Text As Code

Alex Schumacher

Outline

1. Character Encoding
2. Markup Languages
3. Publishing
4. Working With `git`

Why does this matter?

- Foundational knowledge for many tech-related jobs.
- Useful in many areas, not just writing code.
- Lets you edit Wikipedia to say whatever you want.

Character Encoding

- How each letter, symbol, or other piece of text is stored on a computer.
- Translates binary into human.

Definitions

- *Character*: Individual symbols or letters - `A` `!` `+` `Ö`

Definitions

- *Character*: Individual symbols or letters - `A` `!` `+` `Ö`
- *Binary*: Code that represents all characters as sets of `0` or `1` (ex. `A` is `01000001`)

Definitions

- *Character*: Individual symbols or letters - `A` `!` `+` `Ö`
- *Binary*: Code that represents all characters as sets of `0` or `1` (ex. `A` is `01000001`)
- *Encoding*: Translating characters into binary according to a set of rules.

How it works:

- Someone had to decide what set of binary numbers represent what character, creating a `character set`.
- Every computer program references this `character set` to determine what characters to show when you read a file.

Examples

- ASCII (American Standard Code for Information Interchange)
- UTF-8 (Unicode Transformation Format – 8)

Examples

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000	SP			Space
33	041	21	00100001	!	!	!	Exclamation mark
34	042	22	00100010	"	"	"	Double quotes (or speech marks)
35	043	23	00100011	#	#	#	Number sign
36	044	24	00100100	$	$	$	Dollar
37	045	25	00100101	%	%	%	Per cent sign
38	046	26	00100110	&	&	&	Ampersand
39	047	27	00100111	'	'	'	Single quote
40	050	28	00101000	(	(	&lparen;	Open parenthesis (or open bracket)
41	051	29	00101001	)	)	&rparen;	Close parenthesis (or close bracket)
42	052	2A	00101010	*	*	*	Asterisk
43	053	2B	00101011	+	+	+	Plus
44	054	2C	00101100	,	,	,	Comma
45	055	2D	00101101	-	-		Hyphen-minus
46	056	2E	00101110	.	.	.	Period, dot or full stop
47	057	2F	00101111	/	/	/	Slash or divide
48	060	30	00110000	0	0		Zero
49	061	31	00110001	1	1		One
50	062	32	00110010	2	2		Two
51	063	33	00110011	3	3		Three
52	064	34	00110100	4	4		Four
53	065	35	00110101	5	5		Five
54	066	36	00110110	6	6		Six
55	067	37	00110111	7	7		Seven
56	070	38	00111000	8	8		Eight
57	071	39	00111001	9	9		Nine
58	072	3A	00111010	:	:	:	Colon
59	073	3B	00111011	;	;	;	Semicolon

Examples

Apple	
Google	
Microsoft	

Markup Languages

- Combines the character encoding and all the other formatting of a document (font, spacing, margins, etc).
- Interpreted by a program to display or print.

Definitions

- *Plain Text*: Text without any formatting directives - the raw characters saved in a file.
- *Rich Text*: The text you see after the markup language has been rendered (ex. MS Word document)

Example

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua
```

Lorem

~~ipsum dolor sit amet~~, *consectetur adipiscing elit*,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua

Example

Markup directives add formatting:

- `abcd` from `` `abcd` `` as surrounded backticks
- *italic* from `*italic*`
- **bold** from `**bold**`
- ~~strikethrough~~ from `~~strikethrough~~`
- “ blockquote from `>` ”
- source link from `[source link](http://www.website.com)`
- and some fancy emojis 😄 from `:smile:` using words surrounded by colons

Markup Language Examples

- HTML
- XML
- Markdown
- Wikicode
- reStructured Text

Text Editors

- Plain text editors:
  - VSCode
  - Notepad (Windows)
  - LaTeX
- Rich Text editors (aka, "what you see is what you get")
  - MS Word
  - Google Docs

Example

Microsoft Word Documents uncompressed

Publishing

- Software to render the markup language into the end-product - how text is made readable.
- Web, print, PDF, and everything else goes through this process.

Web Publishing

- Files are stored on a server.
- Someone visits the webpage, initiating a connection to the server.
- The server provides the text and markup directives as code.
- The browser (Chrome, Firefox, etc) interprets the code according to the markup directives and renders it so you can read it.

Real World

- Text stored in a markup language gives the browser a place to start.
- Style sheets and frameworks provide all the rest.
- Images, video, and other files are also told how to display with directives in the markup language.

Working with text as code

- Likely you won't need to master a markup language, but you should be familiar with the concepts when working in tech.
- A common collaboration tool is `git` which allows multiple people to work on the same project together.

Git

- `git` allows people to work on different parts of the same project at the same time, then combine them into the final product.
- Work on plain text files in an editor, then save them to a central location so everyone on the team can see them.

Why use git?

- Organizes a large number of shared text files.
- Everyone can see and use the same character encoding, styles, and framework.
- Allows for input and revision before publishing a final version.

GitHub vs `git`

- `git` is a free software
- GitHub is a version of `git` that Microsoft owns and runs. Easy and mostly free to use.

GitHub Process

- Create a `repository`
- Create a `branch`
- Add/edit files and `commit` them
- Submit the changes for a `pull request`
- `Merge` the changes into the `main` branch

GitHub Publishing

- `git` and GitHub are often used as the source for publishing.
- Can be setup to automatically send changes to the `main` branch to a web publishing system.

Homework

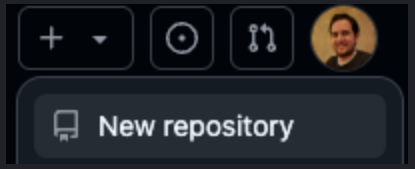
1. Create a free GitHub account
2. Create a public repo with a `README.md` file.
3. Create a new branch
4. Update the `README.md` file with anything you want.
    1. At least three sections with headers
    2. At least two lists
    3. At least five text enhancements (bold, italic, strikethrough, underline, subscript, superscript, etc)
    4. Bonus points:
        1. Pictures
        2. Tables
        3. Links
        4. Table of Contents
5. Merge the branch into the `main` branch

Thanks!

Full presentation available at:
https://github.com/mrhollywoodgates/text_as_code

Homework Walkthrough

1. Create a repository



2. Repository Details:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner *

mrhollywoodgates

Repository name *

my_repo

my_repo is available.

Great repository names are short and memorable. Need inspiration? How about verbose-meme ?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

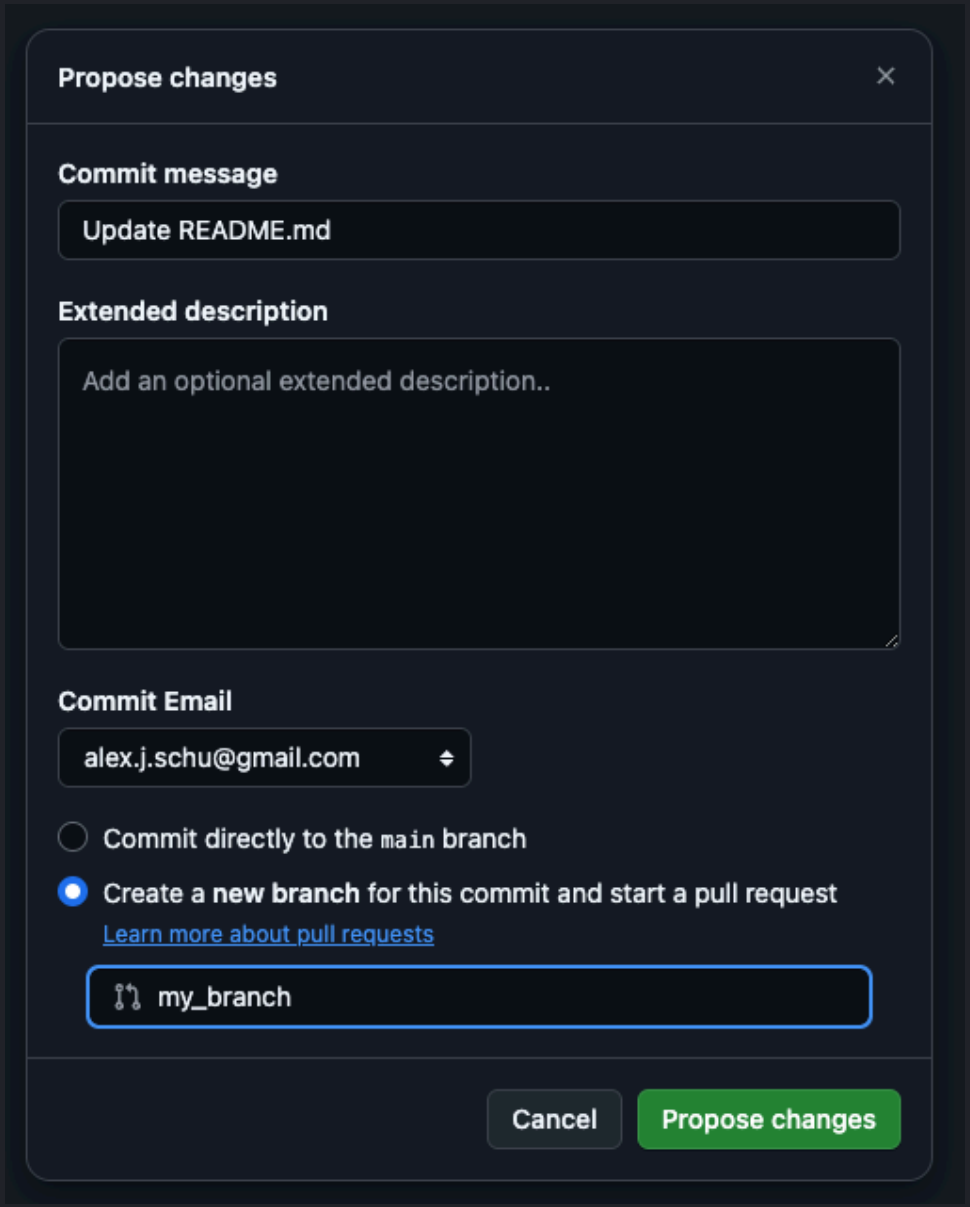
This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

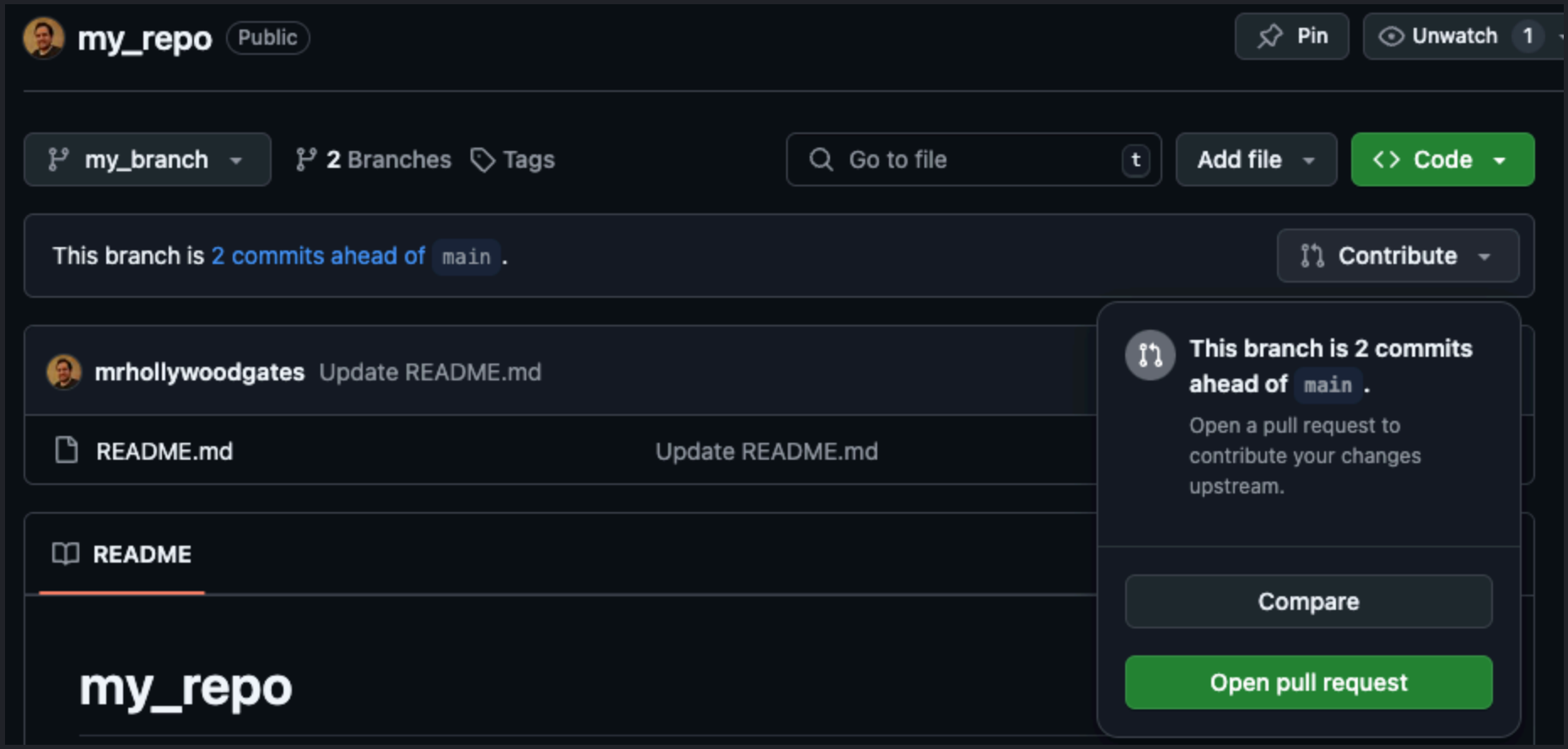
Create repository

3. Edit the file then click `Commit changes...`

4. Commit and create branch



5. Create pull request



6. Pull request details

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about d

base: main ← compare: my_branch ✓ Able to merge. These branches can be automatically merged.

Add a title

My branch

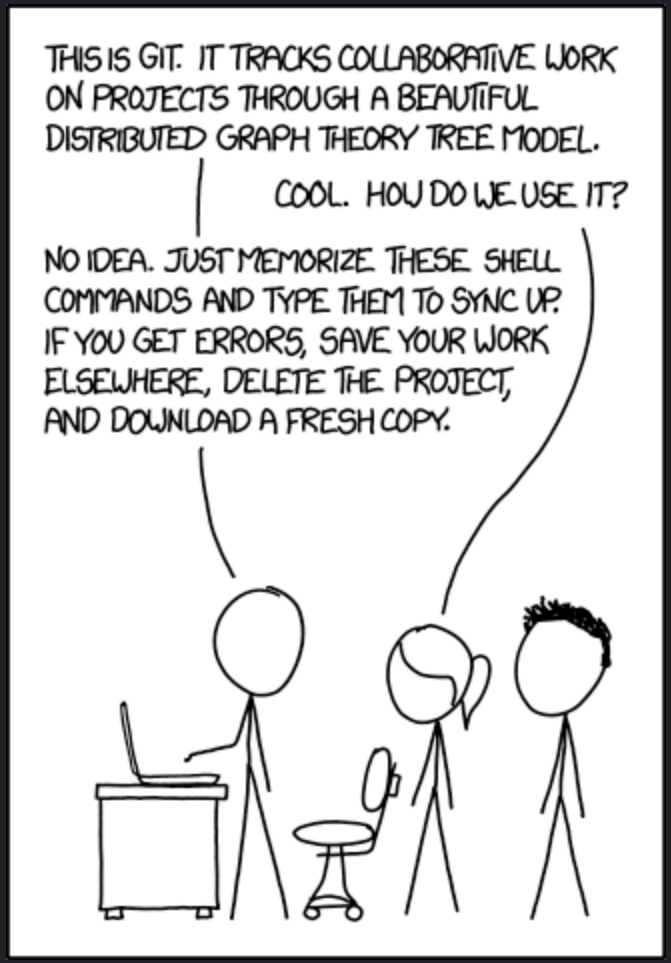
Add a description

Write | Preview

this is why we should make the changes
and this is a summary of the changes

Markdown is supported | Paste, drop, or click to add files

Create pull request


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.