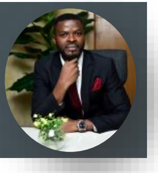
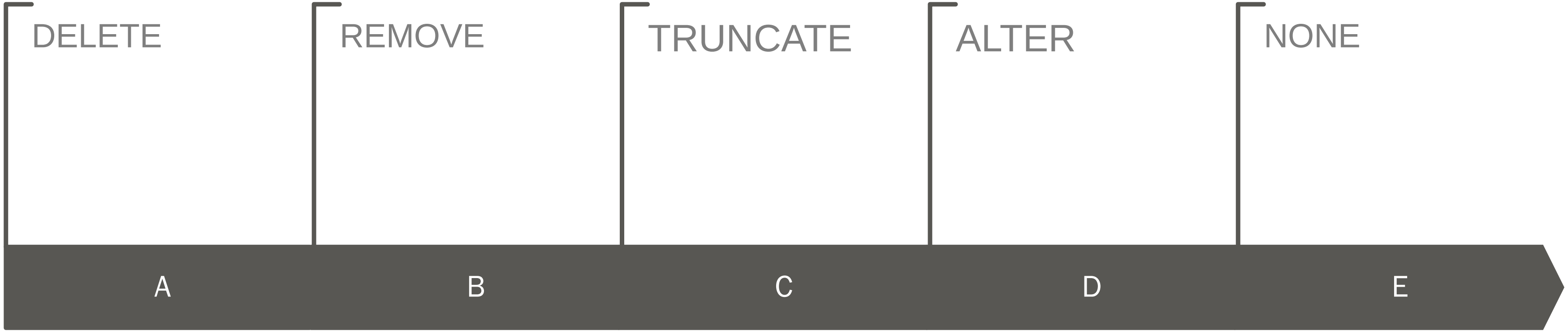
FULL-STACK NANODEGREE SESSION 4

AJIROGHENE SUNDAY



Ajiroghene Sunday | LinkedIn: in/sunday-ajiroghene

1. TO DELETE ALL DATA FROM A TABLE, WHICH SQL STATEMENT IS USED?



DELETE

REMOVE

TRUNCATE

ALTER

NONE

A

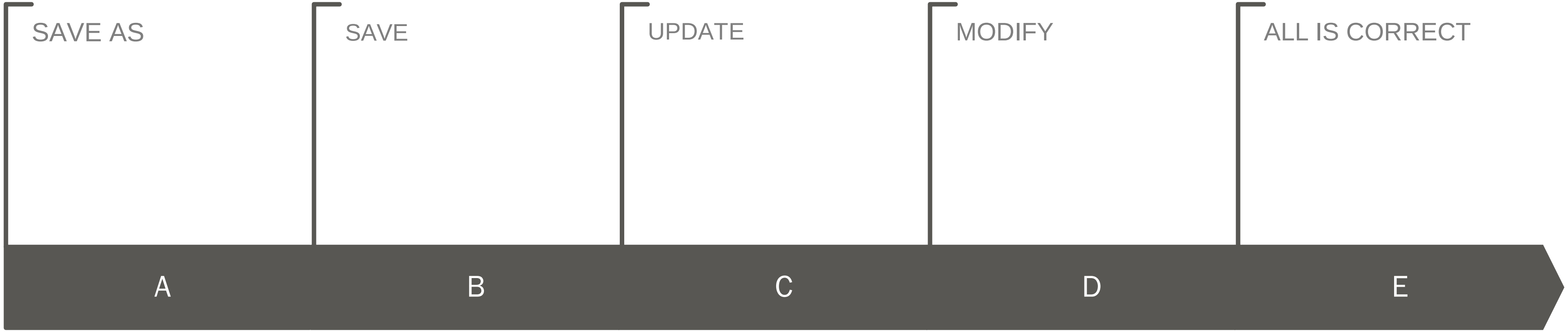
B

C

D

E

2. TO UPDATE DATA IN A DATABASE, WHICH SQL STATEMENT IS USED?



A SAVE AS

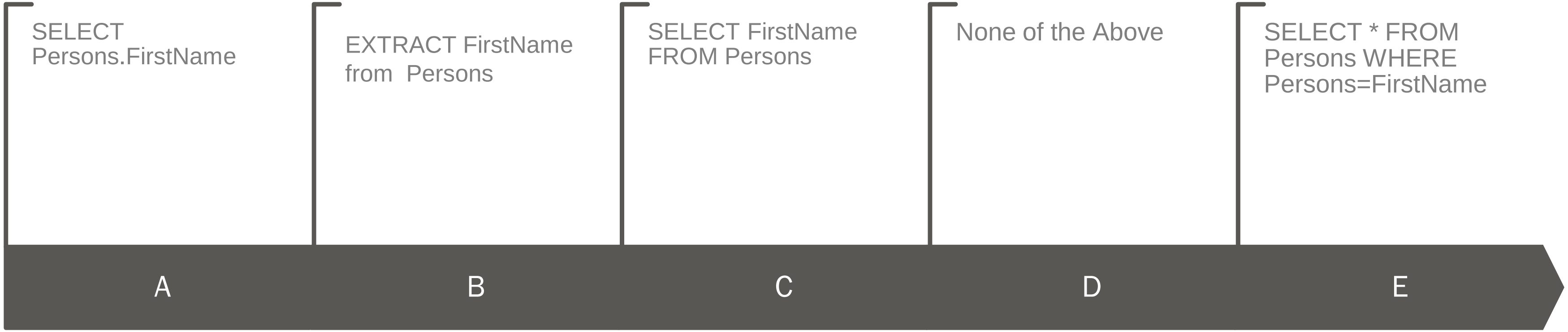
B SAVE

C UPDATE

D MODIFY

E ALL IS CORRECT

3 HOW DO YOU GET A COLUMN CALLED "FIRSTNAME" FROM A TABLE CALLED "PERSONS" IN SQL?



SELECT Persons.FirstName

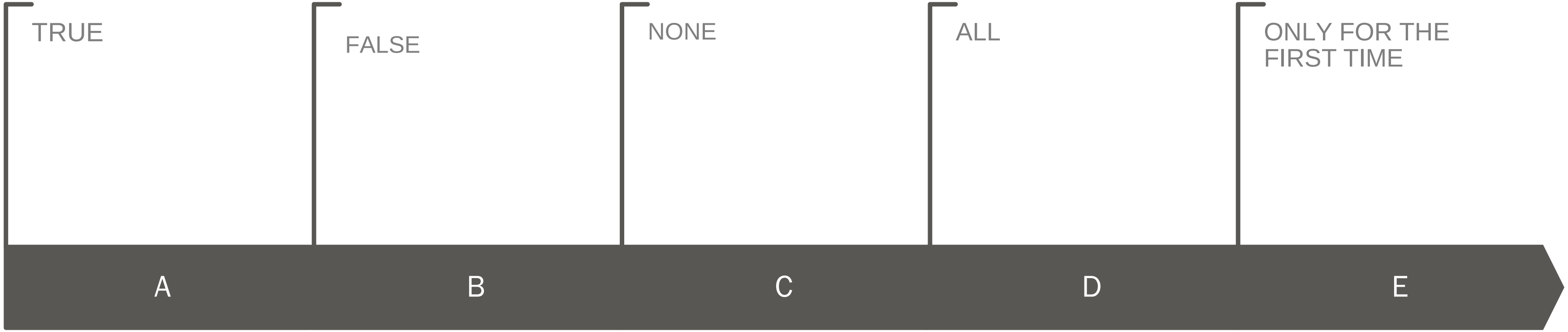
EXTRACT FirstName from Persons

SELECT FirstName FROM Persons

None of the Above

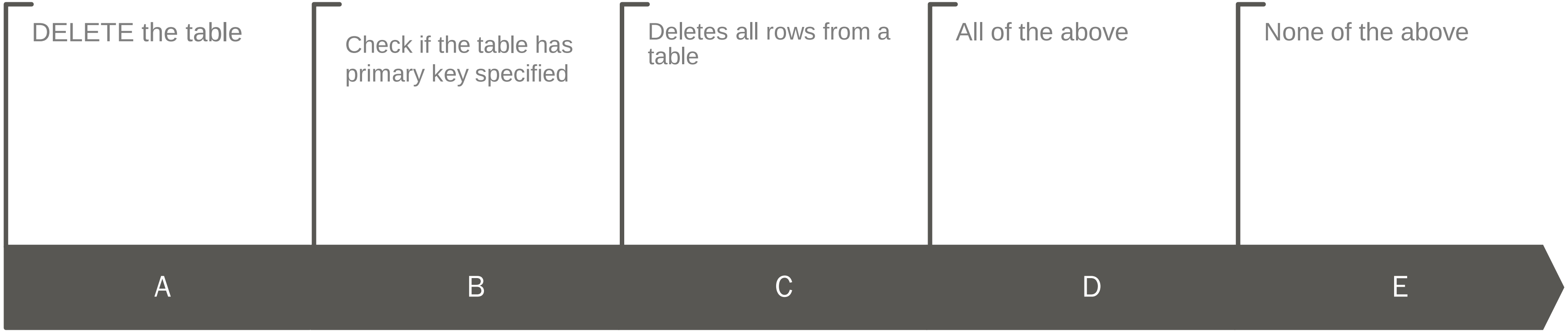
SELECT * FROM Persons WHERE Persons=FirstName

4 WHEN AUTOCOMMIT IS ENABLED, UPDATES ARE MADE AT THE END OF EACH SQL QUERY AUTOMATICALLY.



TRUE	FALSE	NONE	ALL	ONLY FOR THE FIRST TIME
A	B	C	D	E

5 WHAT DOES A TRUNCATE TABLE DO?



A — DELETE the table

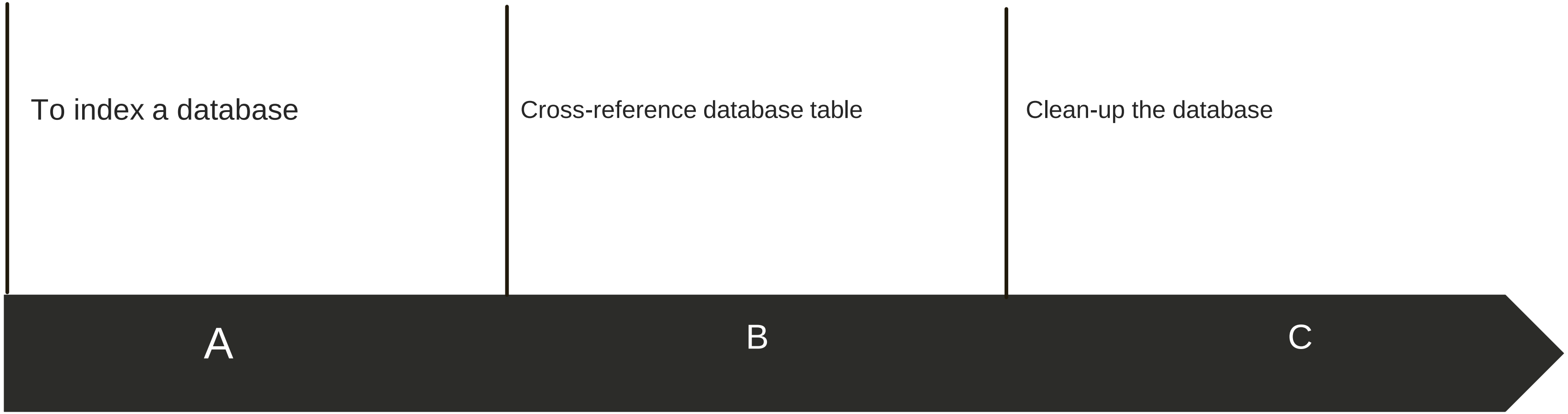
B — Check if the table has primary key specified

C — Deletes all rows from a table

D — All of the above

E — None of the above

6 THE PRIMARY - FOREIGN KEY RELATIONS ARE UTILIZED TO

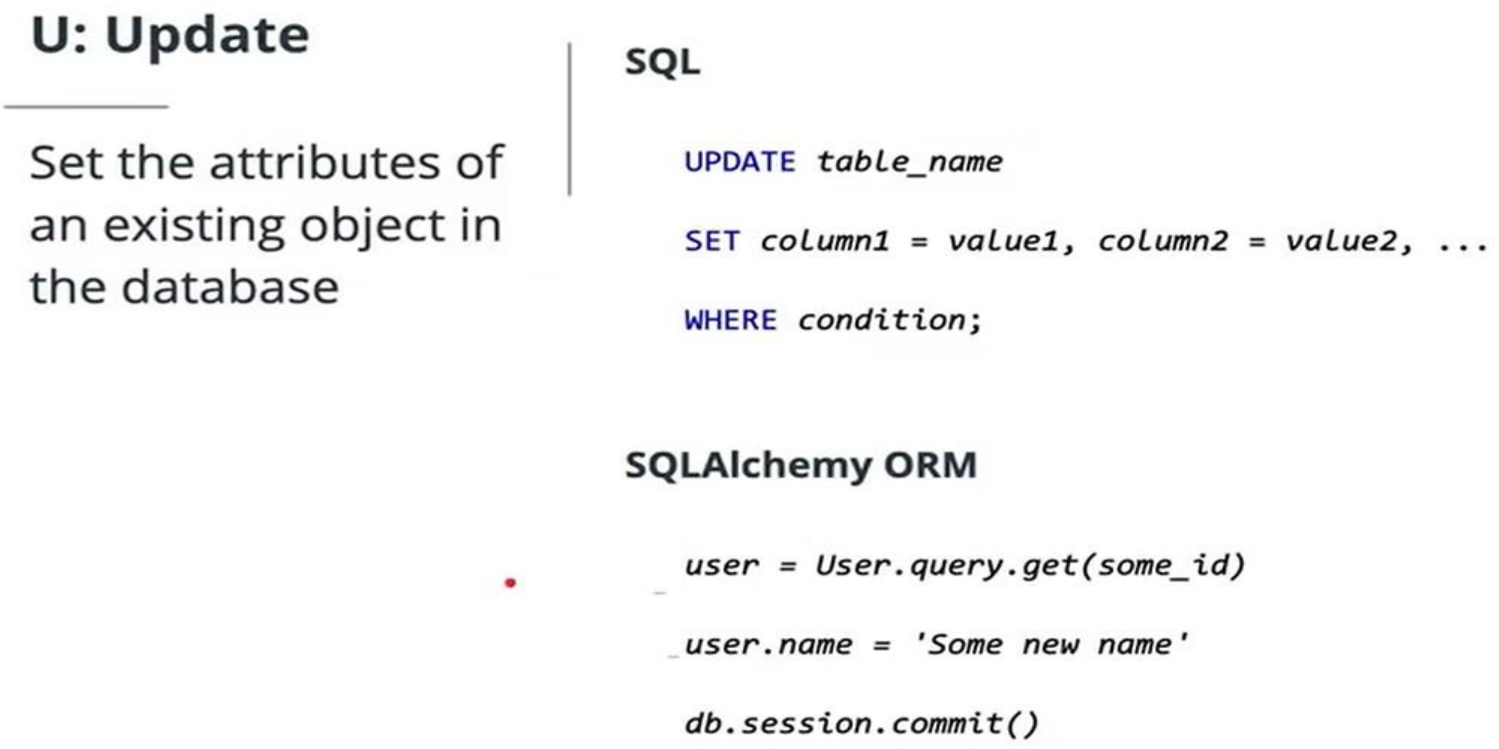


To index a database

Cross-reference database table

Clean-up the database

BONUS: MIND REWIND ON SQL



U: Update

Set the attributes of an existing object in the database

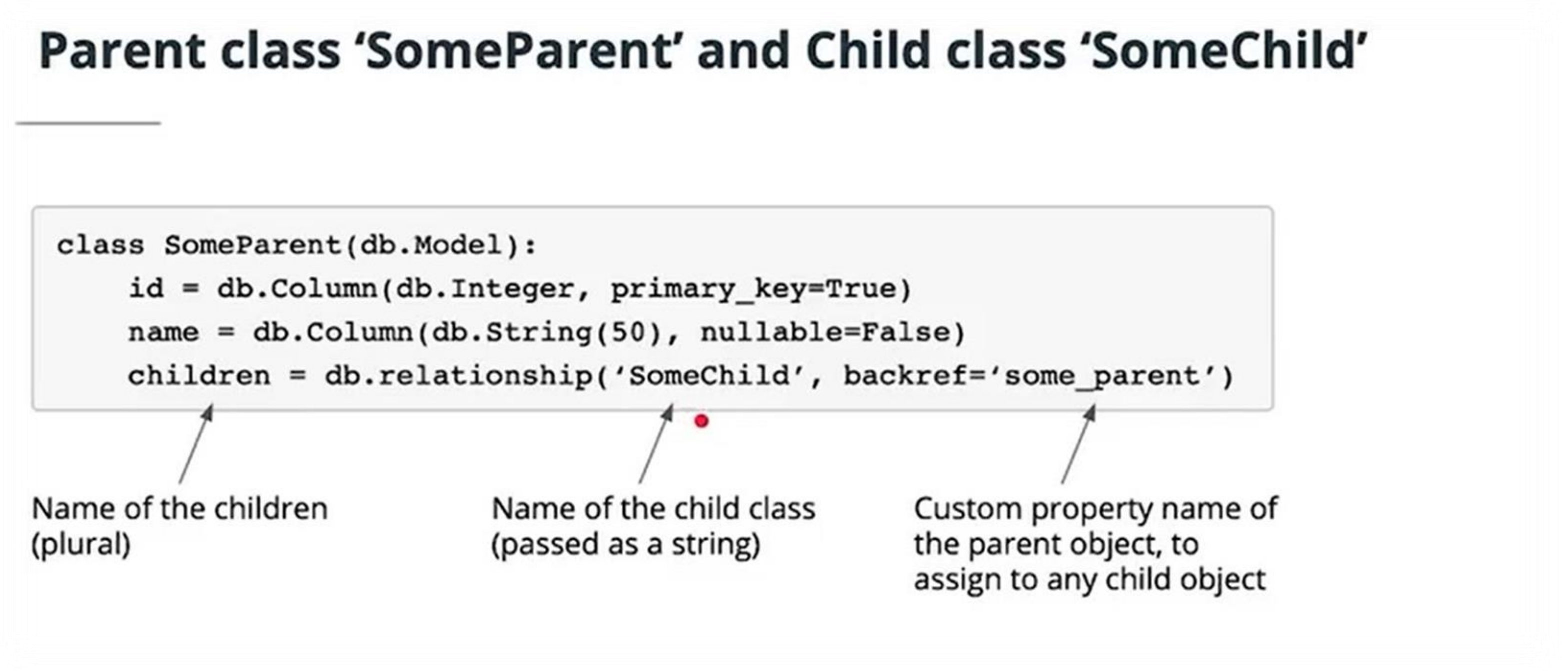
SQL

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

SQLAlchemy ORM

```
user = User.query.get(some_id)
user.name = 'Some new name'
db.session.commit()
```

BONUS: MIND REWIND



Parent class ‘SomeParent’ and Child class ‘SomeChild’

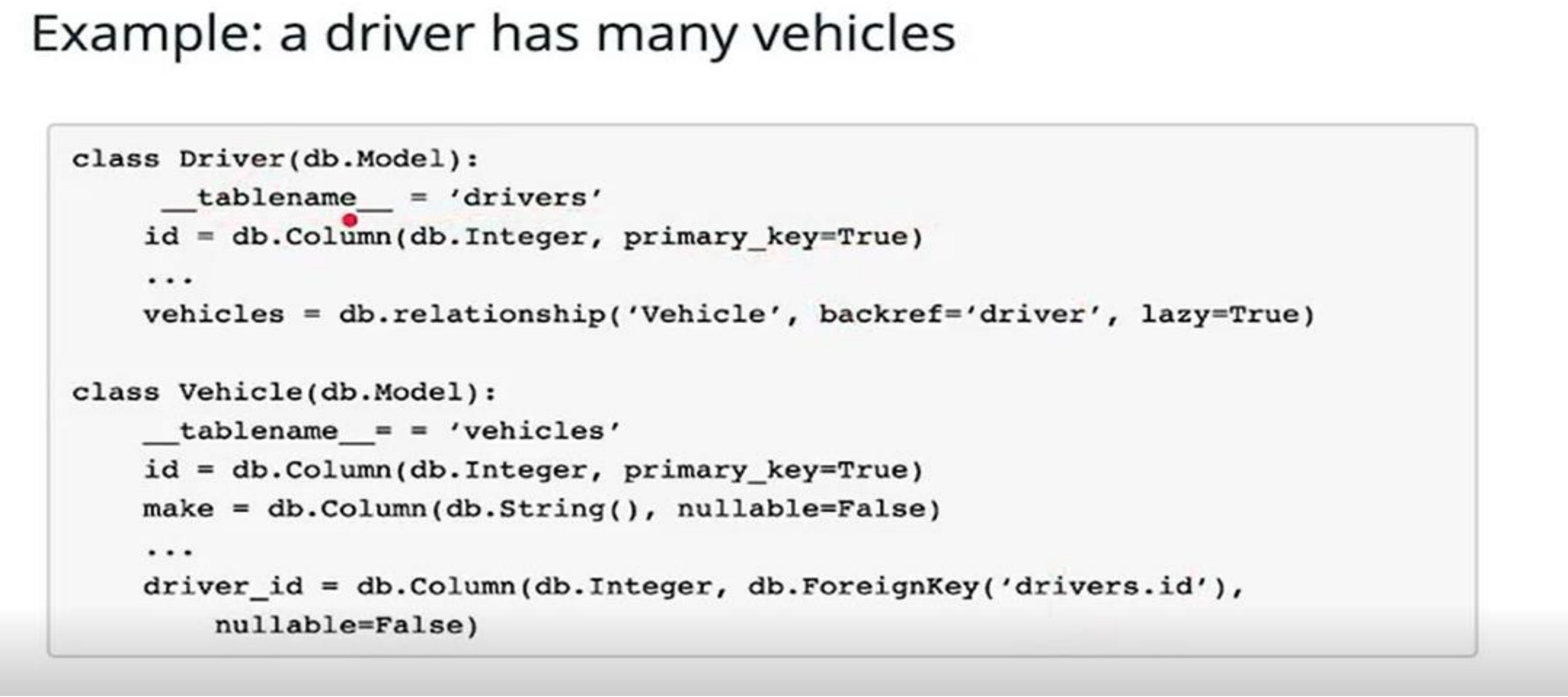
```
class SomeParent(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)
    children = db.relationship(‘SomeChild’, backref=‘some_parent’)
```

Name of the children
(plural)

Name of the child class
(passed as a string)

Custom property name of
the parent object, to
assign to any child object

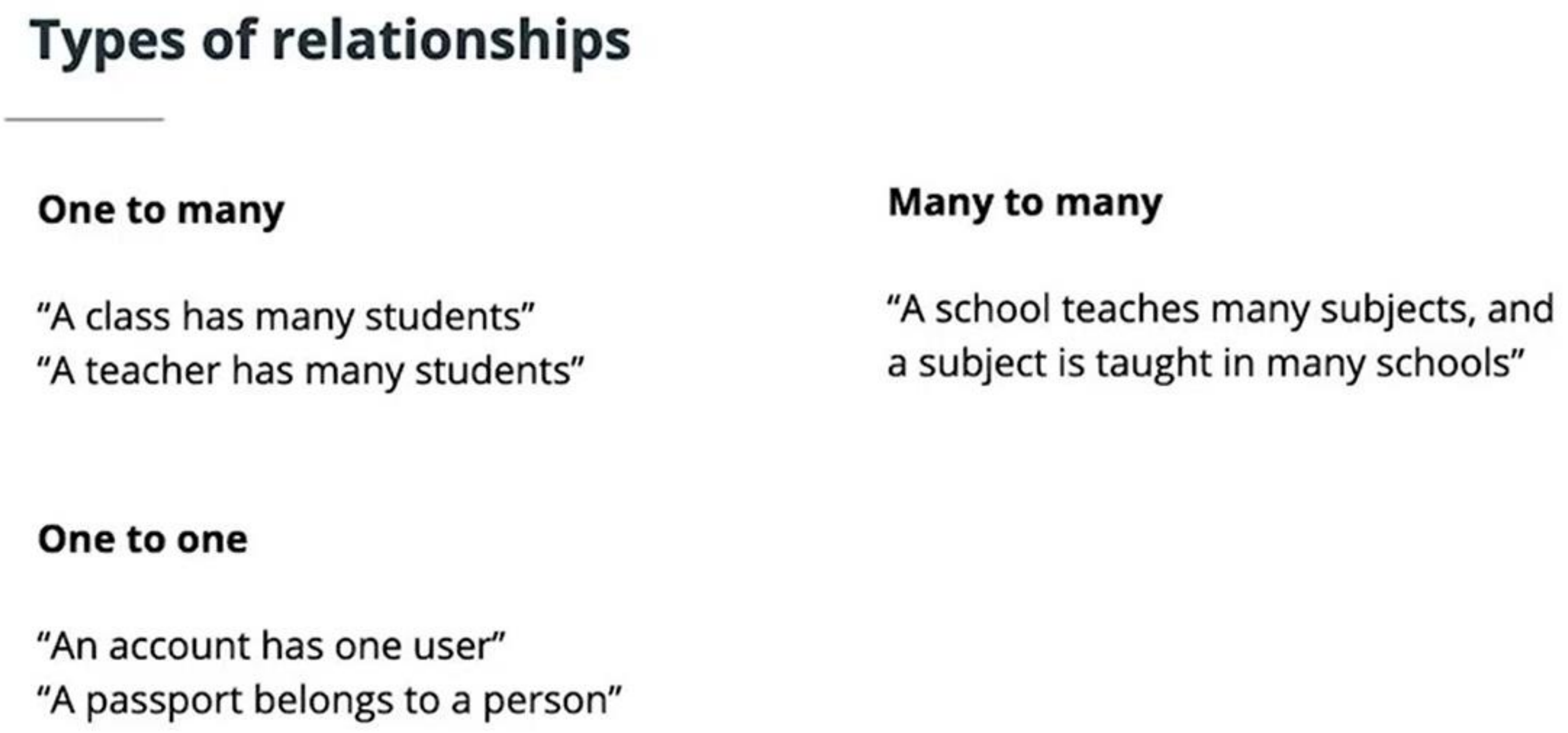
BONUS: MIND REWIND



Example: a driver has many vehicles

```
class Driver(db.Model):
     __tablename__ = ‘drivers’
    id = db.Column(db.Integer, primary_key=True)
    ...
    vehicles = db.relationship(‘Vehicle’, backref=‘driver’, lazy=True)

class Vehicle(db.Model):
    __tablename__= = ‘vehicles’
    id = db.Column(db.Integer, primary_key=True)
    make = db.Column(db.String(), nullable=False)
    ...
    driver_id = db.Column(db.Integer, db.ForeignKey(‘drivers.id’),
        nullable=False)
```

BONUS: MIND REWIND



Types of relationships

One to many

"A class has many students"
"A teacher has many students"

One to one

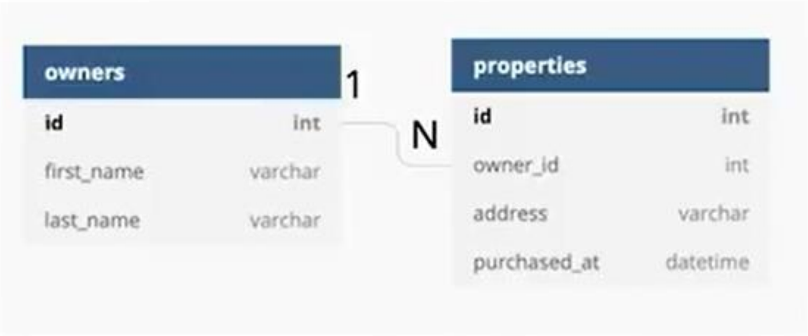
"An account has one user"
"A passport belongs to a person"

Many to many

"A school teaches many subjects, and a subject is taught in many schools"

BONUS: MIND REWIND

Types of relationships

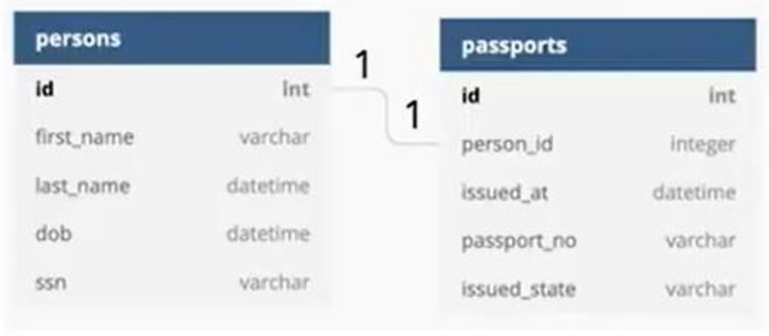


owners	
id	int
first_name	varchar
last_name	varchar

properties	
id	int
owner_id	int
address	varchar
purchased_at	datetime

One to many

"An owner has many properties"
(`properties` stores foreign key `owner_id`)



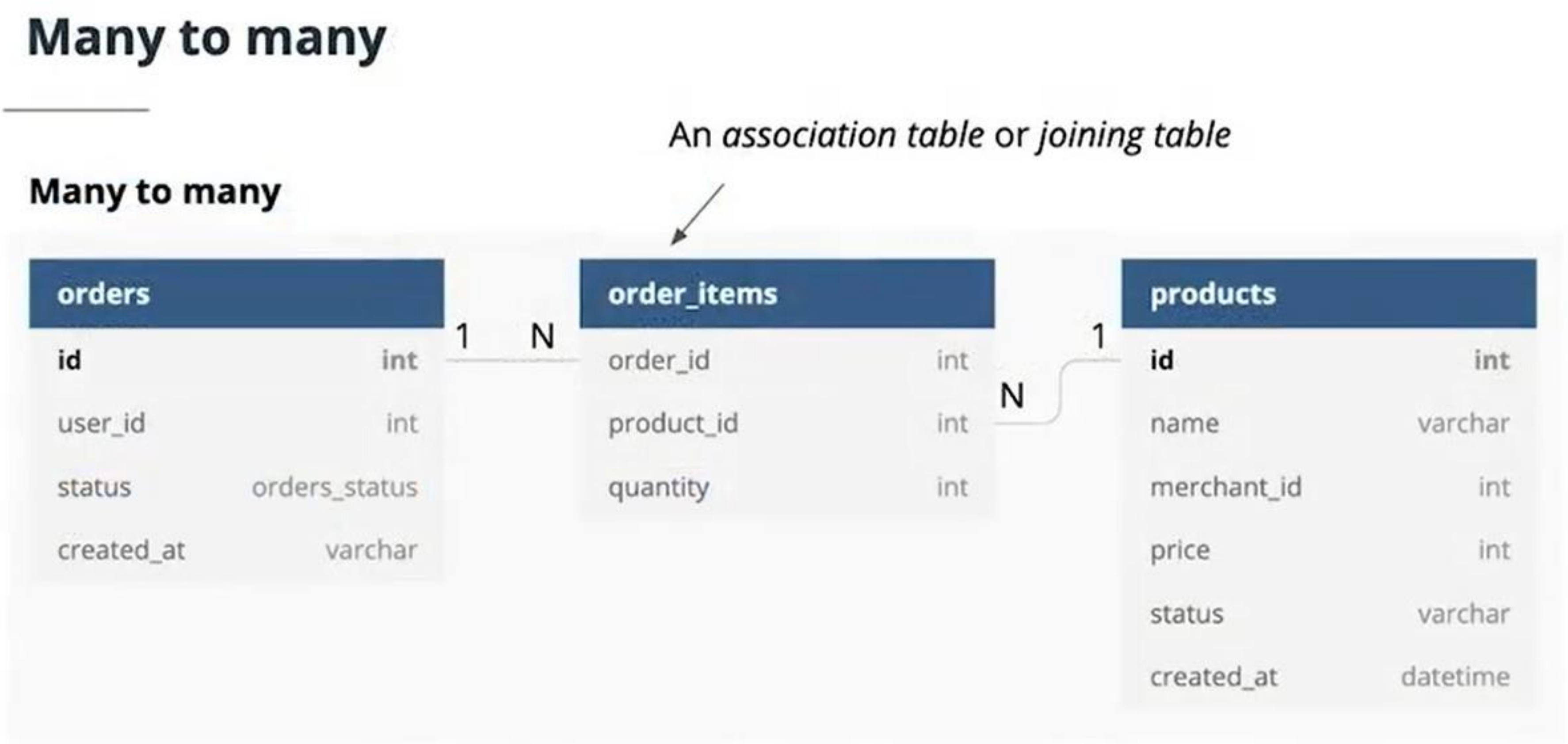
persons	
id	int
first_name	varchar
last_name	datetime
dob	datetime
ssn	varchar

passports	
id	int
person_id	integer
issued_at	datetime
passport_no	varchar
issued_state	varchar

One to one

"A passport belongs to a person"
(`passports` stores foreign key `person_id`)

BONUS: MIND REWIND



Many to many

An *association table* or *joining table*

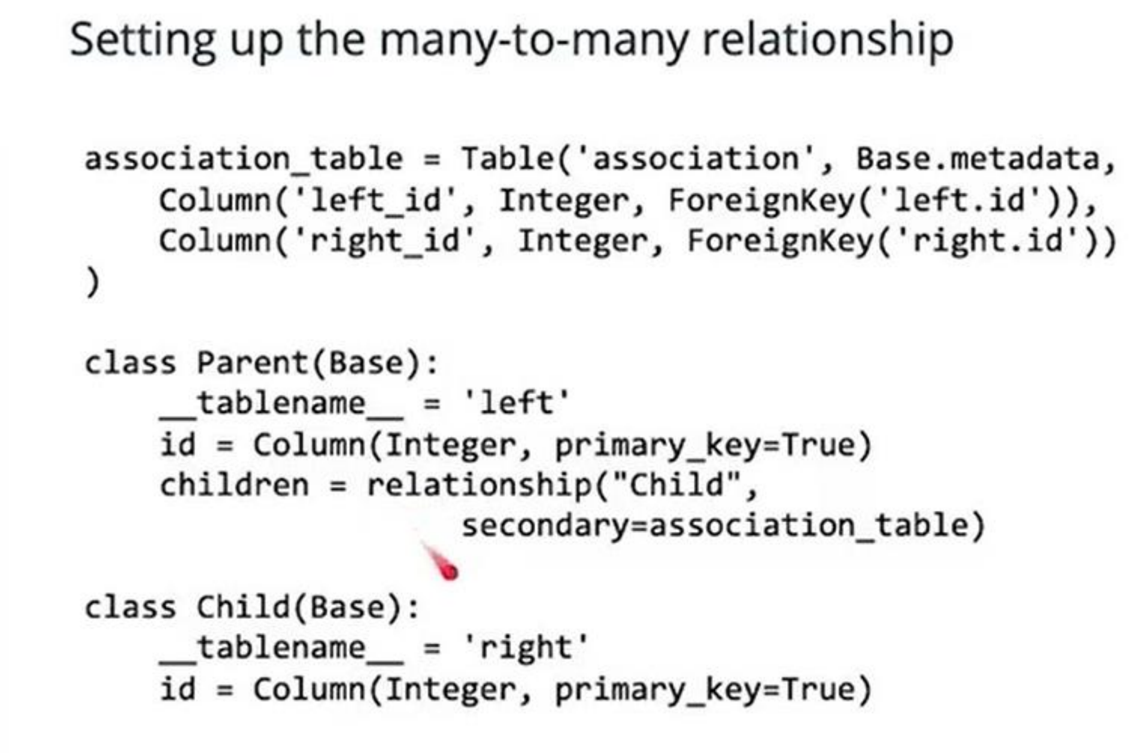
Many to many

orders	
id	**int**
user_id	int
status	orders_status
created_at	varchar

order_items	
order_id	int
product_id	int
quantity	int

products	
id	**int**
name	varchar
merchant_id	int
price	int
status	varchar
created_at	datetime

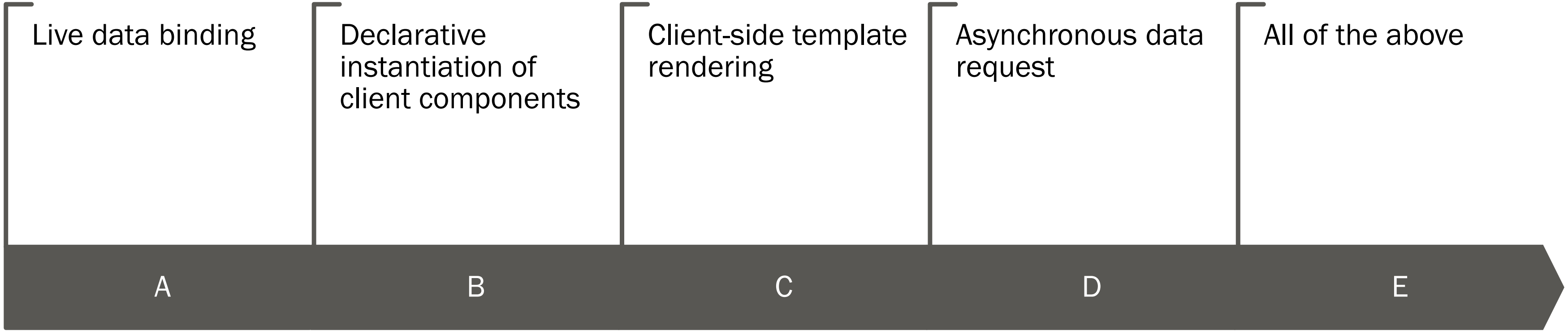
BONUS: MIND REWIND



Setting up the many-to-many relationship

```
association_table = Table('association', Base.metadata,
    Column('left_id', Integer, ForeignKey('left.id')),
    Column('right_id', Integer, ForeignKey('right.id'))
)

class Parent(Base):
    __tablename__ = 'left'
    id = Column(Integer, primary_key=True)
    children = relationship("Child",
                    secondary=association_table)

class Child(Base):
    __tablename__ = 'right'
    id = Column(Integer, primary_key=True)
```

7 WHICH OF THE FOLLOWING ARE THE FEATURES OF AJAX?



Live data binding

A

Declarative instantiation of client components

B

Client-side template rendering

C

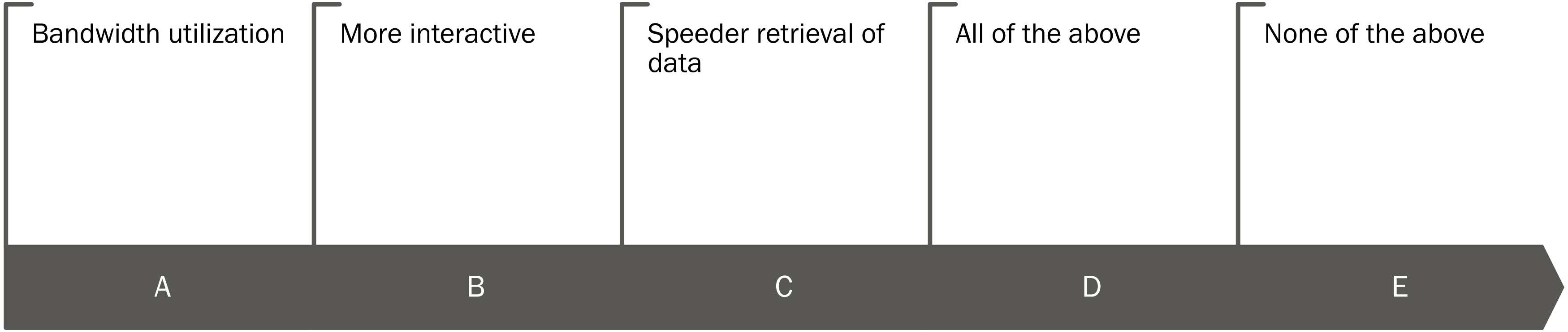
Asynchronous data request

D

All of the above

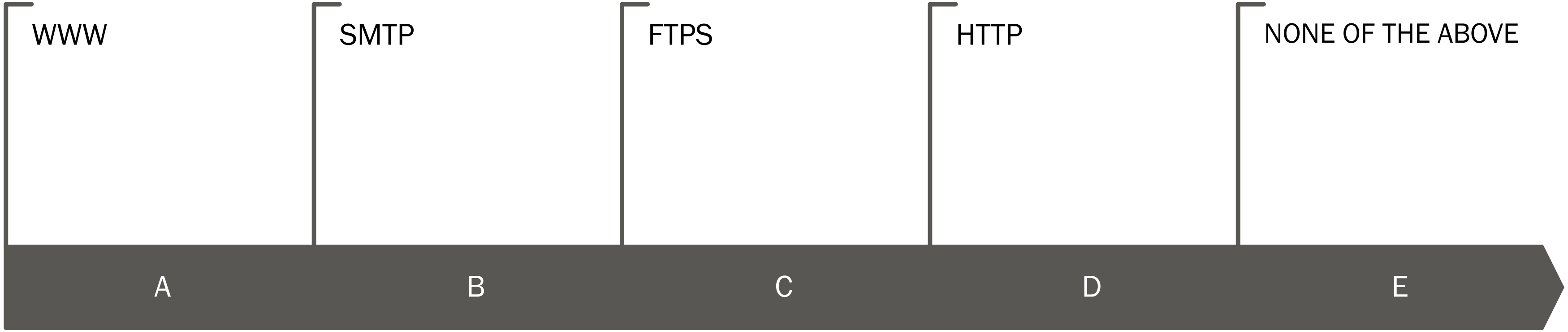
E

8. THE ADVANTAGES OF AJAX IS _______________ .



Bandwidth utilization	More interactive	Speeder retrieval of data	All of the above	None of the above
A	B	C	D	E

9. ____________ IS THE SEVER SUPPORT AJAX ?



A WWW

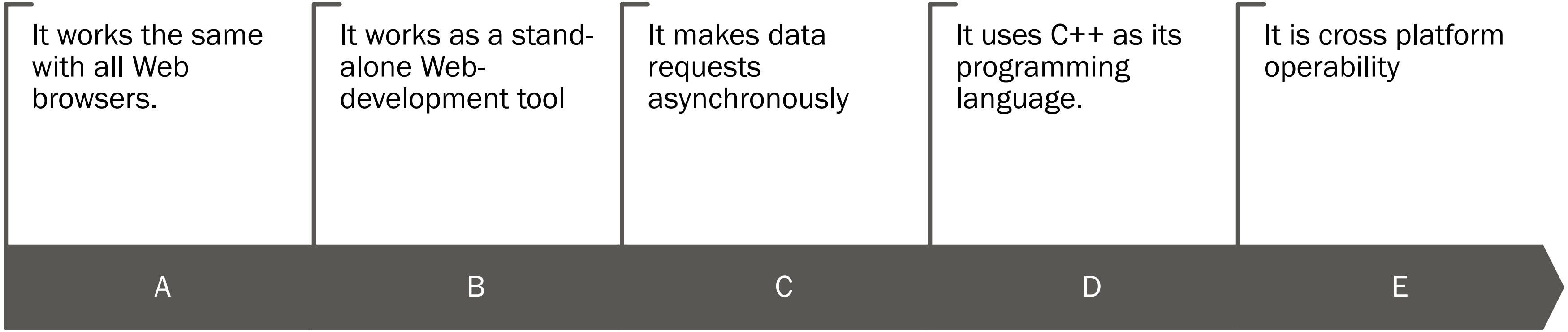
B SMTP

C FTPS

D HTTP

E NONE OF THE ABOVE

10. WHICH OF THE FOLLOWING MAKES AJAX UNIQUE?



A. It works the same with all Web browsers.

B. It works as a stand-alone Web-development tool

C. It makes data requests asynchronously

D. It uses C++ as its programming language.

E. It is cross platform operability

TABLE OF CONTENTS

01

RECAP

LET'S DO A THROWBACK

02

CRUD

CREATE, READ, UPDATE, DELETE

03

AJAX

ASYNCHRONOUS JAVASCRIPT AND XML

04

WHAT NEXT

OUR NEXT STEP FORWARD

“PUSHING FURTHER STANDS YOU OUT.”

–*AJIROGHENE SUNDAY*

INTRODUCTION

IN TODAY'S LECTURE, WE WILL REFRESH OUR KNOWLEDGE ON CRUD OPERATIONS, MVC AND AJAX.

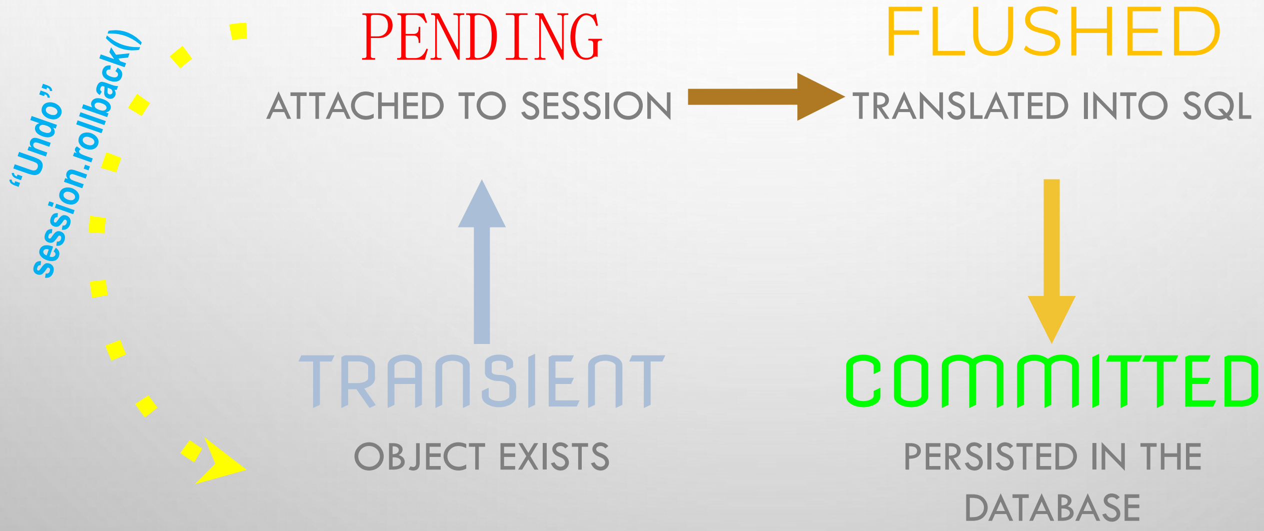
01

RECAP

INTERERSTING US

LET'S TALK TALK ABOUT ALL YOU DID OVER THE WEEK

OBJECT LIFE CYCLE



PENDING

ATTACHED TO SESSION

FLUSHED

TRANSLATED INTO SQL

TRANSIENT

OBJECT EXISTS

COMMITTED

PERSISTED IN THE DATABASE

WHAT ARE MIGRATIONS

- WHAT ARE MIGRATIONS?
  - MANAGEMENT OF INCREMENTAL, REVERSIBLE CHANGES AND VERSION CONTROL TO RELATIONAL DATABASE SCHEMAS.
- WHY DO WE USE THEM?
  - TO KEEP TRACK OF SCHEMA CHANGES
  - TO DO GRANULAR APPLICATION CHANGES OF THE SCHEMA
- HOW DO WE USE THEM?
  - FLASK DB INIT
  - FLASK DB MIGRATE
  - FLASK DB UPGRADE
  - FLASK DB DOWNGRADE

02

CRUD OPERATIONS

WHAT ARE CRUD OPERATIONS?

- CRUD OPERATIONS ARE USED TO INSERT, UPDATE, DELETE AND RETRIEVE RECORDS FROM A DATABASE.

WHY DO WE NEED CRUD?

HELPS US ORGANIZE OUR DATABASE OPERATIONS.

TYPES OF CRUD OPERATIONS

Operation	SQLAlchemy	SQL	Meaning
Create	**book = Book(title=”Python for Beginners”)**	INSERT INTO books (title) VALUES (‘PYTHON for Beginners’);	**Creates a record on the database**
Read	**Book.query.filter(Book.id=1).first( )**	SELECT * FROM books WHERE id=1;	**Retrieves a record from the database**

QUERY METHODS (FILTERING)

Method	Orm	SQL	Meaning
Update	**book.title = 'Intermediate Python'**	UPDATE book SET title='Intermediate Python' WHERE id=2;	**Updates the value of a field.**
Delete	**db.session.delete(book);**	DELETE FROM books WHERE id=2;	**Deletes a record from a database.**

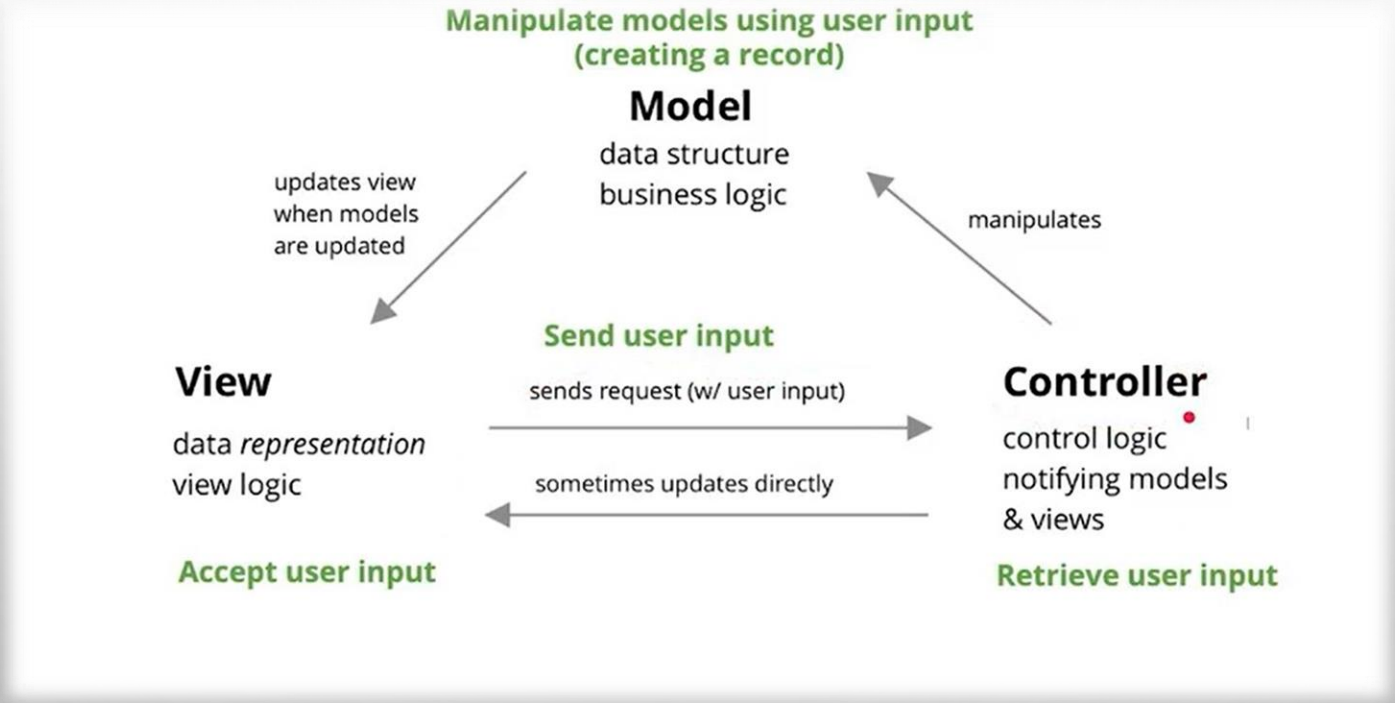
03

AJAX & METHODS

WHAT IS MVC?
(MODEL, VIEW, CONTROLLER)

- MODEL: MANAGES DATA AND BUSINESS LOGIC.
- VIEW: HANDLES DISPLAY AND REPRESENTATION LOGIC.
- CONTROLLER: ROUTES COMMANDS TO MODELS AND VIEWS.

WHAT IS MVC?
(MODEL, VIEW, CONTROLLER)



Manipulate models using user input
(creating a record)

Model

data structure
business logic

updates view
when models
are updated

manipulates

Send user input

View

data *representation*
view logic

sends request (w/ user input)

sometimes updates directly

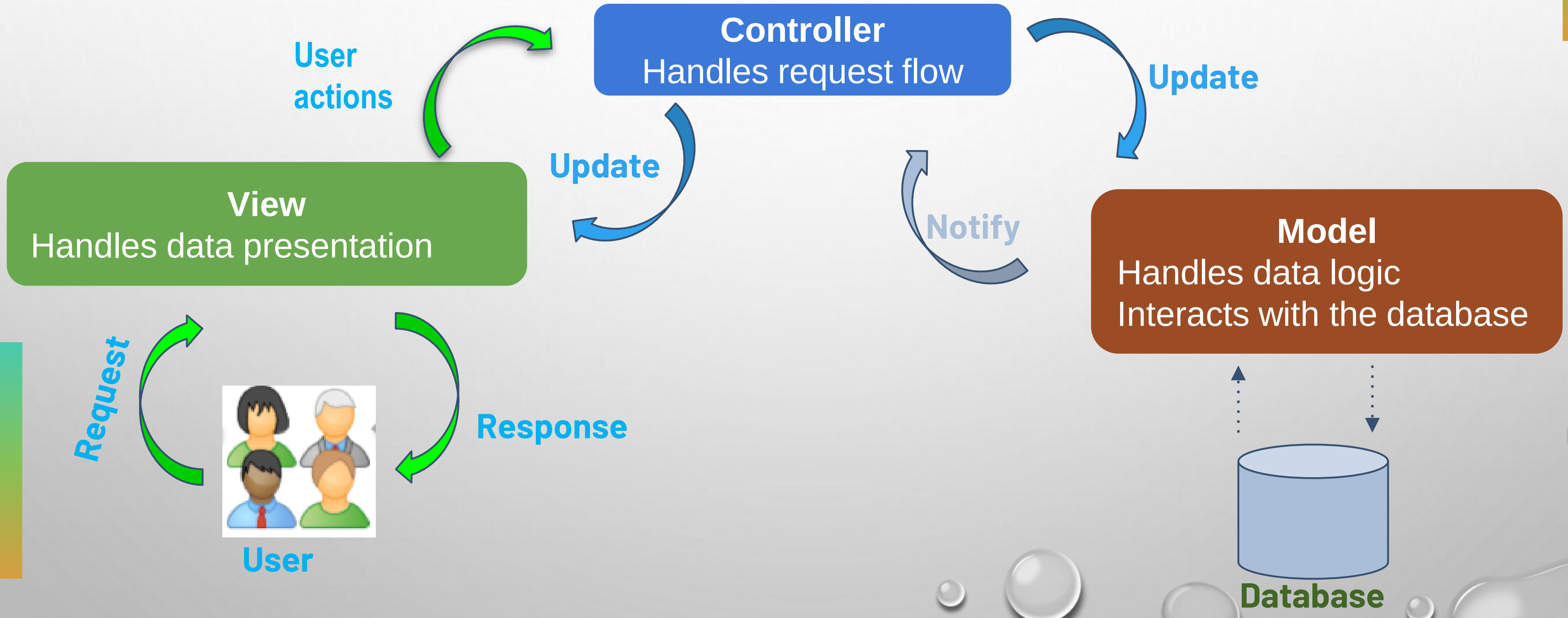
Controller

control logic
notifying models
& views

Accept user input

Retrieve user input

MODEL, VIEW, CONTROLLER

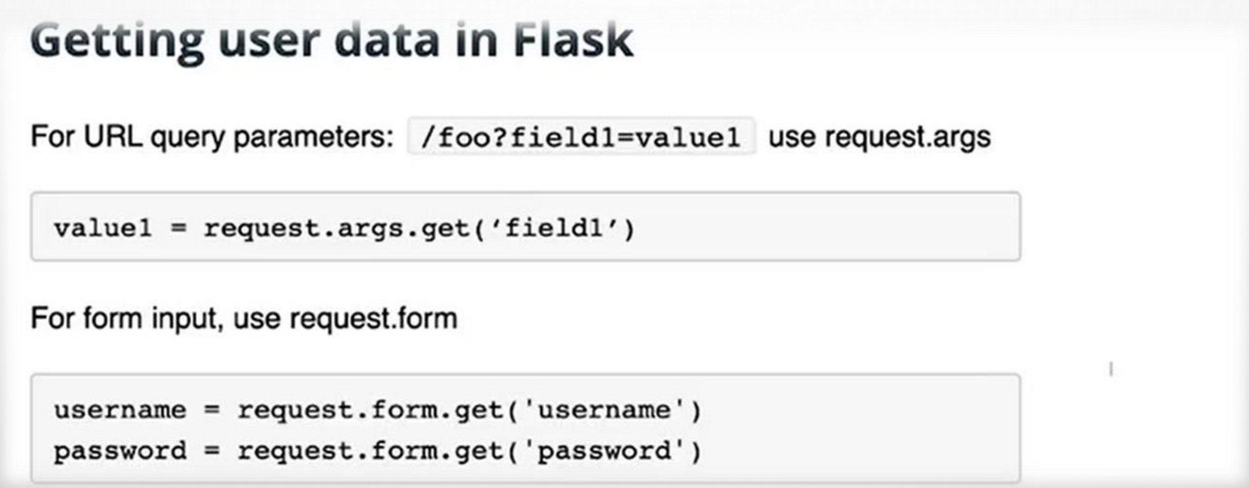


Controller
Handles request flow

View
Handles data presentation

Model
Handles data logic
Interacts with the database

APPLICATION



Getting user data in Flask

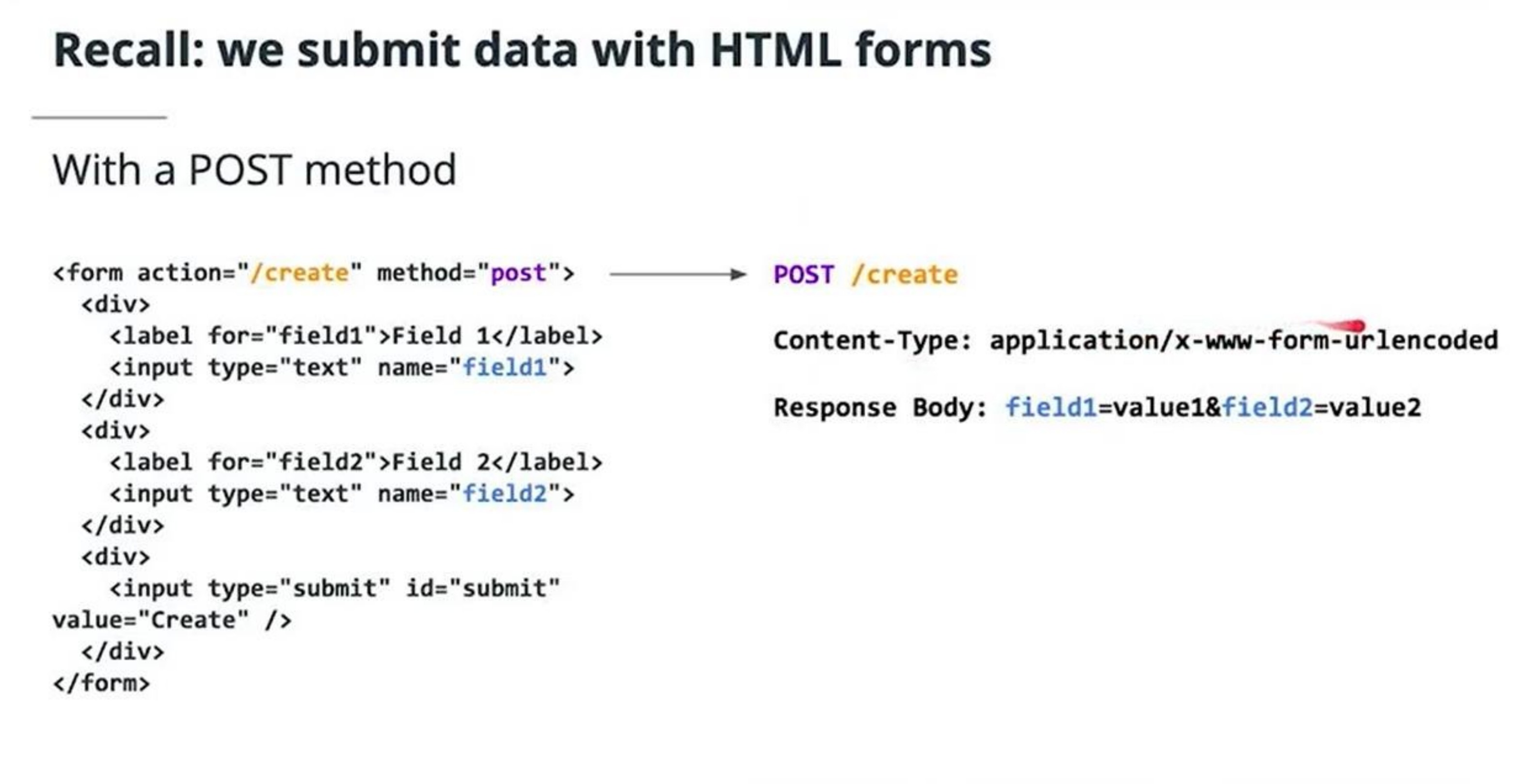
For URL query parameters: `/foo?field1=value1` use request.args

```
value1 = request.args.get(‘field1’)
```

For form input, use request.form

```
username = request.form.get('username')
password = request.form.get('password')
```

APPLICATION



Recall: we submit data with HTML forms

With a POST method

```
<form action="/create" method="post">
  <div>
    <label for="field1">Field 1</label>
    <input type="text" name="field1">
  </div>
  <div>
    <label for="field2">Field 2</label>
    <input type="text" name="field2">
  </div>
  <div>
    <input type="submit" id="submit"
value="Create" />
  </div>
</form>
```

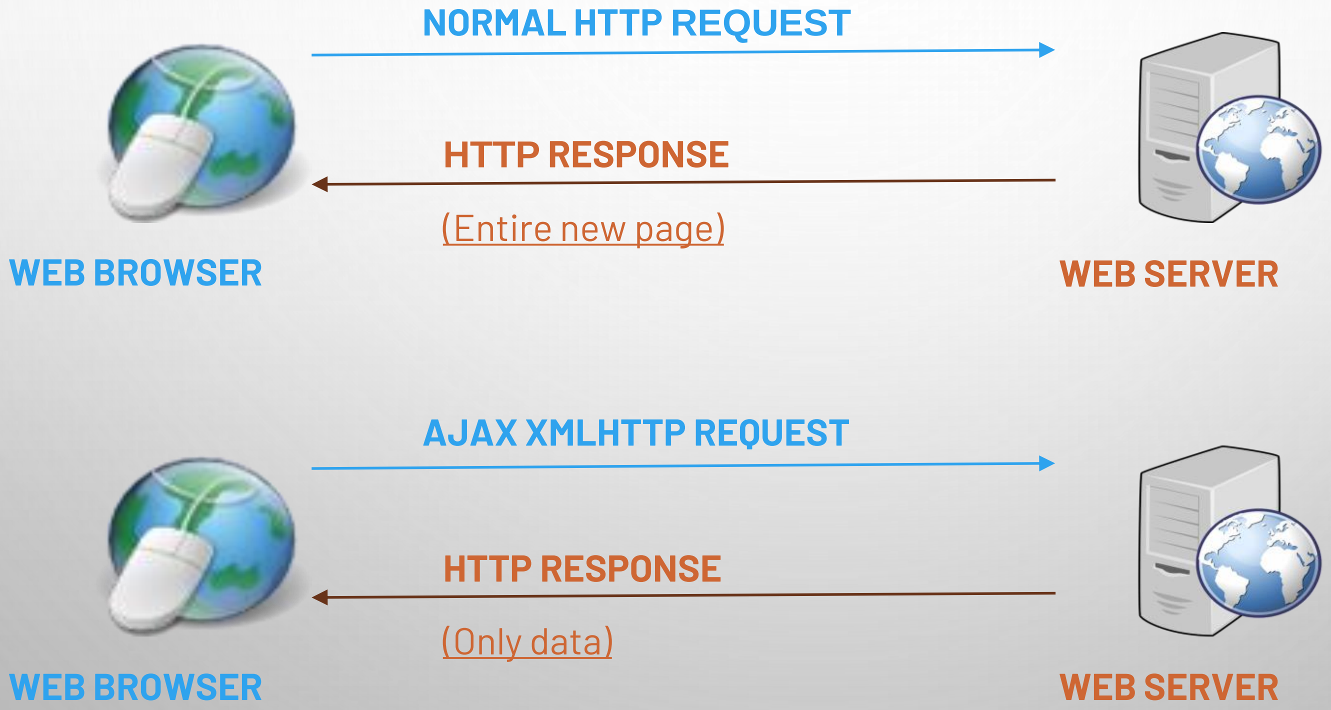
```
POST /create

Content-Type: application/x-www-form-urlencoded

Response Body: field1=value1&field2=value2
```

AJAX

- AJAX STANDS FOR ASYNCHRONOUS JAVASCRIPT AND XML

- IT IS NOT A PROGRAMMING LANGUAGE, IT IS TECHNIQUE USED TO SEND AND RECEIVE DATA IN THE BACKGROUND WITHOUT REFRESHING THE WEB PAGE.

NORMAL HTTP REQUEST VS AJAX



NORMAL HTTP REQUEST

HTTP RESPONSE

(Entire new page)

WEB BROWSER

WEB SERVER

AJAX XMLHTTP REQUEST

HTTP RESPONSE

(Only data)

WEB BROWSER

WEB SERVER

WHAT IS ASYNCHRONOUS PROGRAMMING?

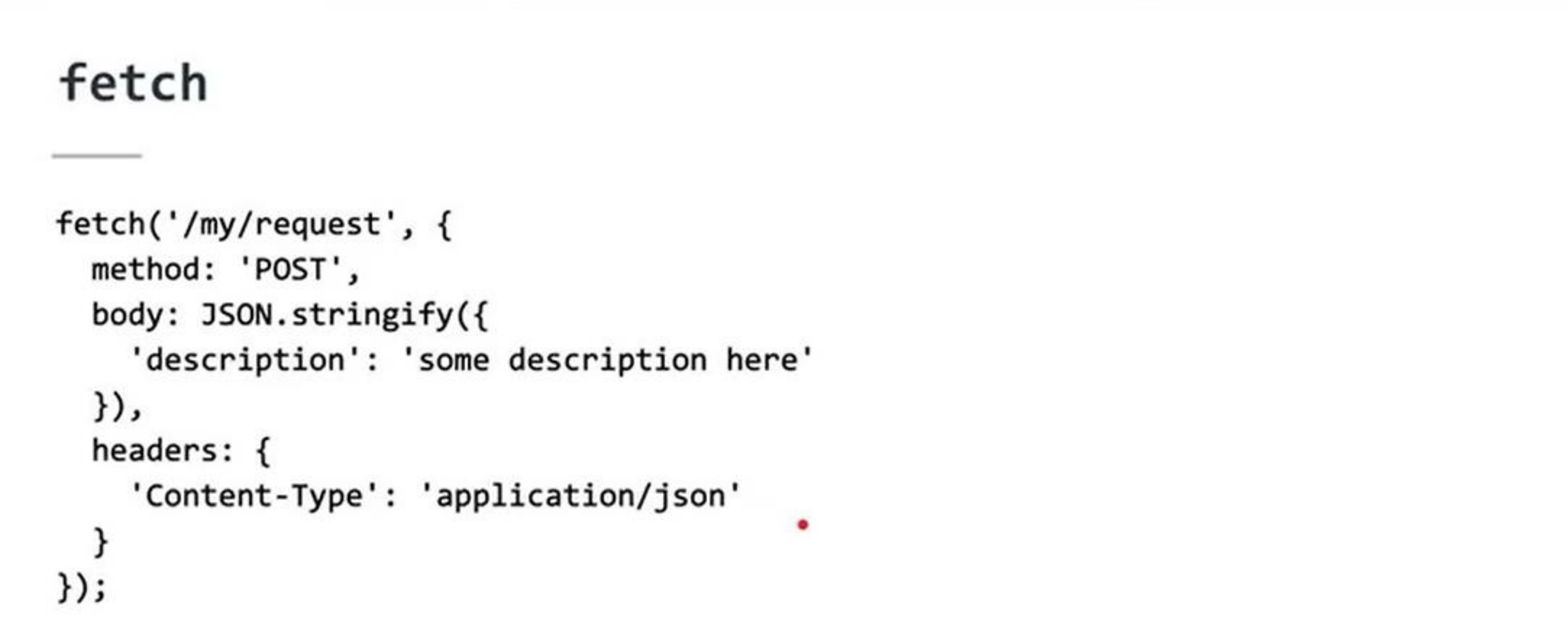
- AN ASYNCHRONOUS PROGRAMMING MAKES IT POSSIBLE EXPRESS WAITING FO LONG -RUNNING APPLICATIONS WITHOUT FREEZING THE PROGRAM DURING THESE ACTIONS.

- X = 2, THE VALUE OF X IS KNOWN
- X = VALUE FROM SERVER ,YOU HAVE TO WAIT SERVER RESPONSE

WHY USE AJAX?

- TO BUILD BETTER , FASTER AND INTERACTIVE WEB APPLICATIONS

APPLICATION



fetch

```
fetch('/my/request', {
  method: 'POST',
  body: JSON.stringify({
    'description': 'some description here'
  }),
  headers: {
    'Content-Type': 'application/json'
  }
});
```

PRACTICAL APPLICATION

DEMO APPLICATION

- GETTING USER DATA
- USING AJAX TO SEND DATA TO FLASK
- USING SESSIONS IN CONTROLLERS
- MODELING RELATIONSHIPS

04 WHAT NEXT

WHAT NEXT

- API INTRDUCTION
- HTTPS AND FLASK BASIC
- ENDPOINT & PAYLOADS
- API TESTING
- API DOCUMENTATION

QUESTIONS

FEEDBACK!

THIS IS WHERE YOU CAN TELL ME WHAT NEEDS IMPROVEMENT

THANK YOU