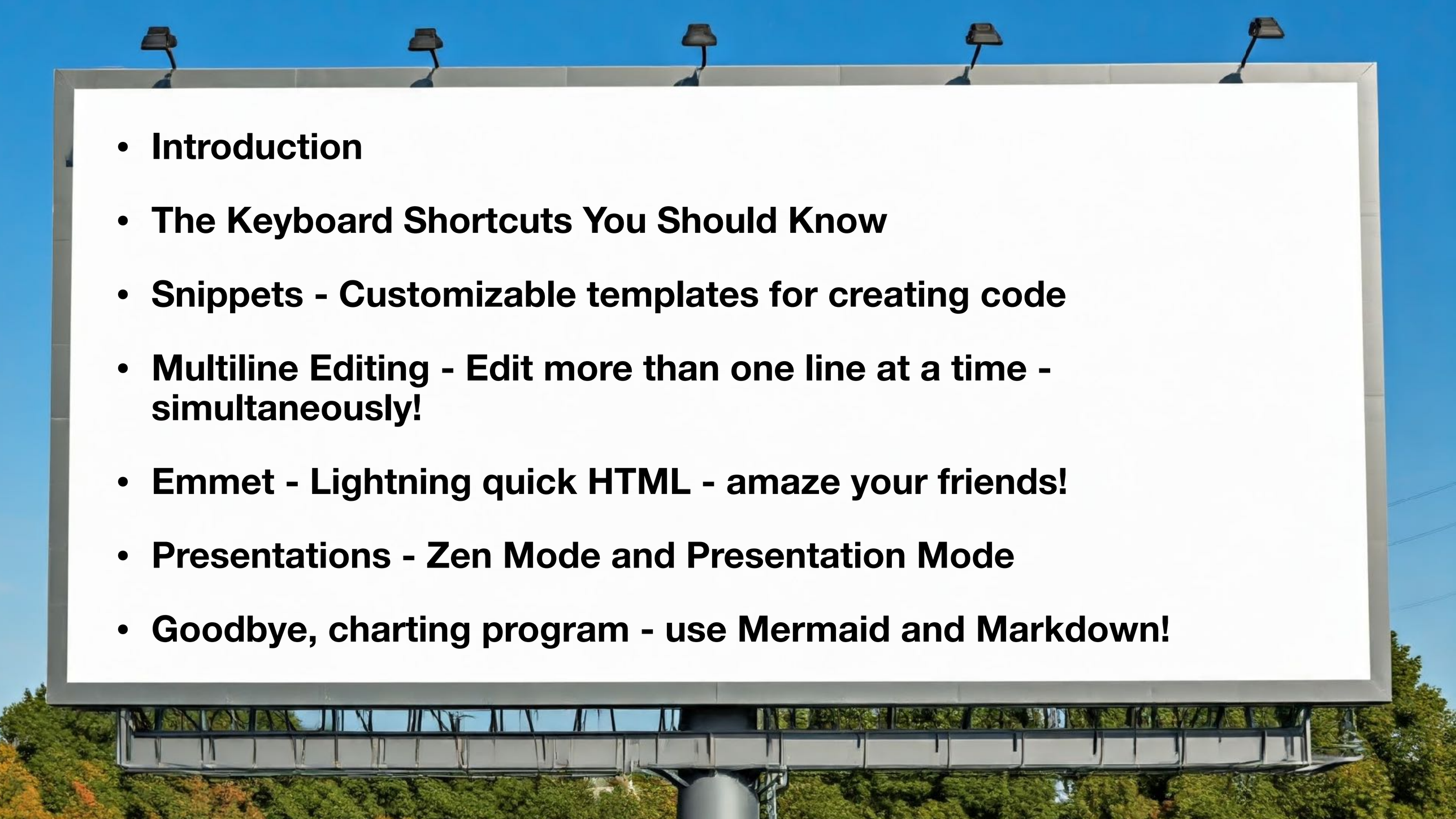
# Step On It!

## *Using Visual Studio Code Features To Speed Up Your Development*

Kelly Morrison


- Introduction
- The Keyboard Shortcuts You Should Know
- Snippets - Customizable templates for creating code
- Multiline Editing - Edit more than one line at a time - simultaneously!
- Emmet - Lightning quick HTML - amaze your friends!
- Presentations - Zen Mode and Presentation Mode
- Goodbye, charting program - use Mermaid and Markdown!

# Kelly Morrison, Ph.D. Computer Engineering

## Solution Architect - Daugherty Business Solutions


BUT WHY?
imgflip.com


I AM WEB DEV.
KNOW CTRL-C
AND CTRL-V.
$200K PLS.
WILL CODE C#
FOR PIZZA


BASIC TRAINING
VISUAL STUDIO CODE SHORTCUTS 101

# What is it?

Keyboard shortcuts to invoke IDE actions.

# How will it make me faster?

Less effort than using right-click menus, selecting from drop-down menus, etc.

Quicker - no need to remove hands from keyboard.

# Basic Commands

**Show the Command Palette: CMD + SHIFT + P (Windows: CTRL + SHIFT + P)**

- Brings up the Visual Studio Code Command Palette
- Almost everything can be accessed from here:
  - Settings, Debug, you name it
- **Memory aid**: “P” for “Palette”

# Basic Commands

**Search in workspace files: CMD + SHIFT + F (Windows: CTRL + SHIFT + F)**

- Brings up the Visual Studio Code Command Palette
- Lets you search for some text in all of your workspace files
- **Memory aid**: “F” for “Find”

# Basic Commands

## Find a file by name: CMD + P (Windows: CTRL + P)

- Searches your project for a file with a matching name
- Useful when projects get large with lots of files
- **Memory aid**: “P” for “Project file” or “find files matching this Pattern”

# Basic Commands

## Undo: CMD + Z (Windows: CTRL + Z)

- Lets you undo the last action
- A lifesaver, especially when you accidentally delete selected text!
- **Memory aid**: “Z” is the last letter of the alphabet - I think of this command as a “last resort” when I’ve made a serious mistake

# Basic Commands

## Redo: CMD + SHIFT + Z (Windows: CTRL + SHIFT + Z)

- Lets you redo the last action
- Useful when toggling back and forth between a change
- **Memory aid**: Same as the undo action (CTRL + Z), but the shift “reverses” it

# Basic Commands

## Zoom In / Out: CMD -, CMD + (Windows: CTRL -, CTRL +)

- Lets you make things bigger or smaller
- Useful when showing your IDE in a meeting or presentation and people are having trouble viewing it
- **Memory aid**: “+” and “-“ for bigger and smaller

# Layout Management Commands

**Toggle the sidebar visibility: CMD + B (Windows: CTRL + B)**

- Shows or hides the sidebar
- Useful when you need the room (e.g., when giving a presentation)
- **Memory aid**: “B” for “bye bar” (or, if you remember NSYNC, “bye bye bar”)

# Layout Management Commands

## Toggle the terminal pane: CTRL + ` (same for Windows)

- Shows or hides the terminal
- Useful when you need a command line
- **Memory aid**: “Backtick” for “Back in time to a command line, when we used the Control key”

# Layout Management Commands

## Close the current window: CMD + W (Windows: CTRL + W)

- Closes the window that you're currently in
- Useful when you're done with a file and want to reduce clutter in the IDE
- **Memory aid**: "W" for "close the Window"

# New Window Commands

## Splitting the current window: CMD + \ (Windows: CTRL + \)

- Makes a separate live editor window of your current file
- Useful when you want to compare two different parts of the same file simultaneously
- **Memory aid**: “\” looks like a divider

# New Window Commands

**Creating a Markdown preview window: CMD + SHIFT + V (Windows: CTRL + SHIFT _+  V)**

- Brings up a new window that shows what your Markdown code looks like when rendered
- Useful when creating Markdown documents (and Mermaid diagrams)
- **Memory aid**: “V” for “View my document”


Snippets

# What is it?

Templates for inserting code.

Lets you type a few characters and have a (customizable) block of code or text inserted.

# How will it make me faster?

Faster than typing code from scratch.

Easier to insert project-specific code.

# Types Of Snippets

- Can install **Global** or **Project-Specific** snippets
- **Global** snippets are available in every project
  - Are kept with your Visual Studio Code installation
- **Project-Specific** snippets are only available in a single project
  - Are kept in the **.vscode** directory in a project

# Configuring Snippets

## Creating a Global Snippets file

- Bring up the Command Palette: **CMD** + **SHIFT** + **P**
- Search for “**Snippets: Configure Snippets**”
- Choose “**New Global Snippets File…**”
- When it asks for a file name, type in the **name of a language**
  - E.g., python, java, markdown, snippets

# Configuring Snippets

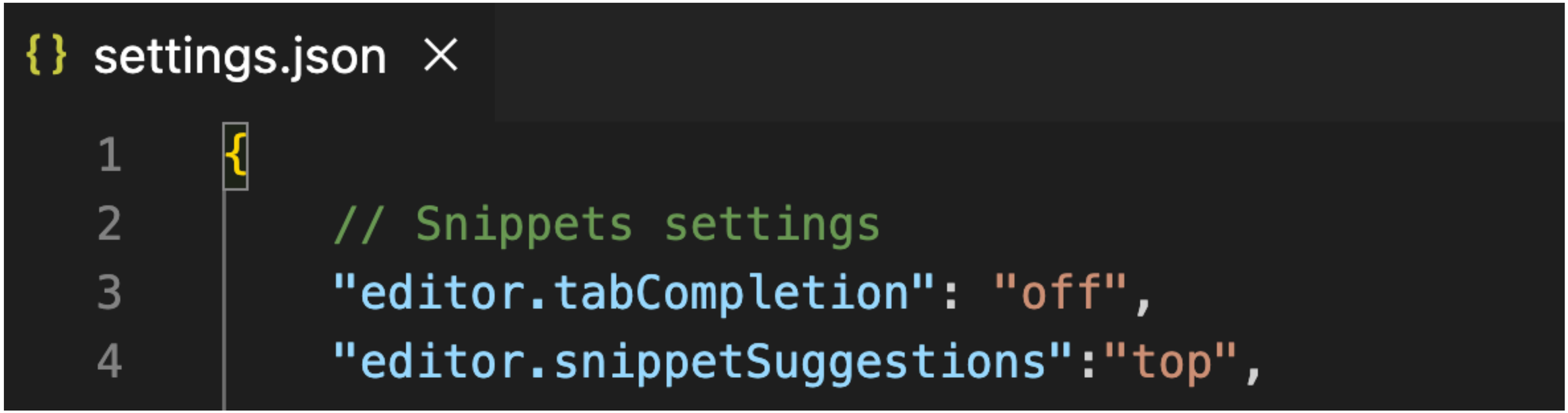
## Creating a Project-Specific Snippets file

- Bring up the Command Palette: **CMD** + **SHIFT** + **P**
- Search for “**Snippets: Configure Snippets**”
- Choose “**New Snippets File for ‘Project Name’…**”
- When it asks for a file name, type in the **name of a language**
  - E.g., python, java, markdown, snippets

# Configuring Snippets

## Project Settings

- Bring up the Command Palette: **CMD** + **SHIFT** + **P**
- Search for “**Preferences: Open User Settings (JSON)**”



# Anatomy of a Snippet

- Prefixes should be short: this is what you type to trigger the snippet insertion
- I use something unusual, like starting the prefix with “zz”

```
"My Cool Snippet": {
    "prefix": "zzmcs",
    "body": ["System.out.println(\"Hello World!\");"],
    "description": "Example Snippet for Java"
},
```

# Naming Snippets

## Useful prefix tips

- Use **zz** (or some other unusual, unique prefix)
- Use company abbreviations:
  - The Home Depot: thdheader, **thdfooter**, **thdnavbar**
- Use codes
  - **ut**___ for unit test snippets, **dm**___ for demo snippets, etc.

# Configuring Snippets

## Guiding the user

- Use **$1**, **$2**, etc. to let the user enter text
  - If you use it more than once in your snippet, the same text is inserted in each place
- Use **$0** to indicate where the cursor should be when finished
- Can use defaults
  - ${1:default_value}, ${2:something}

# Configuring Snippets

## Letting the user choose from one of several values

- ${1|World,Universe,Connect.Tech|}
  - Pipe symbols delimit the beginning and end of available choices
  - Choices are separated by commas
- Brings up a menu that lets the user choose one of the values

# Built-in Snippet Variables

For inserting the current date and time:

- `CURRENT_YEAR` The current year
- `CURRENT_YEAR_SHORT` The current year's last two digits
- `CURRENT_MONTH` The month as two digits (example '02')
- `CURRENT_MONTH_NAME` The full name of the month (example 'July')
- `CURRENT_MONTH_NAME_SHORT` The short name of the month (example 'Jul')
- `CURRENT_DATE` The day of the month as two digits (example '08')
- `CURRENT_DAY_NAME` The name of day (example 'Monday')
- `CURRENT_DAY_NAME_SHORT` The short name of the day (example 'Mon')
- `CURRENT_HOUR` The current hour in 24-hour clock format
- `CURRENT_MINUTE` The current minute as two digits
- `CURRENT_SECOND` The current second as two digits
- `CURRENT_SECONDS_UNIX` The number of seconds since the Unix epoch
- `CURRENT_TIMEZONE_OFFSET` The current UTC time zone offset as `+HH:MM` or `-HH:MM` (example `-07:00`).

For inserting random values:

- `RANDOM` 6 random Base-10 digits
- `RANDOM_HEX` 6 random Base-16 digits
- `UUID` A Version 4 UUID

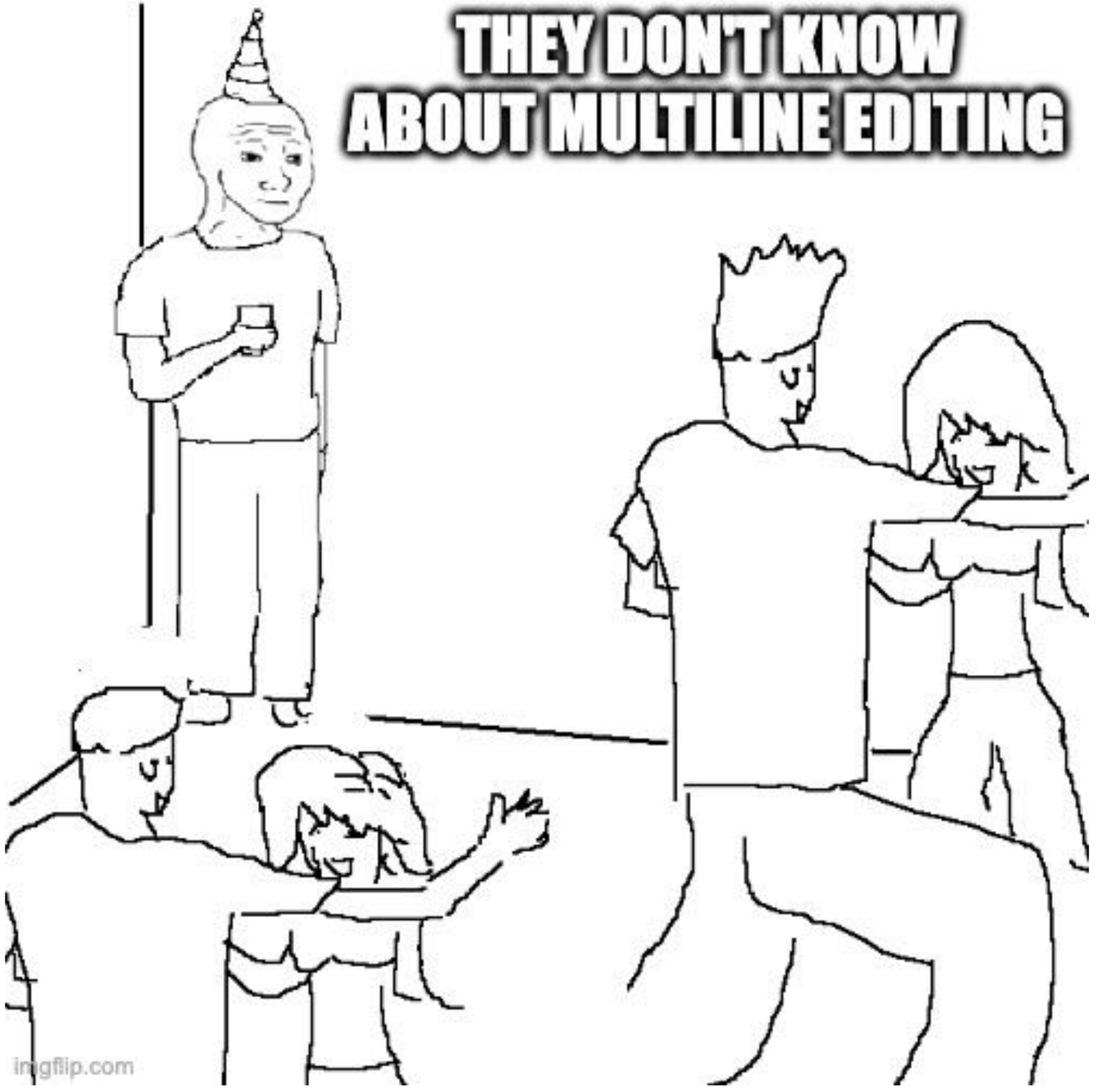
WHY ISN'T THERE A SNIPPET

FOR CREATING SNIPPETS?

# I made one

```
"New snippet": {
    "prefix": "zzsnippet",
    "body": [
        "\"$1\": {",
        "\t\"prefix\": \"$2\",",
        "\t\"body\": [\"$3\"],",
        "\t\"description\": \"$0\"",
        "},"
    ],
    "description": "Creates a new snippet body",
}
```

```
1  Multiline Editing is cool and can sav you a lot of time
2  Multiline Editing is cool and can sav you a lot of time
3  Multiline Editing is cool and can sav you a lot of time
4  Multiline Editing is cool and can sav you a lot of time
5  Multiline Editing is cool and can sav you a lot of time
6  Multiline Editing is cool and can sav you a lot of time
7  Multiline Editing is cool and can sav you a lot of time
8  Multiline Editing is cool and can sav you a lot of time
9  Multiline Editing is cool and can sav you a lot of time
10 Multiline Editing is cool and can sav you a lot of time
11 Multiline Editing is cool and can sav you a lot of time
12 Multiline Editing is cool and can sav you a lot of time
13 Multiline Editing is cool and can sav you a lot of time
14 Multiline Editing is cool and can sav you a lot of time
15 Multiline Editing is cool and can sav you a lot of time
16 Multiline Editing is cool and can sav you a lot of time
17 Multiline Editing is cool and can sav you a lot of time
18 Multiline Editing is cool and can sav you a lot of time
```


THEY DON'T KNOW
ABOUT MULTILINE EDITING
imgflip.com

# What is it?

Lets you edit more than one line of code simultaneously.

# How will it make me faster?

Faster and easier than cut & paste.

Simpler than search & replace.

# Multiline Editing

## Selecting text by position

- Click somewhere on a line.
- Hold down **COMMAND** + **OPTION** and then move the cursor keys up and down to extend the selection.
- When you've selected every line you want to change, start editing.
- Press **ESC** when you're done.

# Multiline Editing

## Selecting text by the text itself

- Select some text.
- Hold down **COMMAND** + **SHIFT + L.**
- Start editing.
- Press **ESC** when you’re done.

# Multiline Editing

## Selecting the ends of lines

- Select some text.
- Hold down **OPTION** + **SHIFT + I.**
- Start editing.
- Press **ESC** when you’re done.

# Emmet

# What is it?

Lets you write HTML and CSS quickly.

It was originally called “Zen Coding”.

# How will it make me faster?

Can generate complex HTML and CSS with just a few keystrokes. And no more worrying about matching opening/closing brackets!

# Basic Emmet

## HTML5

- The **html:5** abbreviation will generate an HTML5 page template

# Basic Emmet

## Emmet 101

- You can type element names (e.g., **ul**, **li**, **etc**.)
  - Longer names have shortcuts (**bq** for **blockquote**)
- The **>** symbol is used to denote a child element
  - **div>div>span** would create a span within a div within a div
- **.name** is used to add a **class** called *name*
- **#id** is used to add an **id** called *id*

# Basic Emmet

## Emmet 101 (continued)

- The ***n** is used to create n copies of an element
  - **div*5** would generate 5 divs
- The **$** symbol is used with ***n** to create a number
  - **div.item$*5** would generate the following:

```
<div class="item1"></div>
<div class="item2"></div>
<div class="item3"></div>
<div class="item4"></div>
<div class="item5"></div>
```

# Basic Emmet

## Emmet 101 (Adding symbols at the same level)

- The **+** symbol adds another symbol at the same level
  - **div>p+span** would yield:

```
<div>
    <p></p>
    <span></span>
</div>
```

# Basic Emmet

## Emmet 101 (Adding symbols up a level)

- The ^ symbol means to go back up a level
  - **div>div>span>{Hi}^p** would go back up to the span level and then add a **p**
  -

```
<div>
    <div>
        <span>Text</span>
        <p></p>
    </div>
</div>
```

  - **div>div>span>{Hi}^^p** would create this:

```
<div>
    <div><span>Text</span></div>
    <p></p>
</div>
```

# Basic Emmet

## Lorem

- You can add “Greek” text with the **lorem** operator
- You can also specify how many words to generate
  - **lorem:4** will generate **4** words
  - **lorem:15** will generate **15** words

# Adding your own Emmet snippets

## Overview

- Two steps:
  - Create a file called **snippets.json** (NOT the same as for the Snippets feature)
    - Can go inside any top-level folder in your project (e.g., “**emmet/**“)
  - Configure Emmet in your project
    - Need to set **“emmet.extensionsPath”** to point to your top-level folder in your **settings.json** file

# Adding your own Emmet snippets

## A sample snippet file

Link to Emmet code that shows where it looks for snippets

```
{
    "html": {
      "snippets": {
        "zzmain": "header+main+footer>div>{(c) 2024 Connect.Tech Demos, Not Really Inc.",
        "zzmenu": "art>div[id=${1}]>ul>li.item$*5>lorem5"
      }
    }
}
```

# Adding your own Emmet snippets

## Configuring Emmet settings

- **COMMAND + SHIFT + P** to bring up the Command Palette
- Select **“Preferences: Open User Settings (JSON)”**

# Emmet “cheat sheet”

https://docs.emmet.io/cheat-sheet/

# Presentation And Zen Mode

# What is it?

Lets you remove clutter for peaceful coding and better presentations.

# How will it make me faster?

A more peaceful coding environment with less distractions may improve productivity.

Saves time when showing code in presentations, PR reviews, etc.

# Presentation Mode

**Resets font sizes, colors, etc. for better presentations**

- Press **CTRL + SHIFT + P**
- Do it again to toggle back to normal view

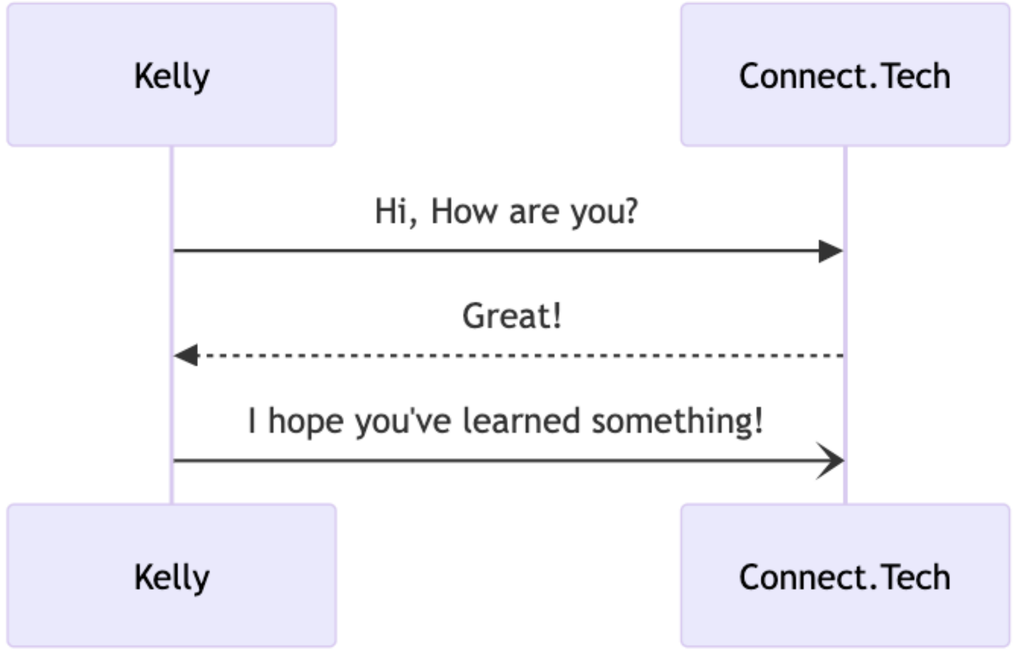
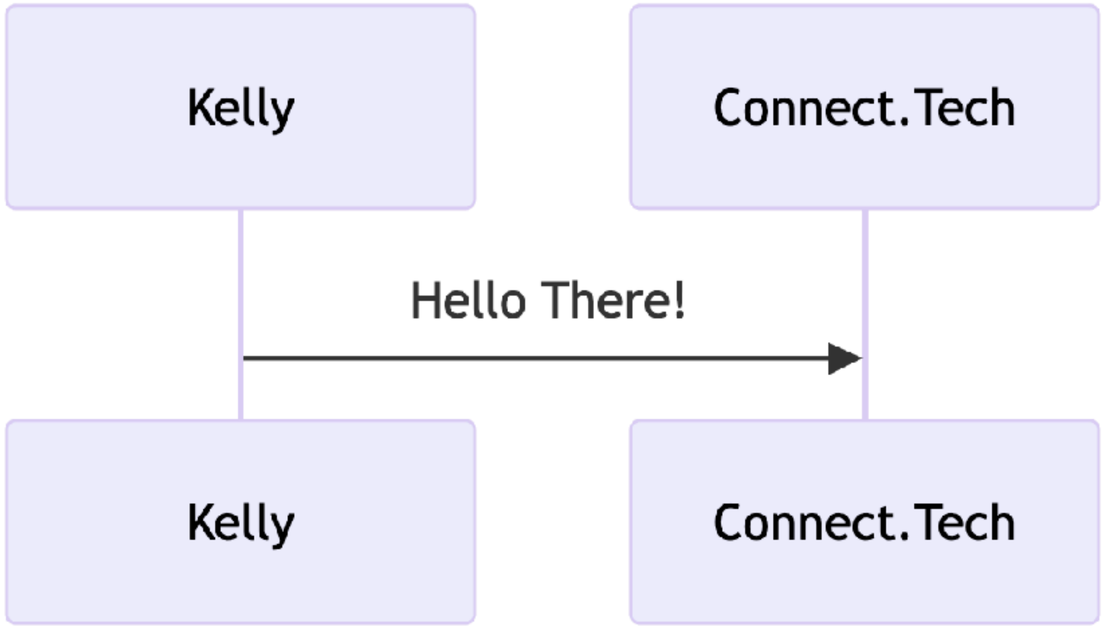
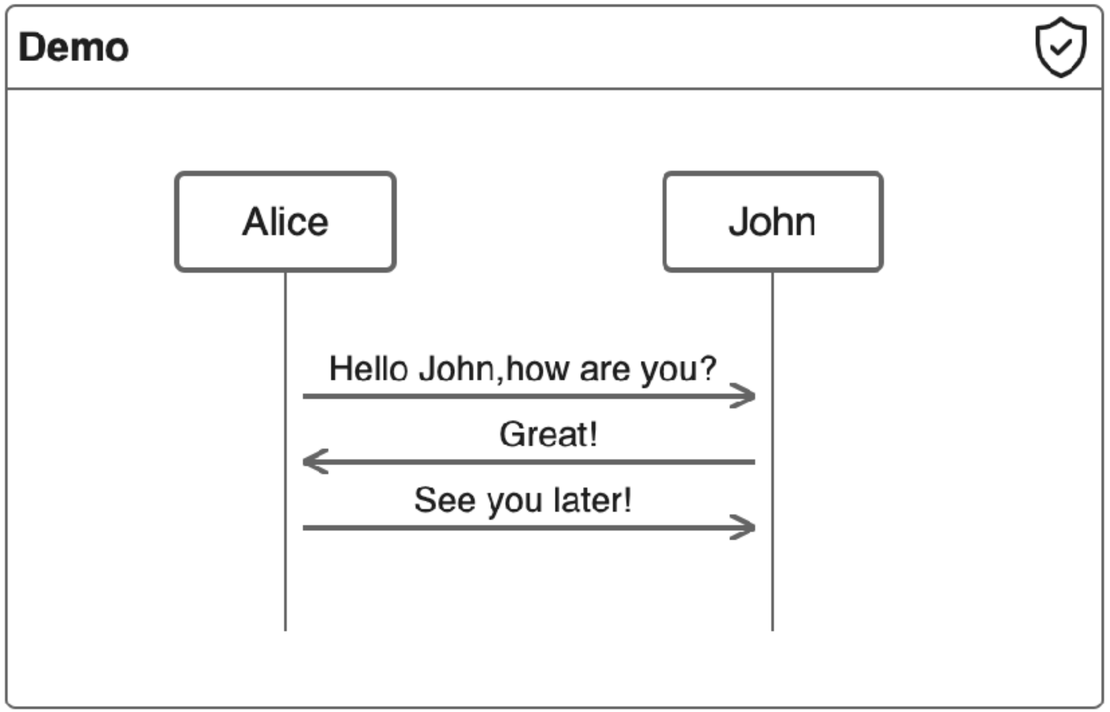
# Zen Mode

## Shows only the file you’re working in

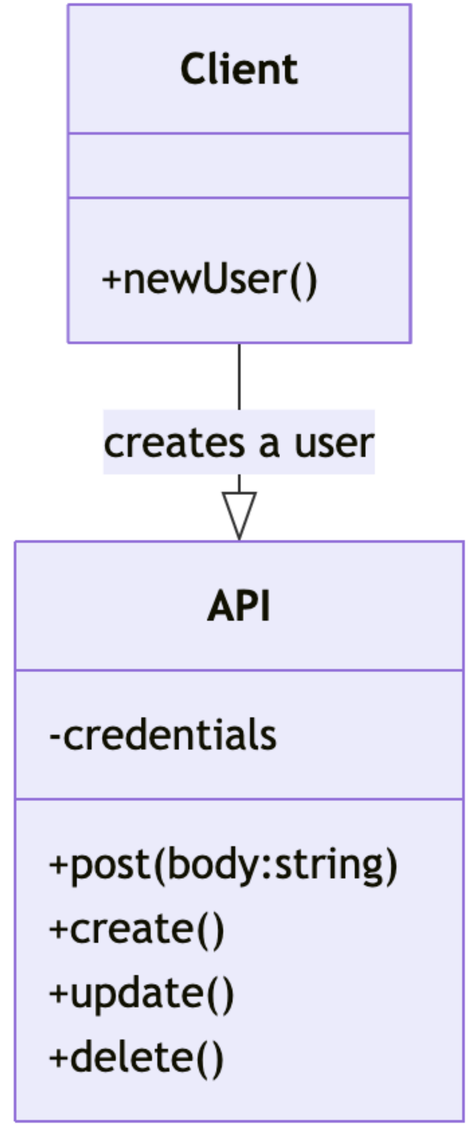
- Click in the window you want to see
- Press **CTRL + K**, let go, then press **Z**
- Do it again to toggle back to normal view
    - Make sure to move the cursor OUTSIDE your window

```
# Mermaid diagrams

## A simple diagram

```Mermaid
sequenceDiagram
    Kelly->>Connect.Tech: Hi, How are you?
    Connect.Tech-->>Kelly: Great!
    Kelly-)Connect.Tech: I hope you've learned something!
```
```

# Mermaid diagrams

## A simple diagram



# What is it?

Lets you generate diagrams in a text editor.

Can render on GitHub!

# How will it make me faster?

Diagrams can reside in a code repository.

Diagrams are text, so can be versioned, compared to previous versions, etc.

Diagrams render in the browser on GitHub.

# Mermaid

## Simple text-based diagrams in a Markdown document

- https://mermaid.js.org/intro/
- Create a Markdown document
- Insert a “mermaid” block

````
```mermaid
sequenceDiagram
    participant Kelly
    participant Connect.Tech
    Kelly->>Connect.Tech: Hello There!
```
````



# Mermaid

## ZenUML

````
## Zen UML example
```mermaid
zenuml
    title Demo
    Alice->John: Hello John, how are you?
    John->Alice: Great!
    Alice->John: See you later!
```
````

## Zen UML example



# Mermaid

## Class diagram

````
```mermaid
classDiagram
    Client --|> API: creates a user
    class Client {
        +newUser()
    }
    class API {
        -credentials
        +post(body:string)
        +create()
        +update()
        +delete()
    }
```
````

## Class diagram example



# Mermaid

## Other diagram types

- There are many other types of Mermaid diagram
  - State diagrams, ERD, Gantt, Pie Chart, User Journey, Timelines, etc.

Daugherty

BUSINESS SOLUTIONS

Link to my GitHub repo of presentations.

**Kelly Morrison**
Manager / Application Architect at Daugherty Business Solutions

kellyivymorrison@gmail.com

LinkedIn