same url.

Concentration Version 2 – Clone the Repo

- Now repeat the process and convert Concentration to an object oriented ES6 style application.
  - Accept the github classroom invitation
  - Go to the github repo that’s created for you and copy the url
  - Using your file system, open a git bash window in your lab 2 folder
  - Type git clone and paste the url
  - You may want to rename the folder on your machine

Concentration Version 2 – Starting Files

- Let’s look at the starting files together
  - indexStart.html
  - indexV1.js – is the completed version from lab 1
  - indexStartV2.js – is the starting file for the ES6 Class version. I’ve given you some comments to walk you through the process for this application.
- You can replace your js file with mine or use my comments and your code as you do the conversion.

TTT Version 2 – Clone the Repo

- Now repeat the process and convert TTT to an object oriented ES6 style application.
  - Accept the github classroom invitation
  - Go to the github repo that's created for you and copy the url
  - Using your file system, open a git bash window in your lab 2 folder
  - Type git clone and paste the url
  - You may want to rename the folder on your machine

TTT Version 2 – Starting Files

- Let’s look at the starting files together
  - indexStart.html
  - indexV1.js – is the completed version from lab 1
  - indexStartV2.js – is the starting file for the ES6 Class version. I’ve given you some comments to walk you through the process for this application.
- You can replace your js file with mine or use my comments and your code as you do the conversion.

Lab 2 - Review

- StopWatch, Concentration and TTT version 2 are all lab 2.
- Let's look at the evaluation form together briefly.
- Please notice that I'm interested in encouraging you to use git and github in a reasonable way. You MUST use git and github to get full credit for the lab.
- Please notice that I'm interested in encouraging you to deploy your JavaScript applications to a production web server. You MUST host your applications on citstudent.lanecc.edu to get full credit for the lab.
- You should be finished with the lab at this point. The lab is worth 30 points. You have one week after the due date to complete the lab with no late penalty. You may also make corrections and resubmit within that week with no point penalty. That effectively gives you an extra week to finish the lab BUT you should do your best to finish the lab by the time of the due date!

What’s Next

- Modern JavaScript Development Tools
- Local Storage
- HTML Form Validation
- The lab is 2 problems - ToDoList and Bookmarker Applications
- Reading Quiz 3
- Participation Score 3
- Don’t forget
  - Lab 2
  - Reading Quiz 2
  - Participation Score 2