iX

Embrace Opportunity

Web Development

Week 1 | Day 2

i_X

Agenda

- Intro to Git and GitHub

- JavaScript fundamentals

- JavaScript DOM

- Exercises & homework

Git and Github

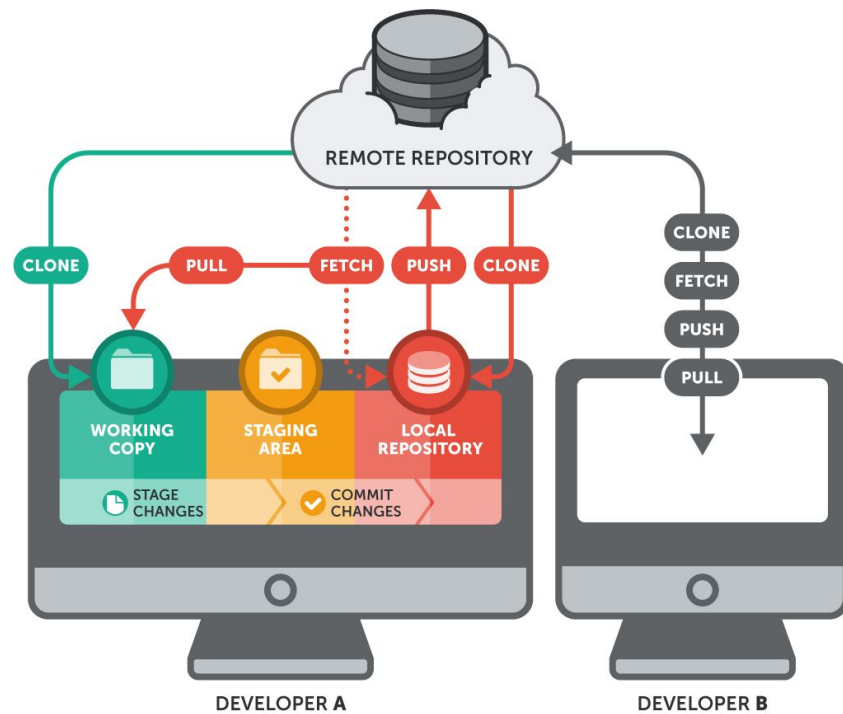An explanation, overview and simple common commands.

i_X

Git and Github

Github

Repository for your code.

Git

Helps you manage different versions.

i_X

Git

CLONE PULL FETCH PUSH CLONE CLONE FETCH PUSH PULL

REMOTE REPOSITORY

WORKING COPY

STAGING AREA

LOCAL REPOSITORY

STAGE CHANGES

COMMIT CHANGES
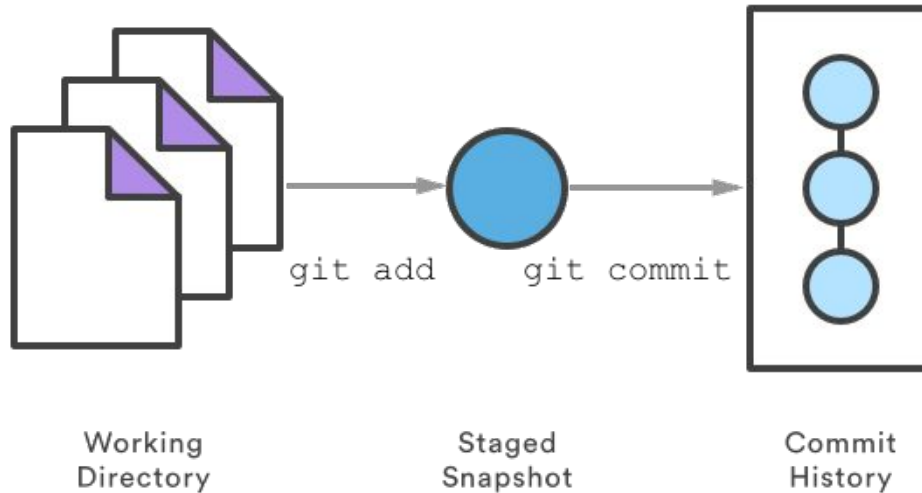
DEVELOPER A

DEVELOPER B

i_X

Git 'Trees'

- **Working Directory -** Data on the local filesystem
- **Staging Area (AKA Index) -** Proposed next commit
- **Commit History -** A history of commits in the form of a graph (directed acyclic graph)

i_X

Git Recap

The main components of a Git repository

git add   git commit

Working Directory   Staged Snapshot   Commit History

atlassian.com/git/tutorials/resetting-checking-out-and-reverting

i_X

Git Tutorial

Simply add, commit and push

i_X

Create and Initialize

We need to create a new folder, change into it and initialize it

```
mkdir wd-novel

cd wd-novel

git init
```

By running *git init* we add a hidden **.git** folder in our working directory. This folder keeps track of all of files and folders for the git command line interface.

i_X

Create a new file with content

Create a new file called `page-1.txt` and start writing your awesome novel.

```
git status
```

Reveals that there are untracked files in the working directory.

Stage the changes (pre-commit)

Get ready to commit the file by running:

```
git add page-1.txt
```

After running a *git status* again, we can see that it is now ready to be committed (or staged).

i_X

Commit the Content

We are now ready to commit.

```
git commit -m "My first page"
```

The above command commits the work and adds the content to the history.

Push to the server

Create a new repository on GitHub, then connect your local repository to the **remote repository**

```
git remote add <url>
```

You can now push your repository up to the remote repository.

```
git push origin master
```

i_X

Git Tutorial

Branching
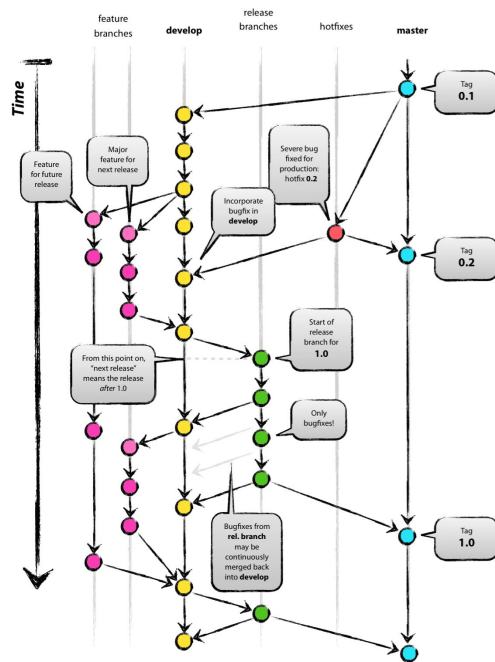
i_X

Git History Visualized

HEAD

- Reference to the commit from which the working directory was initialized

  - Intuitively, "the commit you are on"

  - Reference to the parent of the next commit

- Automatically updates itself with new commits

i_X

Git Branches

nvie.com/posts/a-successful-git-branching-model/

i_X

Branching

- Encapsulate changes via independent lines of development
- Prevent introducing unstable code into the "main" codebase
- Chance to clean up project history before integrating changes into the "main" codebase

i_X

Branching

- Branches are references to commits, not containers of commits

- Like HEAD, the active branch, if any, updates itself with new commits

- Intuitively, they are "branches" in Git history graph

i_X
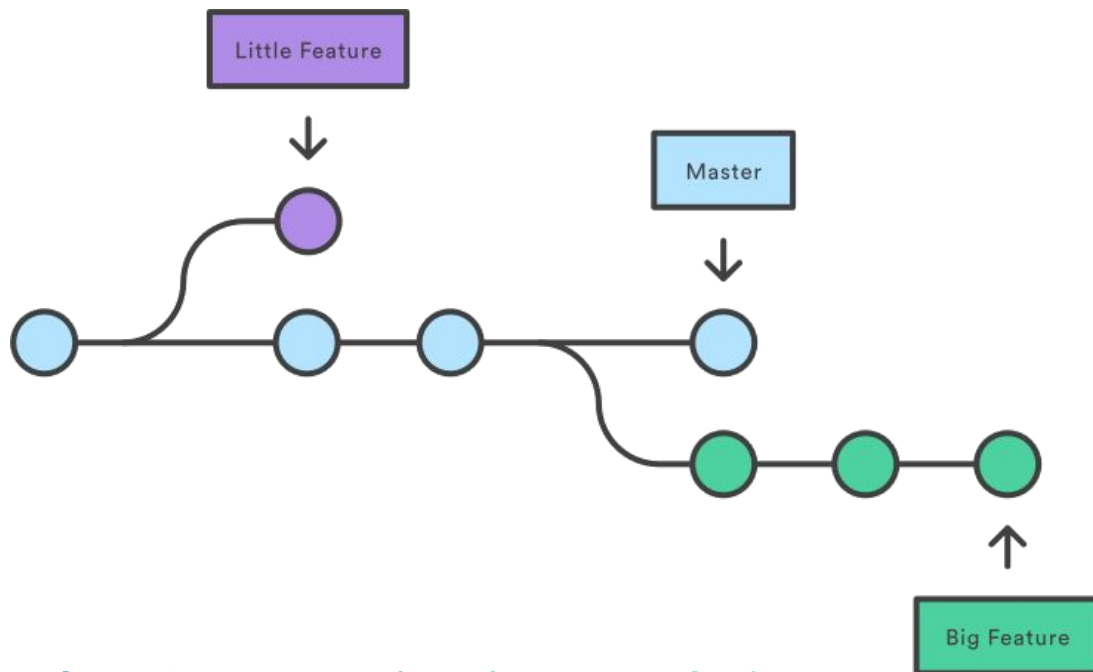
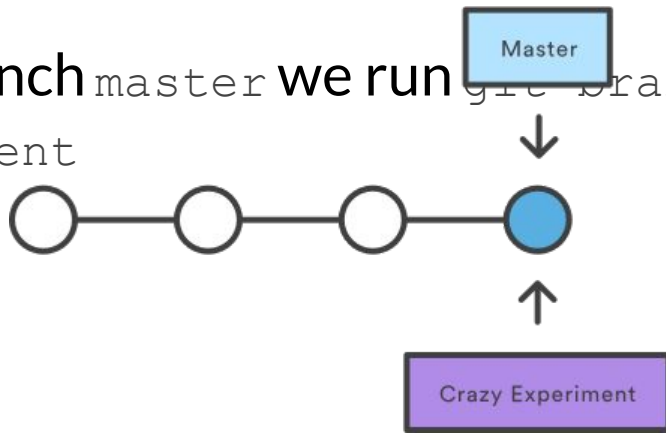# Git Branches

atlassian.com/git/tutorials/using-branches

i_X

Creating Branches

- `git branch <branch-name>`

  - Only creates `<branch-name>`

  - Does not make `<branch-name>` the active branch

- Assume on branch `master` we run `git branch`
  `crazy-experiment`

i_X

Switching Branches

- `git branch <branch-name>`

  - Only creates `<branch-name>`

- `git checkout <branch-name>`

  - Makes <active-branch> the active branch

- **Shortcut!** `git checkout -b <branch-name>`

  - Creates and checks out `<branch-name>`

i_X

Merging Branches

- `git branch <branch-name>`

    - Only creates `<branch-name>`

- `git merge <branch-name>`

    - Merges `<branch-name>` into the current branch

    - Remember, only the current branch is updated

- Sometimes Git won't be able to figure out how to merge

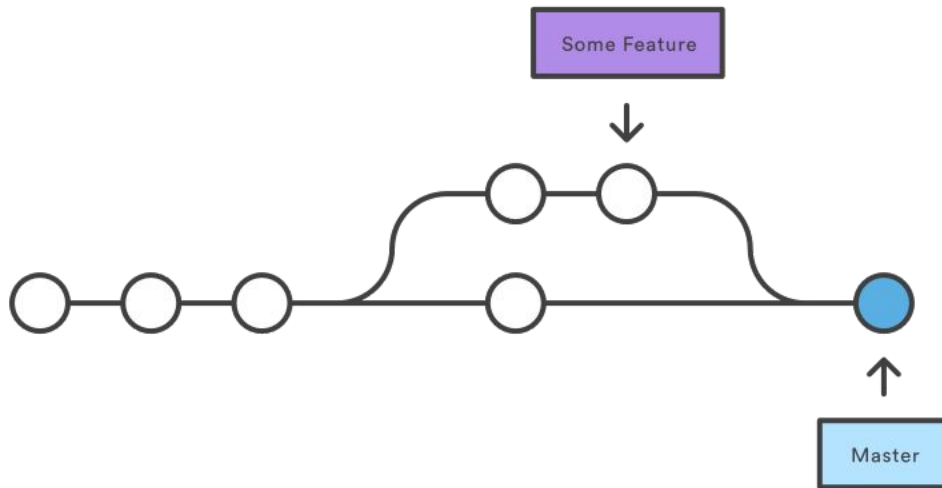   - Merge conflict - we have to manually resolve in the source code

i_X

Merging Branches

- Assume on branch `master` we run `git merge crazy-experiment`

Before Merging

After a 3-way Merge

Some Feature

Master

Some Feature

Master

i_X
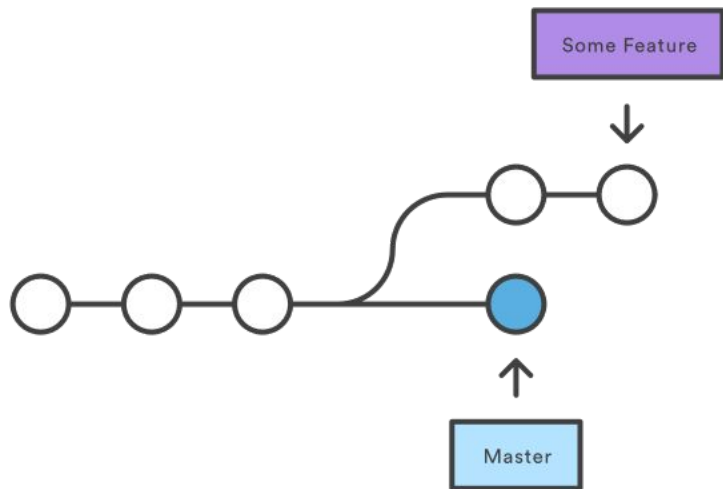
Practice

1. mkdir git-practice
2. cd git-practice
3. echo "First line\n" >> file.txt

4. git init
5. git add -A
6. git commit -m "first commit"

7. echo "Second line\n" >> file.txt
8. git add -A
9. git commit -m "second commit"

10. git branch feature
11. git checkout feature

12. echo "feature" >> file.txt

13. git add -A

14. git commit -m "commit from feature"

15. git checkout master

16. echo "master" >> file.txt

17. git add -A

18. git commit -m "commit from master"

19. git merge feature

<RESOLVE MERGE CONFLICT>

20. git add -A

21. git commit -m "resolved conflict"

i_X

JavaScript fundamentals

The need to know basics.

i_X

JavaScript

- High level object oriented programming language (abstraction)

- Interpreted (just-in-time compiling)

- ECMAScript specification (ES6)

- Runs in the browser (e.g. google V8 Engine)

- We can also run as a server thanks to run-time environments like node JS

i_X

Why JavaScript

- Used in web frameworks like Angular, React, VueJs

- Used to build very fast full stack applications

- Used to build mobile frameworks like React Native and Ionic

- Used in desktop application development

i_X

JS - Crash Course

- Variables & Data Types
- Arrays
- Objects Literals
- Methods for string, arrays objects, etc
- Loops - for, while .. of ForEach, map
- Conditions
- Functions
- OOP
- DOM

i_X

Variables

- What's a variable?
  - A value given a name so that it can be referred to throughout a program
- Syntax for defining variables in JS
  - Keywords: **var**, **let**, **const**
  - Syntax: `var name = value;`
  - Differ in scope and mutability
- Scope: region of a program where the variable binding is valid
- Mutability: whether or not the value of the variable can be changed

i_X

Variables cont.

- **var**
  - Once defined, is defined throughout the function
  - Mutable: value can be changed
- **let**
  - Once defined, is defined for the rest of the code block
  - Mutable: value can be changed
- **const**
  - Once defined, is defined for the rest of the code block
  - Immutable: value cannot be changed

i_X

Primitive Types

- Data types that are not objects and have no methods
- Important JavaScript primitive types
  - **string, number, boolean, null, undefined, symbol**
- Immutable: cannot be altered

i_X

Primitive Types cont.

- **Strings**
  - "Hello" or 'Hello'
  - Concatenation: "Hello" **+** " world"

    → "Hello world"

- **Numbers**
  - 1 21 734
  - parseInt(78.5) → 78
  - parseFloat(78.5) → 78.5

i_X

Primitive Types cont.

- **Booleans**
  - `true` and `false`
  - Truthy values such as "`hello`" and 5
  - Falsy values such as `null`, `undefined`, `NaN`, `''`, and 0
- **Undefined**
  - Value indicating that a variable has not been assigned a value or declared yet

i_X

Arrays

- Type of **data structure**
- Collection of elements that are each accessible at an **index**
- **Instantiation**: `let myArray = [3, 1, 10];`
- Add elements: `my_array.push(4, 2)`
  - → `[3, 1, 10, 4, 2]`
- Remove last element: `my_array.pop()`
- Indexing into the array: `myArray[3] → 4`
- Size of the array: `myArray.length → 5`

Objects

- **Object**: a collection of properties
- **Property**: an association between a key and a value
- ```
  let personObject = {
  ```

  ```
          name: sam,

          age: 25,

          country: Portugal

  }
  ```

Objects cont.

- `personObject`
  - `{name: Sam, age: 25, country: Portugal}`
- `personObject.name`

  - `Sam`

- `personObject.country = United States`
- `personObject.hasOwnProperty('friend')`
  - `false`
- `personObject.hasOwnProperty('age')`

  - `true`

i_X

Operators

- **Arithmetic**
  - +, -, /, *, % (modulo)
- **Relational**
  - <, >, <=, >=, ==, ===
    - == checks if two values are equal, regardless of type
    - === checks if two values are equal and have the same type
- 1 == '1' → true
- 1 === '1' → false

i_X

Conditionals

- **if-then** statements
- Different actions are performed depending on the **boolean** value of a statement
- Any statement that evaluates to a boolean can be used in a conditional

i_X

Conditionals cont.

- **if** (myVar > 0) {

    console.log('greater than zero');

  } **else if** (my_var <= -20) {

    console.log('less than/equal to -20');

  } **else** {

    console.log('in range');

  }

- **const** color = myVar >= 0 ? "red" : "blue";

i_X

Switch Statements

- Allows the value of a variable to **control the flow** of a program
- Provide a number of possible values for the variable
- For each case a course of action is taken
- Must **break** after each case for the evaluation of the switch statement to stop

i_X

Switch Statements cont.

- ```
  let my_var = 0;
  switch (my_var) {
      case 0:
          console.log(false);
          break;
      case 1:
          console.log(true);
          Break;
      default:
          console.log('error');
  }
  ```

i_X

Loops

- Used to repeat a block of code a given number of times or until a certain condition is met
- 3 types of loop
  - **for**
    - Allows a block of code to be repeated a given **number** of times
  - **while** & **do-while**
    - Allow a block of code to be repeated until a **condition** is met

i_X

Loops cont.

- **do-while** loops: code is executed, then condition is checked
- **while** loops: condition is checked, then code is executed
- Exit condition must be reachable to avoid an **infinite loop**
- ```
  var i = 0;
  do {
      console.log(i);
      i++;
  }
  while (i < 5);
  ```

i_X

Exercise

Write a loop that counts sheep

Have it execute 5 times.

Example output,

"Sheep number 1"

"Sheep number 2"

....

"Sheep number 5"

Functions

- **Block of code** written for a specific task
- Must be **called** to be executed
- Can take in data as **parameters**
- Can also return data
- Make code more readable and easier to understand and scale

i_X

Functions cont.

- **function** concat(str1, str2) {
  return str1 + str2;
  }

  console.log(concat("hello ", "world"));

- → "hello world"

OOP - Object Oriented Programming

- **Prototypes (ES5)**
- **function** Person(firstName,lastName, dob) {

    this.firstName;

    this.lastName;

    this.dob;

  }

  const person1 = new Person("John", "Doe", "4-3-1993")

  console.log(person1);

- Using Date object: this.dob = new Date(dob);

i_X

OOP - Cont (Prototypes - ES5)

- Add methods to objects
- **function** Person(firstName,lastName, dob) {

  this.firstName;

  this.lastName;

  this.dob;

  This.getFullName = function(){

      return this.firstName+" "+this.lastName;

  }

  }

iX

OOP - Cont (Prototypes - ES5)

- **function** Person(firstName,lastName, dob) {

    this.firstName;

    this.lastName;

    this.dob;

    this.getFullName

  }

  Person.**prototype**.getFullName() = **function**(){

    return this.firstName+" "+this.lastName;

  }

i_X

OOP - Cont (Classes - ES6)

- **class** Person{

  constructor(firstName,lastName, dob) {

     this.firstName;

     this.lastName;

     This.dob;

  }

  getFullName(){

  return this.firstName+" "+this.lastName;

  }

  }

i_X

JavaScript documentation

https://developer.mozilla.org/en-US/docs/Web/JavaScript

i_X

JS Exercise

- Create and array of object of the following structure:
  - Use let keyword
  - Name array "people"
  - Object Structure
    - { name: "name", age: 25, email: "name@mail.com" }
- Use the map array function to create a new array called "names" populated with all the name properties from the objects in the people array

i_X

JS Exercise

- Use the forEach array function to loop through all names and print out the name to the console
- Create a function called "getNames" and add the logic you wrote using the map function
- Now when creating your variable called names use you "getNames" function instead

i_X

JavaScript DOM

A comprehensive outline of the document object model.

i_X

The JavaScript DOM

- The DOM is a structured representation of an HTML document
- Tree of nodes or elements created by the browser
- Node or element: any html tags
- Use JS to manipulate these dom elements or nodes
- The DOM is object oriented, each node has its own set of properties and methods
- We can change and add or remove them

- The browser gives us a window object and inside that we have a document object (loaded web-page or document)
- Root element, html element
- head tag, body tag same level
- head: meta-tags title
- body: output, h1 tags, links, headers, footers etc

Let's check the Document object

console.log(document);

i_X

The JavaScript DOM

DOM tree

i_X
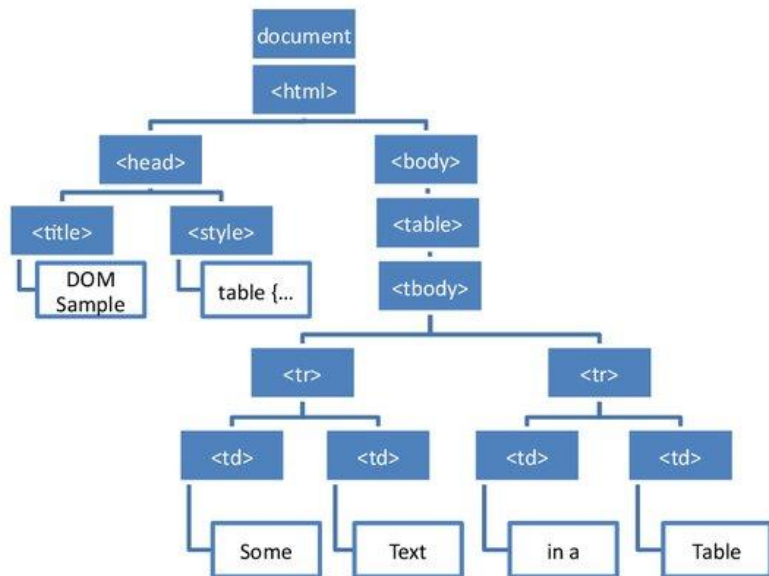
The DOM

- Examining the DOM

- Single element selectors

- Multiple element selectors (HTML collection or Node list)

- Traversing the DOM

- Creating element and adding attributes

- Editing/Removing elements and attributes

- Event listeners and the event object

- Mouse, Input and form events

- Event bubbling and delegation

- Local Storage

i_X

Document Object Model

- What is the DOM?
  - https://www.w3schools.com/whatis/whatis_htmldom.asp
- HTML tree of objects. This Defines:
  - Properties, Methods, Events
- Enables binding between HTML, JS and CSS (API for JS)

i_X

Exercise and homework

Exercises: DOM projects
Homework: Movie Game

i_X

Looking ahead and homework

Exercise

- Simple Interest calculator
- Number Guesser

Homework

- Movie guessing game using what we have learned so far

i_X

Thanks for listening!

Tomorrow we will take a look at object oriented programming.

i_X
See you tomorrow