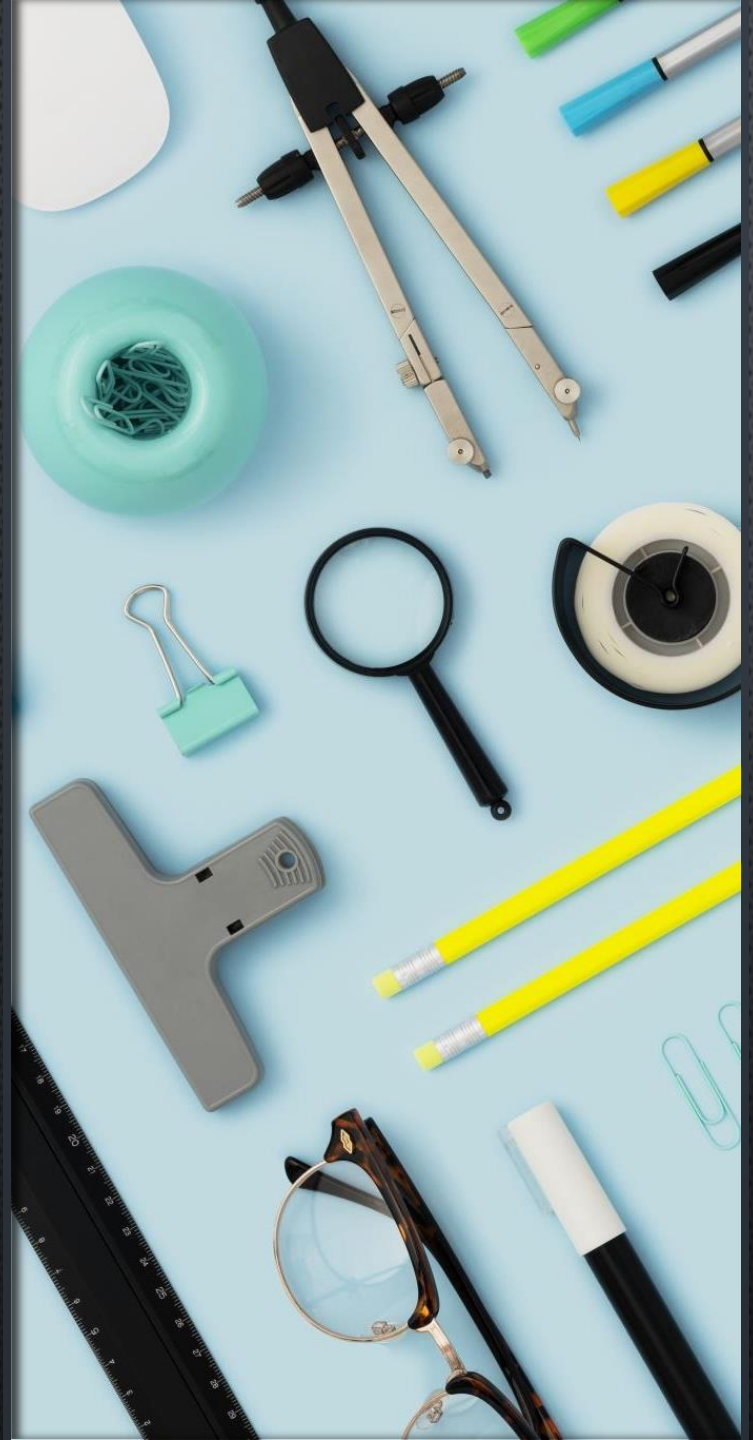
WELCOME TO CSCI E-33A WEB PROGRAMMING W/ PYTHON AND JAVASCRIPT SECTION 4

TF FOR THIS SECTION: GLENN LANGDON

LANGDON@CS50.HARVARD.EDU



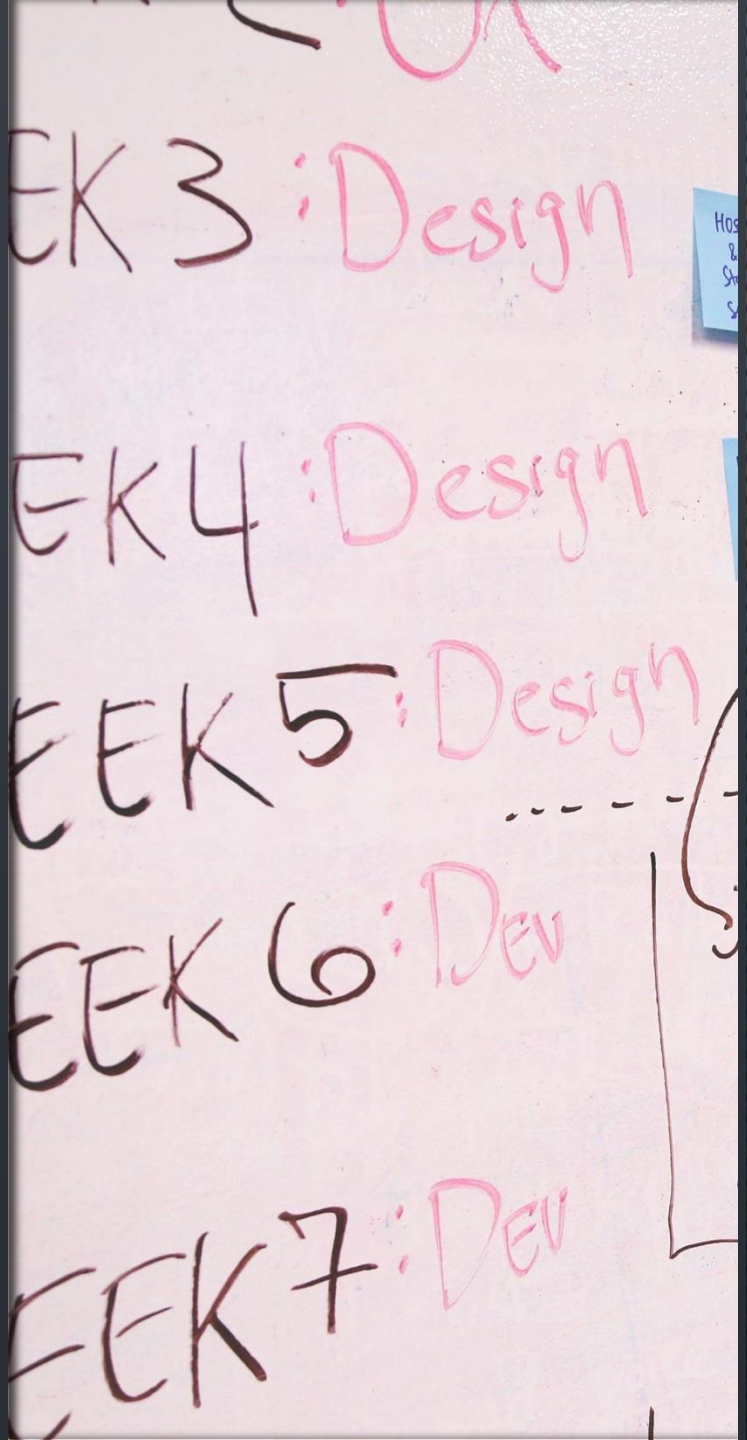
Program for today

- What's in store for the rest of the semester
- JavaScript
- Quick review of some aspects Django
- Preview of Project 3: Mail

QUESTIONS?

What we've done and what's left

- Good News! We've covered just about everything you need to know in Python and Django for the remainder of the semester
- There are a few abstractions that we will add to our existing knowledge base, but nothing huge.
- Current assignment is all JS and HTML (more on that later)
- After that, we synthesize everything we've done in the remaining project
- Then…



IT'S TIME FOR A PROJECT YOU DEFINE

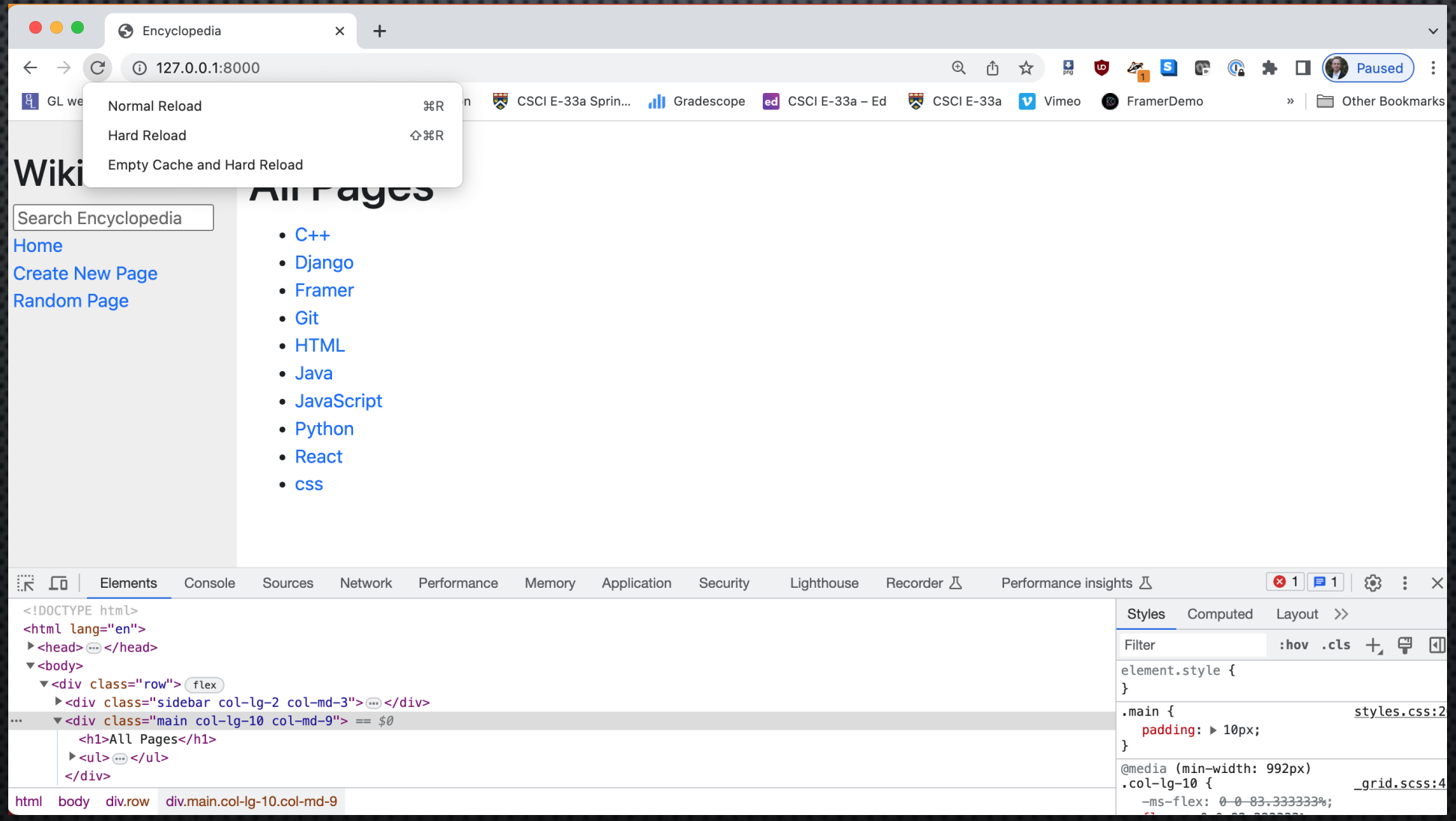
A FEW DJANGO CONCEPTS MIGHT BEAR A QUICK REVIEW

- WHY ISN'T MY CSS CHANGING?
- I KEEP GETTING A MODEL ERROR AFTER CHANGING MY DATABASE
- A LITTLE MORE ON M2M RELATIONSHIPS

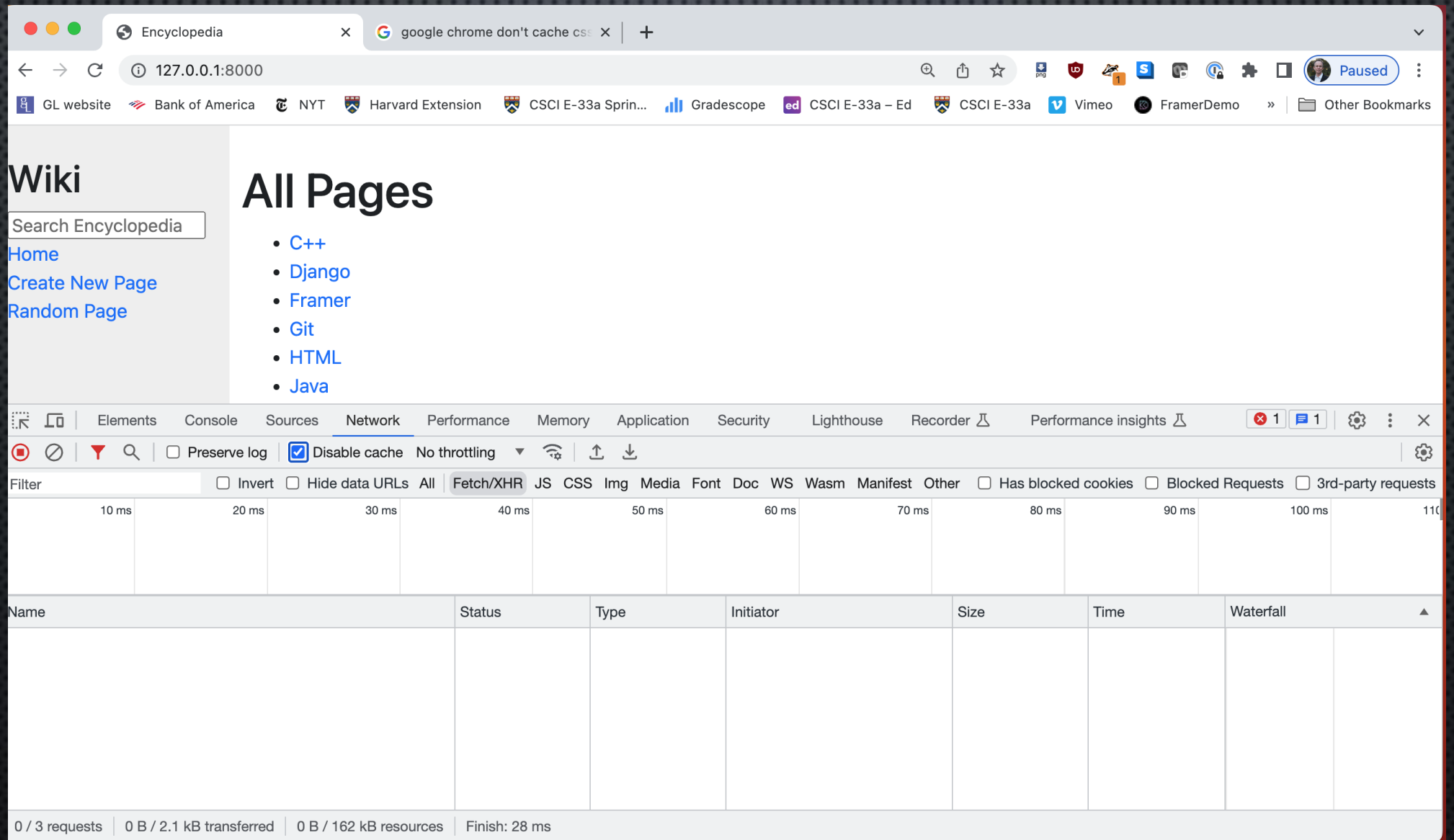
WHY ISN'T MY CSS CHANGING?

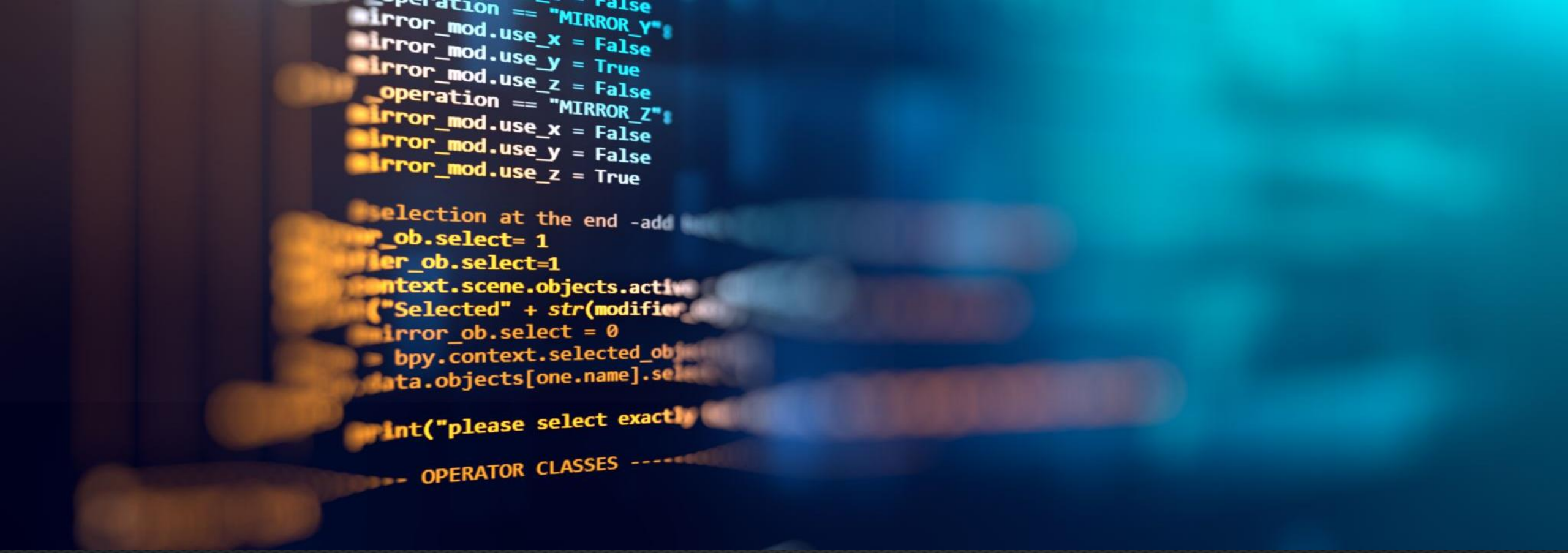
- YOUR BROWSER CACHES YOUR STATIC FILES
- THAT'S A GOOD THING FOR PERFORMANCE
- IT'S A PAIN FOR DEVELOPERS CONSTANTLY CHANGING CSS AND JS FILES
- HERE'S WHAT YOU CAN DO ABOUT IT…

MANUALLY CLEAR YOUR BROWSER'S CACHE



DISABLE THE CACHE




irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
operation == "MIRROR_Z"
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True
selection at the end -add
_ob.select= 1
er_ob.select=1
("Selected" + str(modifier
irror_ob.select = 0
bpy.context.selected_ob
ata.objects[one.name].se
int("please select exactl
OPERATOR CLASSES

JAVASCRIPT

JAVASCRIPT

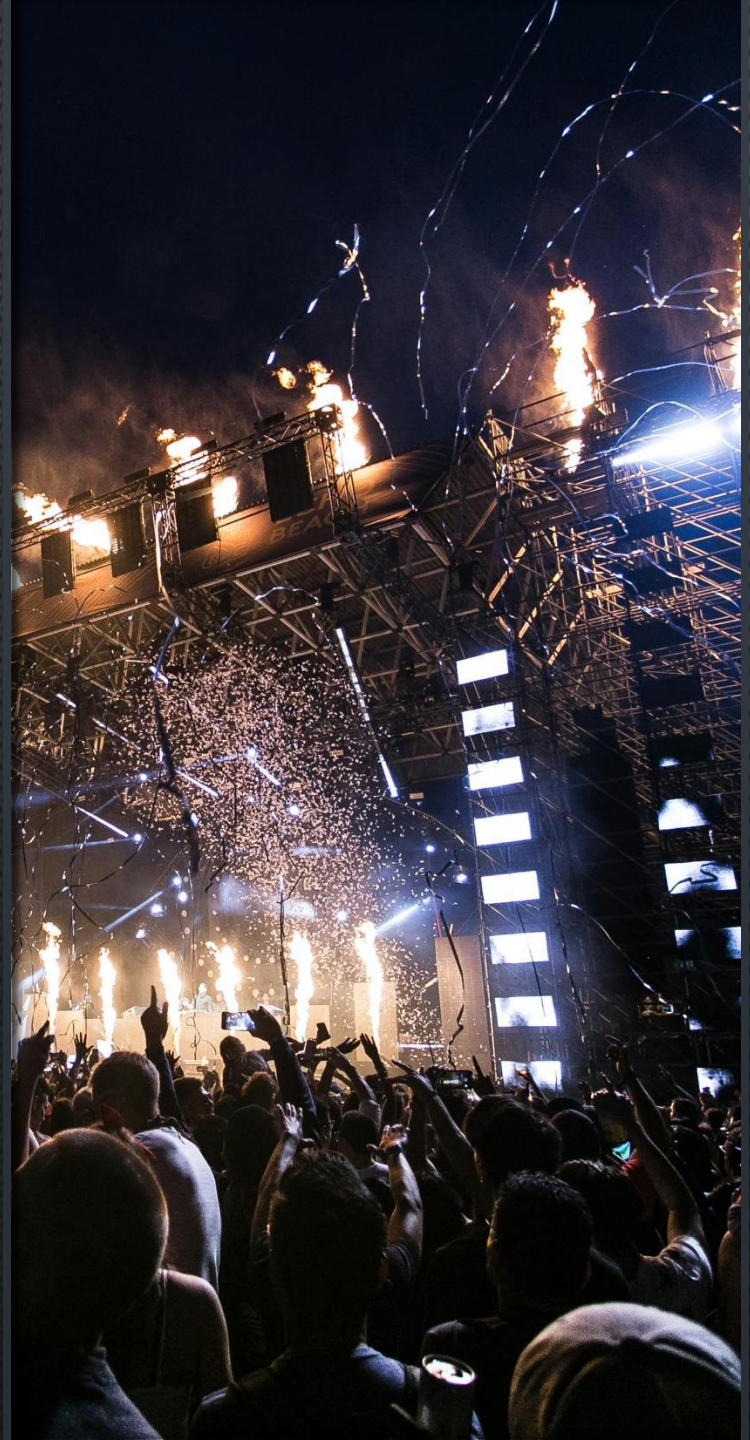
- **JAVASCRIPT (JS)** IS A LIGHTWEIGHT INTERPRETED (OR JUST-IN-TIME COMPILED) PROGRAMMING LANGUAGE WITH FIRST-CLASS-FUNCTIONS.

What does that mean?

- Lightweight:
  - Minimalist syntax and features
  - Efficient resource usage
  - Dynamic (variables do not need explicit type definitions)
- Interpreted
  - Execution without compilation
  - Browser execution

SELECTING DOM OBJECTS

HERE'S AN EXAMPLE…



JAVASCRIPT EVENTS

- EVENTS ARE ACTIONS OR OCCURRENCES THAT HAPPEN IN THE BROWSER, WHICH CAN BE DETECTED AND HANDLED BY JAVASCRIPT, SUCH AS A USER CLICKING ON A BUTTON OR TYPING IN A FORM FIELD.
- EVENT HANDLING INVOLVES WRITING CODE THAT RESPONDS TO THESE EVENTS.

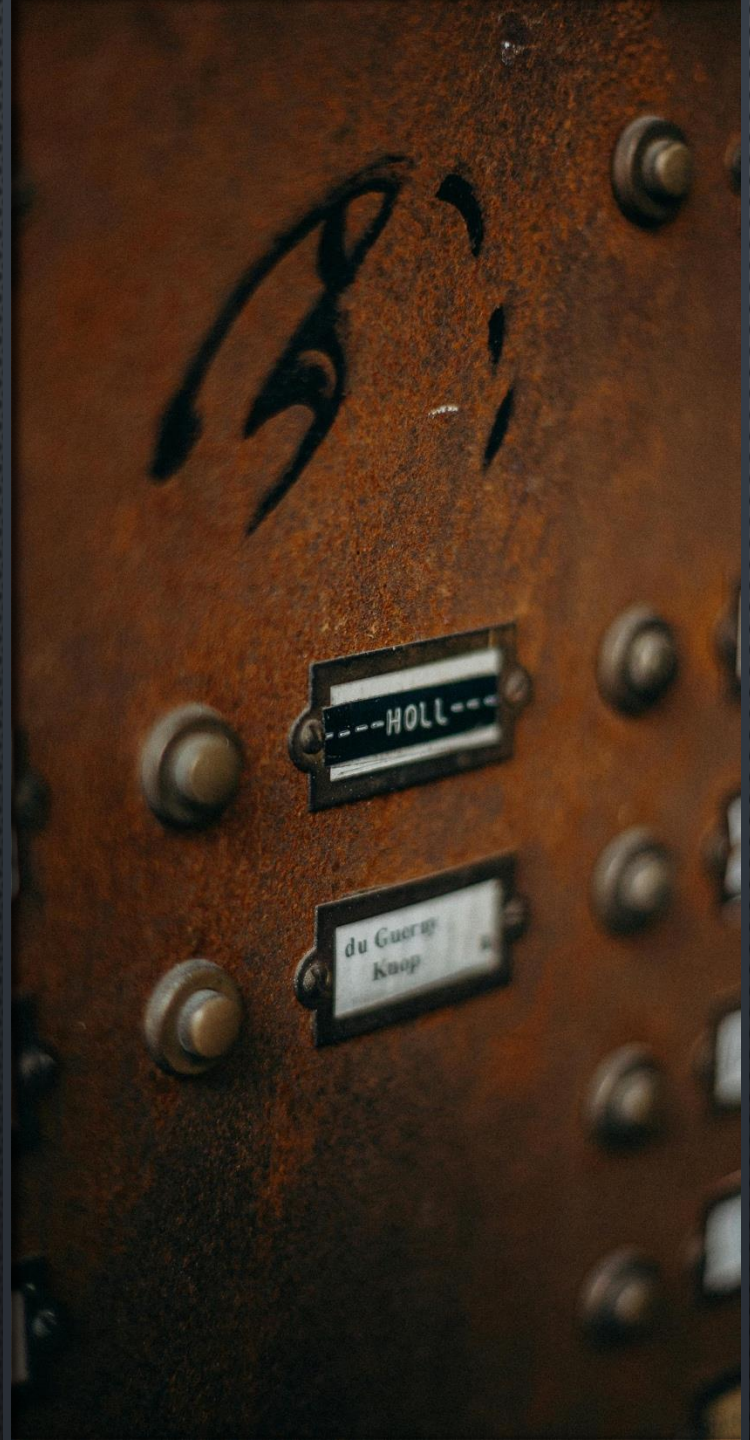
JAVASCRIPT EVENT OBJECT

- It contains information about the event that occurred and provides methods and properties that you can use to interact with the event.
- The event object is automatically passed to event handler functions when an event occurs. Anytime you interact with your browser, your browser creates an event
- This event object contains a bunch of properties that describe it (type, target, current target, among others)
- To do something with the event we need to create an event handler and register it with an element in the DOM

```
210
211    document.querySelector('#some_id').addEventListener('click', some_method);
```

Common JavaScript Event Types

- Mouse
  - click, mouseover, mouseout
- Keyboard
  - keydown, keyup
- Form
  - submit, blur, focus
- Window
  - load, DomContentLoaded



EVENT LISTENER VS. EVENT HANDLER

- EVENT LISTENER: LIKE A DOORBELL. WAITING FOR SOMEONE TO PRESS IT.
- EVENT HANDLER: WHAT HAPPENS WHEN THE DOORBELL IS PRESSED.

Registering an event handler on an element

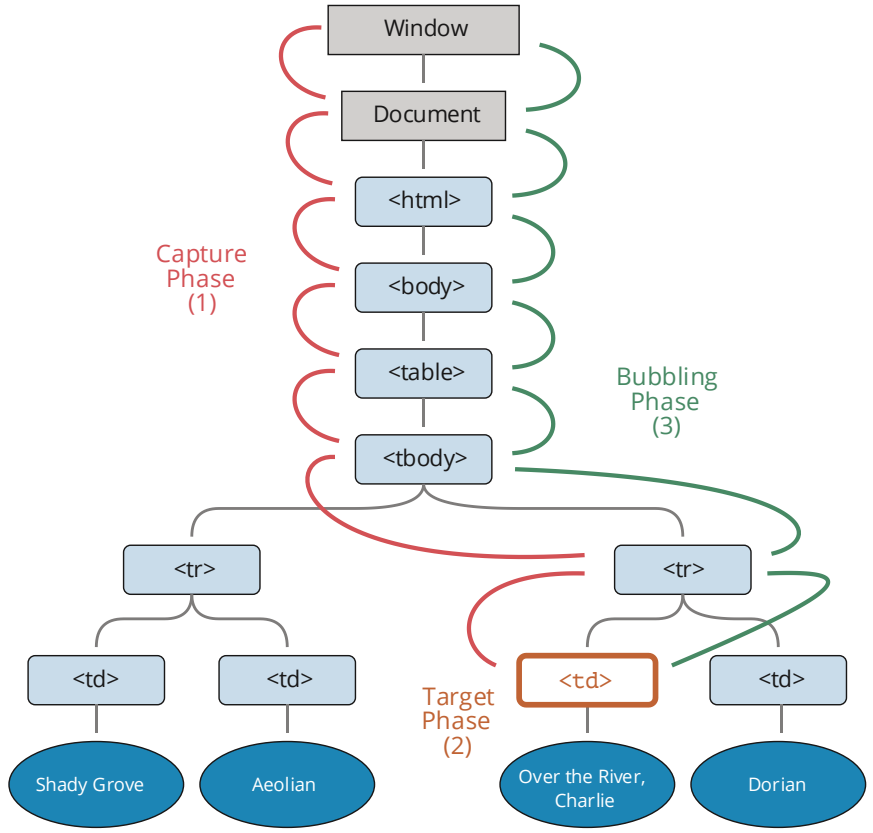
- Works by adding a function to the list of event listeners for a specified event on the EventTarget on which it's called.
- addEventListener(type, event handler)

```
document.querySelector("#then-btn").addEventListener('click', () => {
    getWord();
});
```

HERE ARE SOME EXAMPLES OF EVENTS AND EVENT HANDLING

FINDING THE TARGET

- When an event is fired, the event object has to find ALL elements that could potentially have listeners registered to them for that event
- The event object starts at the outer most starting point, the Window object, and works its way down toward the target element
- When it reaches the target it executes, if it has an event handler registered to it.
- Then the event object makes a return trip to the window object



Event target, event current target

- event.target = Returns the element where the event occurred, the element that triggered the event.
- Event.currentTarget = Returns the element to which the handler is attached, the element to which the event listener is bound.
- Here's a few examples…

CREATING ELEMENTS IN JAVASCRIPT

- There are times when you'll need to create DOM elements from underlying dynamic data.

- Here's an example.

```
207
208  document.querySelector("#someID").addEventListener('click', function(event) {
209    event.preventDefault();
210  })
```

PREVENTING DEFAULT BEHAVIOR

```
206
207  function do_something(event: any, arg: any): void
208  function do_something(event, arg) {
209    some_defined_func(arg);
210    // prevent page refresh
211    event.preventDefault();
212  }
```

Sometimes we want to prevent default behavior for an event

For example, preventing a page refresh on a form submit

PROMISES, PROMISES

- We don't go into promises in this course but a basic understanding of how promises work can help you understand some errors that may arise in your code.

Promise

The `Promise` object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

ASYNCHRONOUS PROGRAMMING

A function starts a long operation, something that takes more time than reading consecutive lines of JS code.

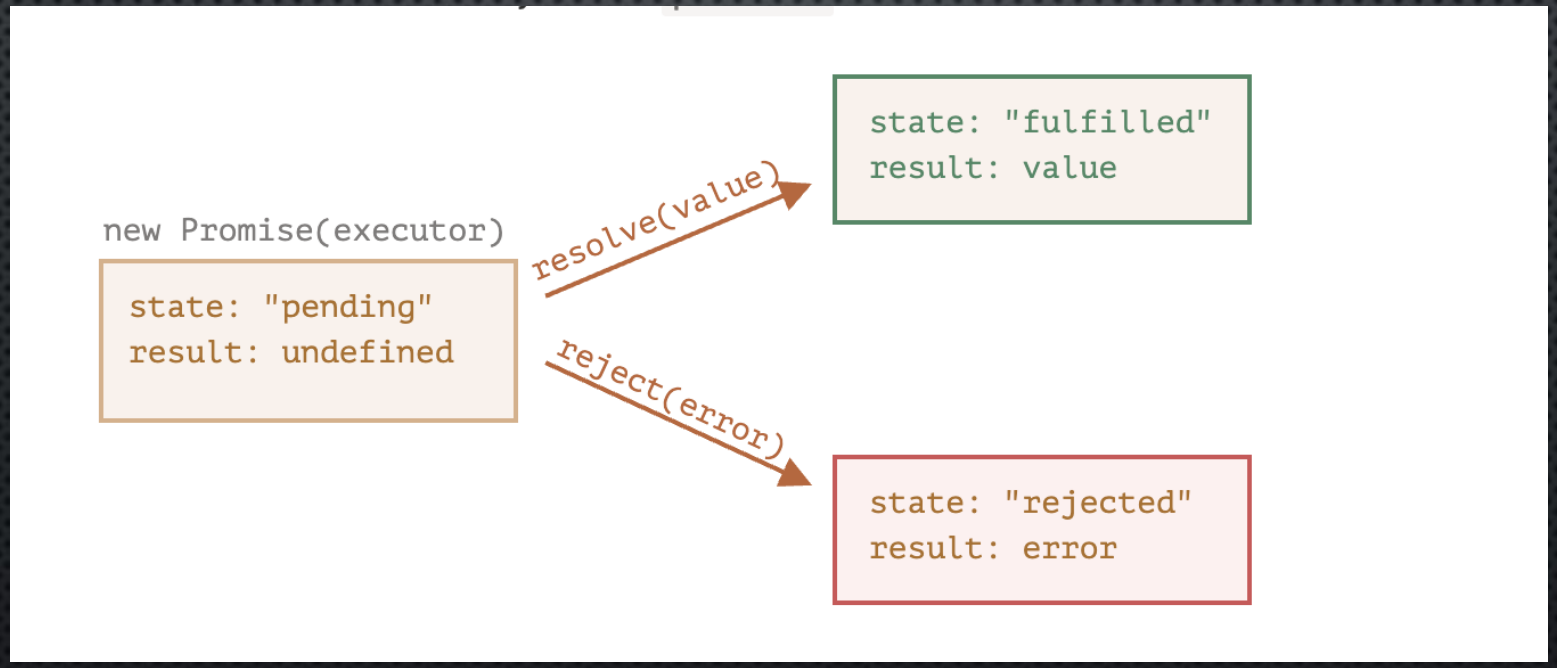
The function returns right away to be responsive to other events.

The program is notified when the operation is complete.

"I PROMISE A RESULT"

- A promise being "a JavaScript object that links producing code and consuming code. Producing code taking some time, a call to an API for example, and consuming code waiting for the result. Let's say displaying the returned information on a web page.

PROMISE STATES

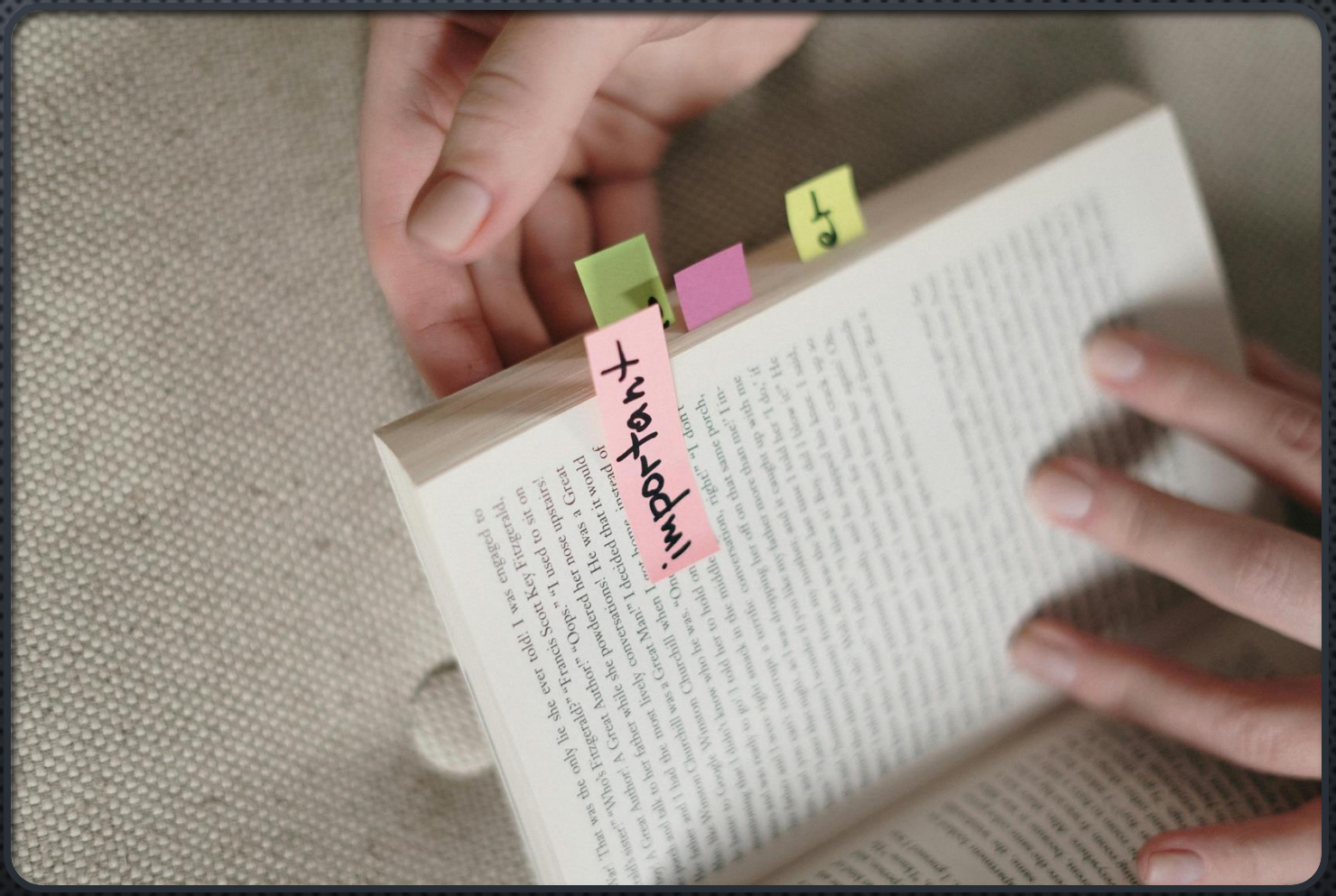


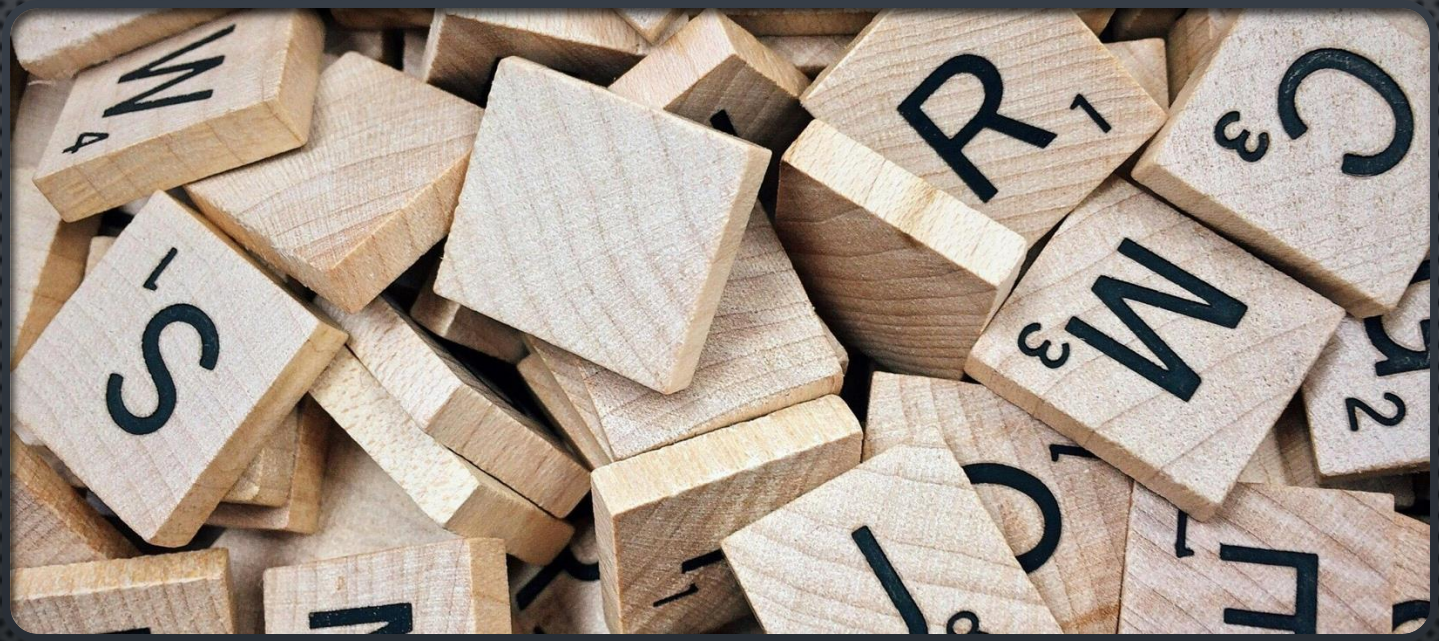
YOU CAN THINK OF PROMISES LIKE AN ASSEMBLY LINE

- THE LINE ALWAYS KEEPS MOVING
- BUT BEFORE THE NEXT ASSEMBLER CAN WORK, THE PREVIOUS ASSEMBLER MUST PASS THE PARTIALLY ASSEMBLED ITEM TO THEM FOR FURTHER WORK.

SO WHY IS THIS IMPORTANT?

- IF IN YOUR JS PROGRAMMING YOU'RE ASSIGNING A VALUE TO A VARIABLE (OR VALUE TO A DOM OBJECT)BEFORE AN ASYNC CALL HAS RESOLVED YOU'RE GOING TO GET A NULL VALUE ERROR.
- SO, IT REQUIRES CODE TO BE WELL-STRUCTURED

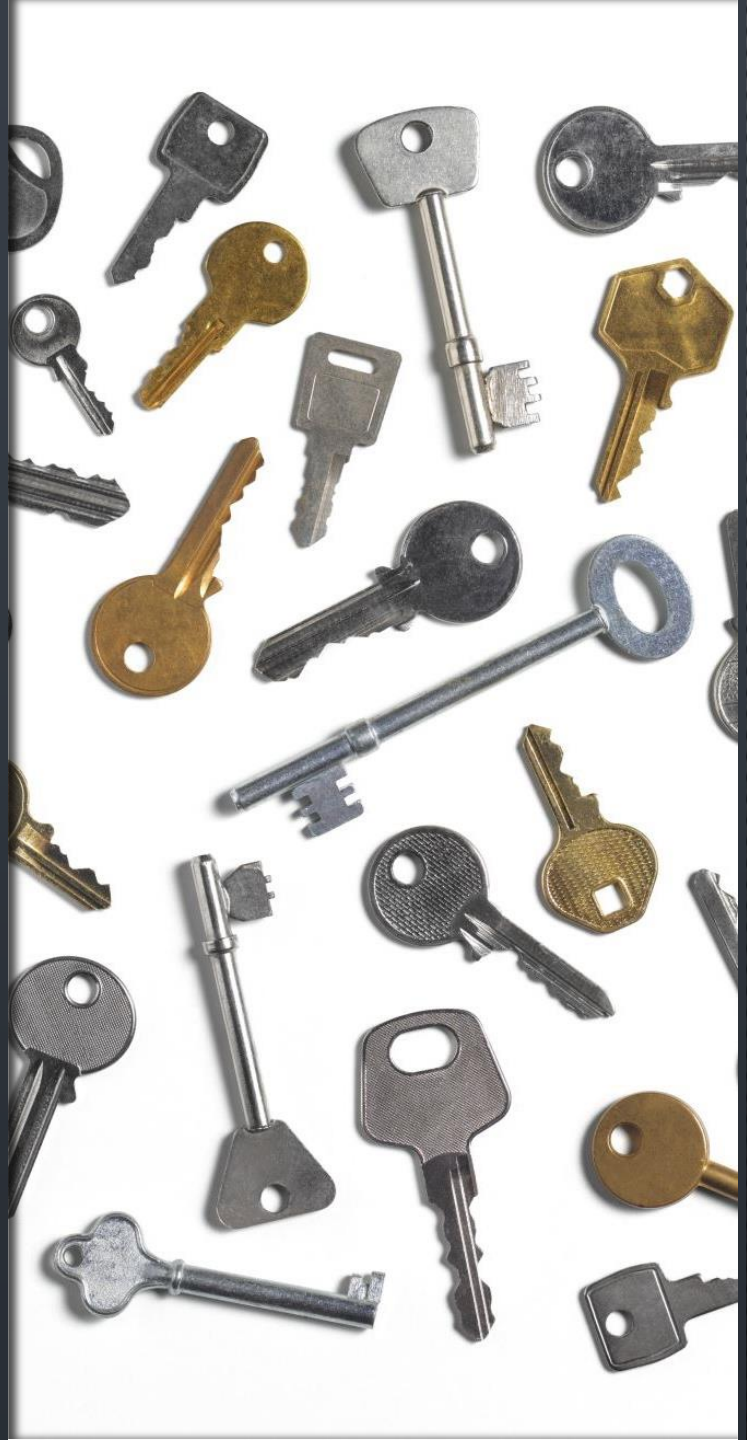




HERE'S A DEMONSTRATION OF AN ASYNCHRONOUS API CALL TO A WEBSITE THAT RETURNS RANDOM WORDS

Async/await vs fetch

- Fetch (ES6)
- Async/await (ES7)
- When making async requests, you can use either then() or async/await
- They are very similar
- Async/await will pause the function execution until the promise settles. With then() then function will continue to execute but JS won't execute the .then() callback until the promise settles.
- Write your code as if you were writing synchronous code



Adding a foreign key to a model which already has instances can throw an error

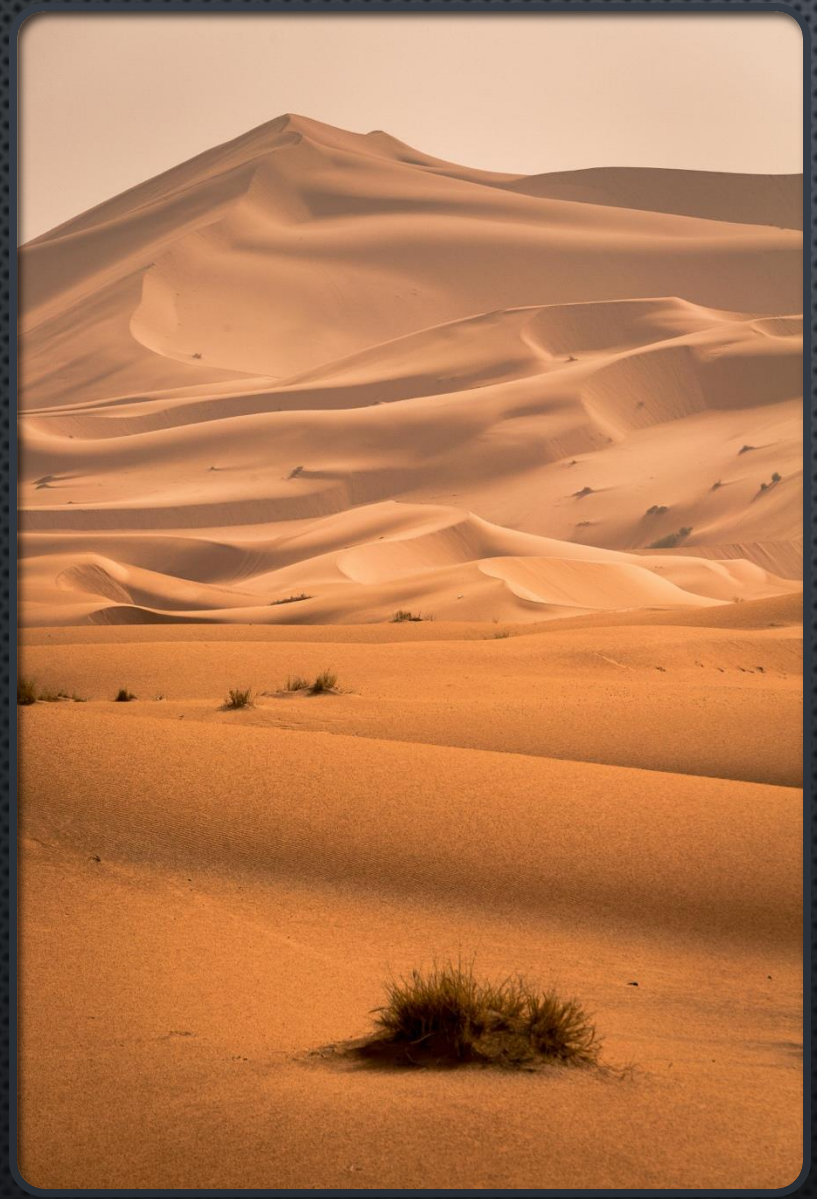
- Why?
- Because the new ForeignKey field requires a value for all existing records of a model, but they don't have one yet.
- What are the solutions?

IntegrityError: NOT NULL constraint failed

- 1) Allow NULL values (Temporarily, hopefully)
  - Add *null=True* to your foreign key model function
- 2) Provide a default value in your foreign key model function
  - *Default=<some_existing instance_id>*

- If you need to start over:
  - Delete your migrations and pycache file
  - Keep the __init__.py
  - Delete the db.sqlite3 file
  - Make a new migration
  - Migrate

- In most cases it's better not to allow NULL for a ForeignKey for increased data integrity



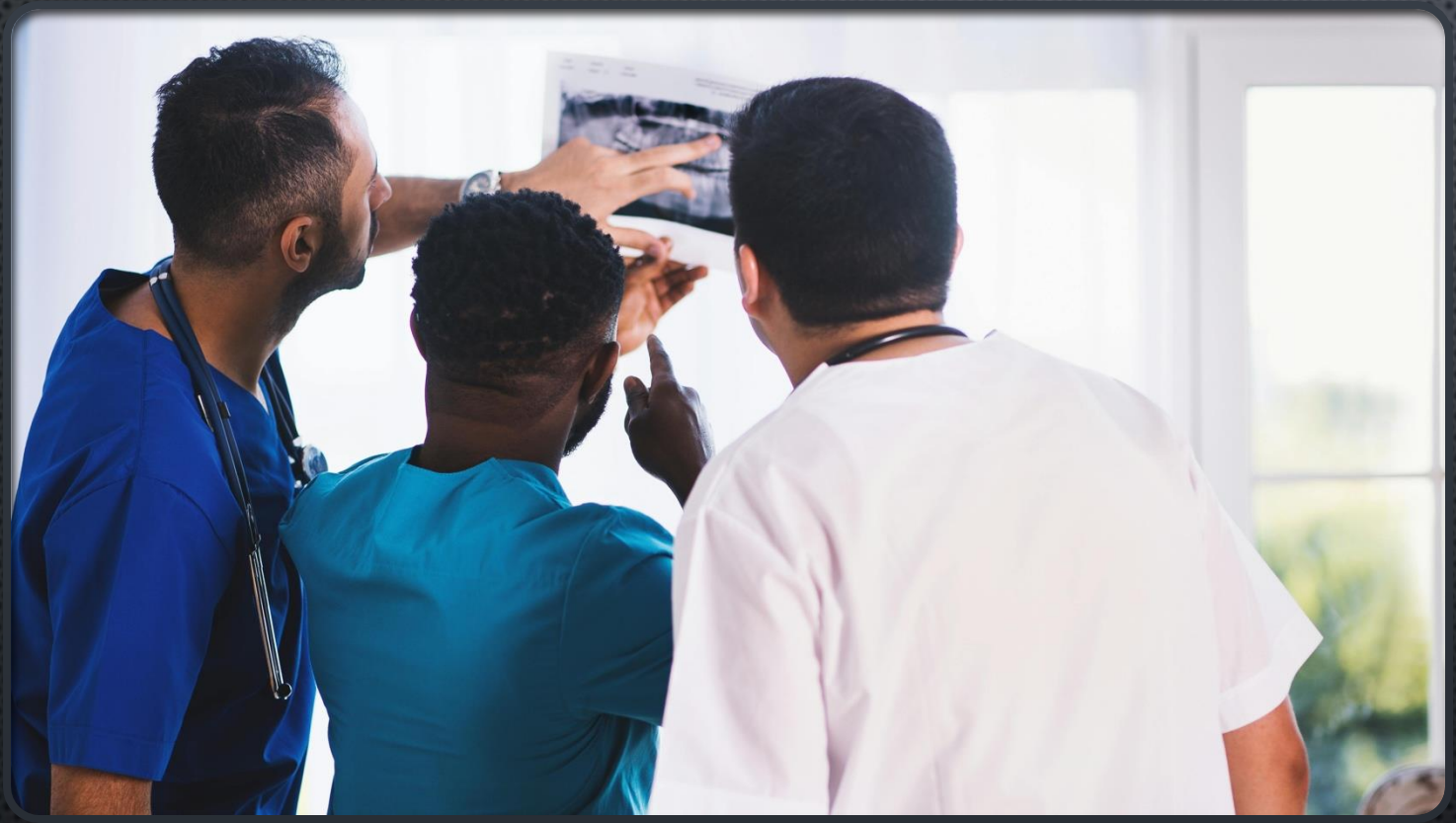
IN DJANGO STRUCTURING DATA EFFICIENTLY CAN MAKE THE REST OF YOUR CODE **DRY**ER

(Don't Repeat Yourself)

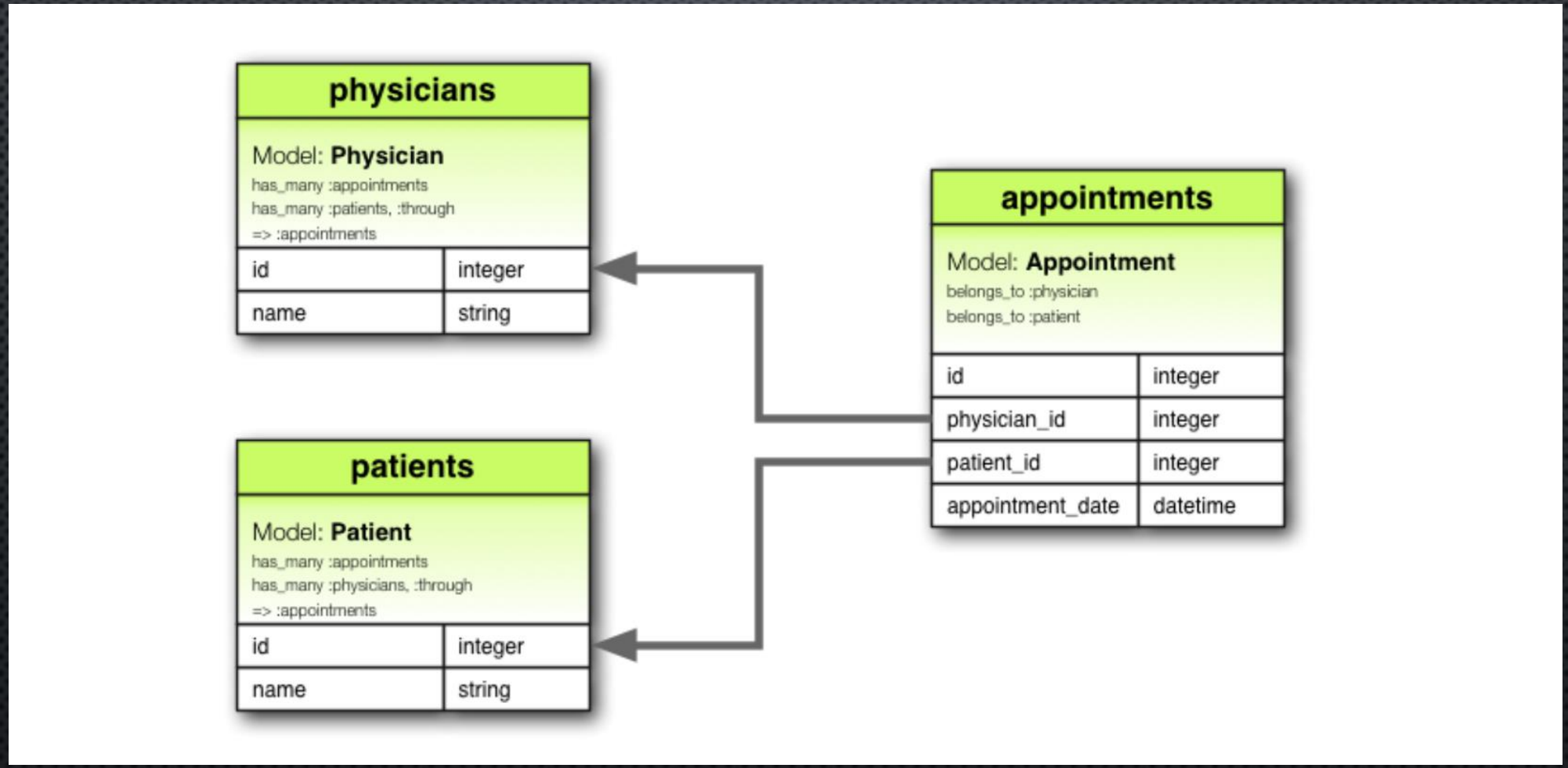
HOW WOULD WE MODEL THIS?

We want to design a simple database structure for tracking appointments in a multi-doctor clinic.

In a very basic way, how could we represent that in SQL relations (tables)?



LET GO INTO BREAKOUT GROUPS FOR A FEW MINUTES AND FIND A WAY TO MODEL THIS


physicians
Model: Physician
has_many :appointments
has_many :patients, :through
=> :appointments
id
integer
name
string
patients
Model: Patient
has_many :appointments
has_many :physicians, :through
=> :appointments
id
integer
name
string
appointments
Model: Appointment
belongs_to :physician
belongs_to :patient
id
integer
physician_id
integer
patient_id
integer
appointment_date
datetime


MAIL ASSIGNMENT
LET'S HAVE A QUICK LOOK AT IT

MAIL ASSIGNMENT

We've written all the python code for you

JavaScript and HTML is all you have to worry about.

MAIL ASSIGNMENT

Read the directions carefully

And…

HAVE A GREAT WEEK!