CRUD Review

Operations with SQLAlchemy

In this lesson, we performed all of our CRUD operations with SQLAlchemy on an SQLite database. Before we perform any operations, we must first import the necessary libraries, connect to our restaurantMenu.db, and create a session to interface with the database:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from database_setup import Base, Restaurant, MenuItem

engine = create_engine('sqlite:///restaurantMenu.db')
Base.metadata.bind=engine
DBSession = sessionmaker(bind = engine)
session = DBSession()
```

CREATE

We created a new Restaurant and called it Pizza Palace:

```
myFirstRestaurant = Restaurant(name = "Pizza Palace")
session.add(myFirstRestaurant)
sesssion.commit()
```

We created a cheese pizza menu item and added it to the Pizza Palace Menu:

```
cheesepizza = menuItem(name="Cheese Pizza", description = "Made with all natural
ingredients and fresh mozzarella", course="Entree", price="$8.99", restaurant=myF
irstRestaurant)
```


READ We read out information in our database using the query method in SQLAlchemy:

```
firstResult = session.query(Restaurant).first()
firstResult.name

items = session.query(MenuItem).all()
for item in items:
    print item.name
```

UPDATE

In order to update and existing entry in our database, we must execute the following commands:

1. Find Entry
2. Reset value(s)
3. Add to session
4. Execute session.commit()

We found the veggie burger that belonged to the Urban Burger restaurant by executing the following query:

```
veggieBurgers = session.query(MenuItem).filter_by(name= 'Veggie Burger')
for veggieBurger in veggieBurgers:
    print veggieBurger.id
    print veggieBurger.price
    print veggieBurger.restaurant.name
    print "\n"
```

Then we updated the price of the veggie burger to $2.99:

```
UrbanVeggieBurger = session.query(MenuItem).filter_by(id=8).one()
UrbanVeggieBurger.price = '$2.99'
session.add(UrbanVeggieBurger)
session.commit()
```

DELETE

To delete an item from our database we must follow the following steps:

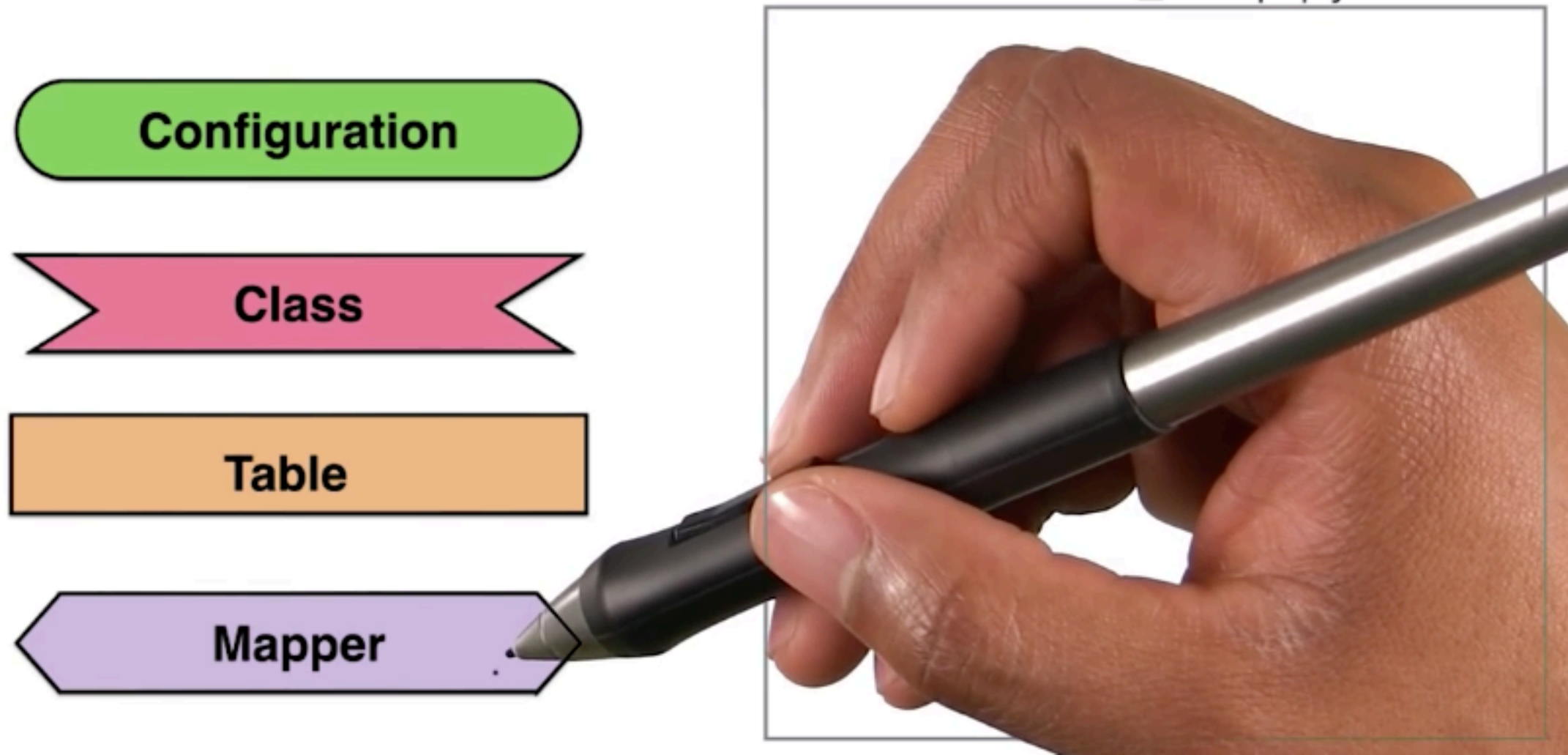
1. Find the entry
2. Session.delete(Entry)
3. Session.commit()

We deleted spinach Ice Cream from our Menu Items database with the following operations:

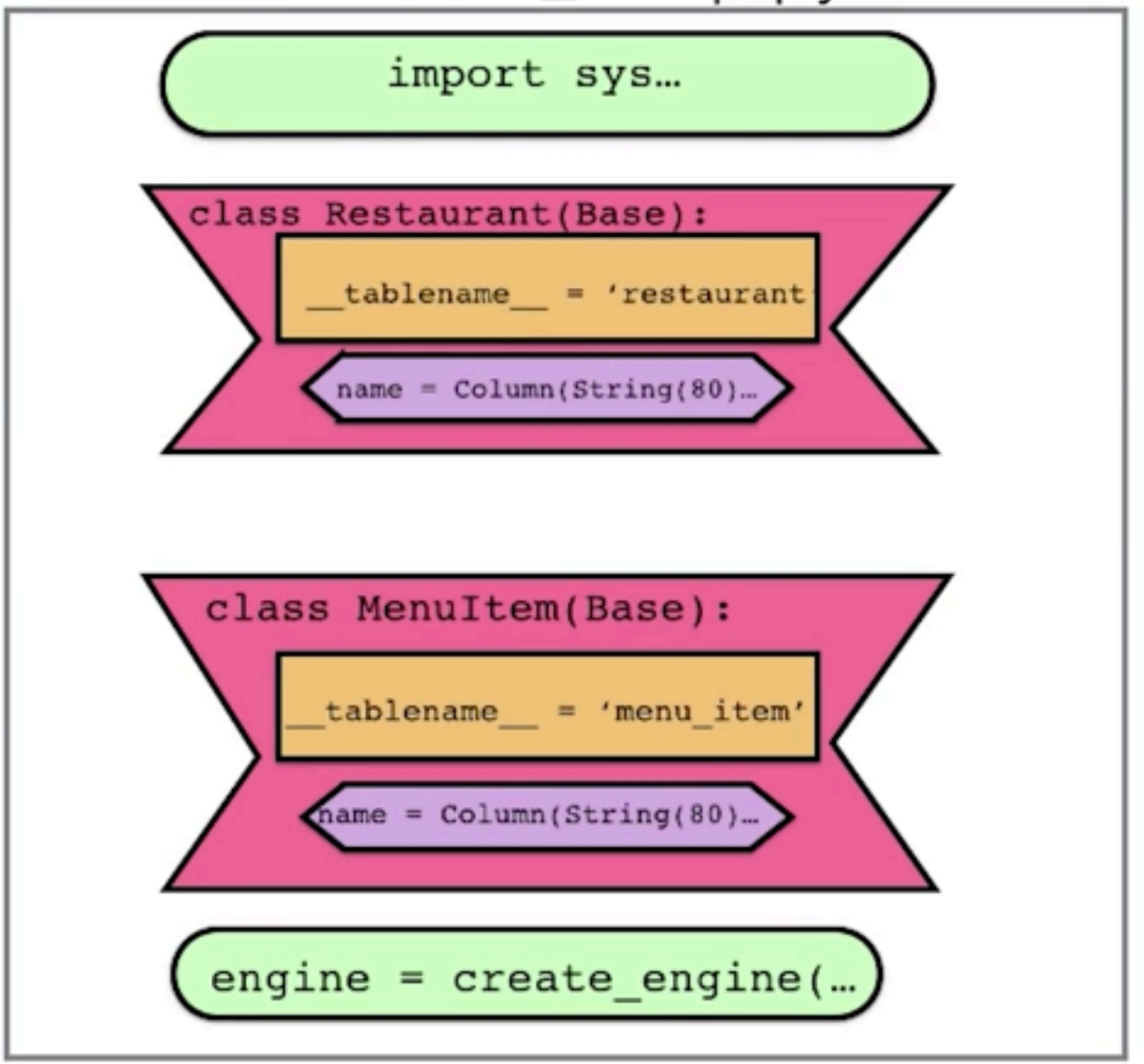
```
spinach = session.query(MenuItem).filter_by(name = 'Spinach Ice Cream').one()
session.delete(spinach)
session.commit()
```

Creating a database with SQLAlchemy

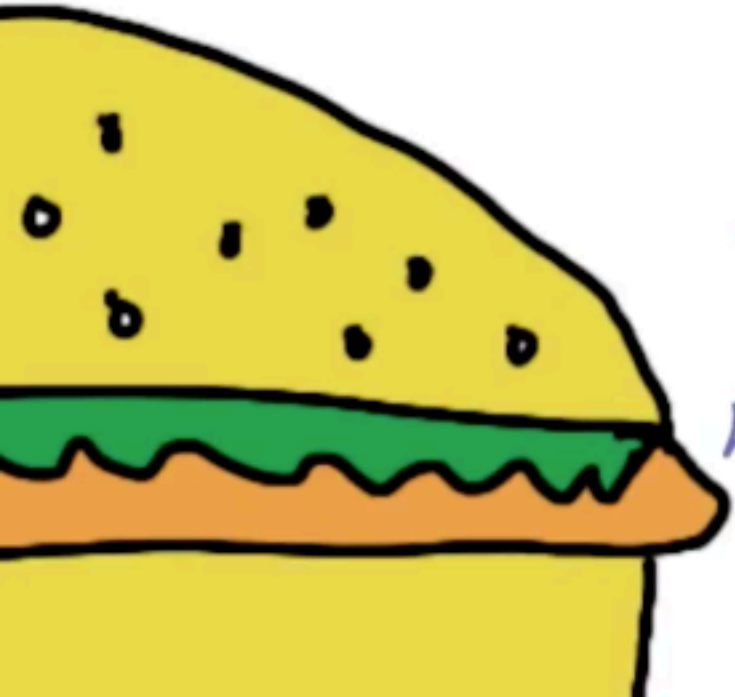
database_setup.py



database_setup.py



Update



1. Find entry
2. Reset values
3. add to session
4. session.commit()

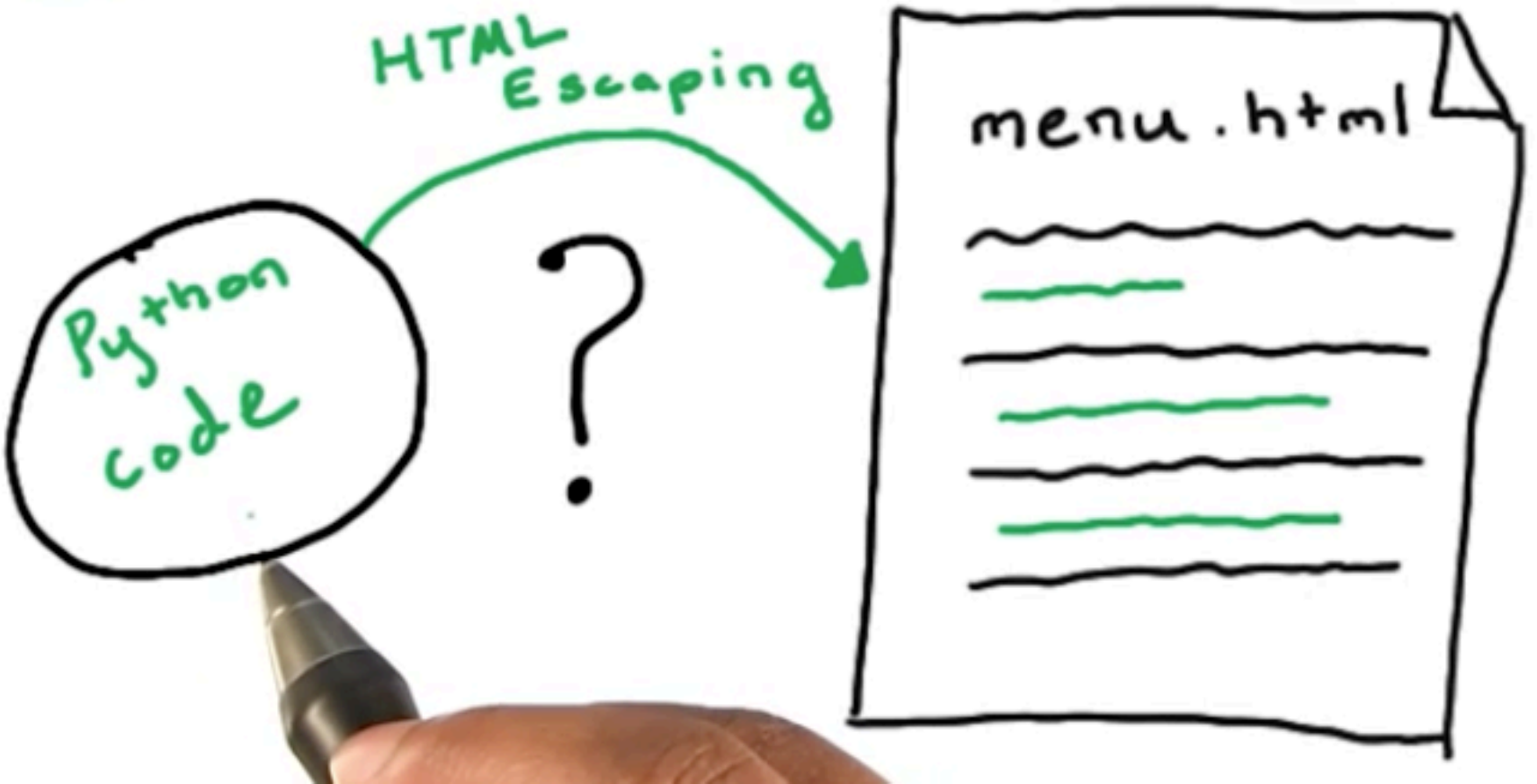
Rendering Templates

render_template(____,____)

templateName.html

Variable = key

Rendering Templates



Rendering Templates

{% logical code %}

{{ printed code }}

URL Building

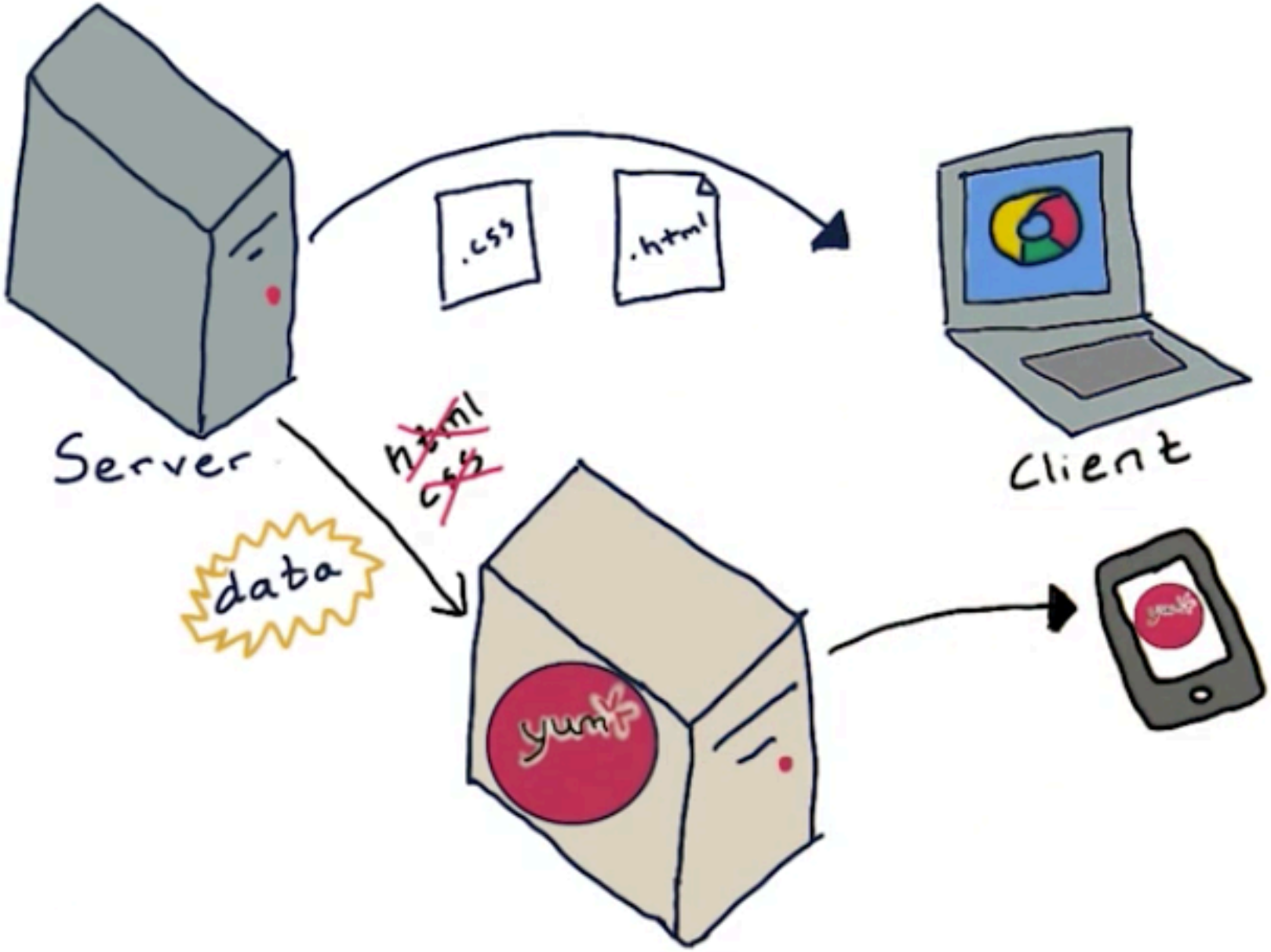
url_for( )

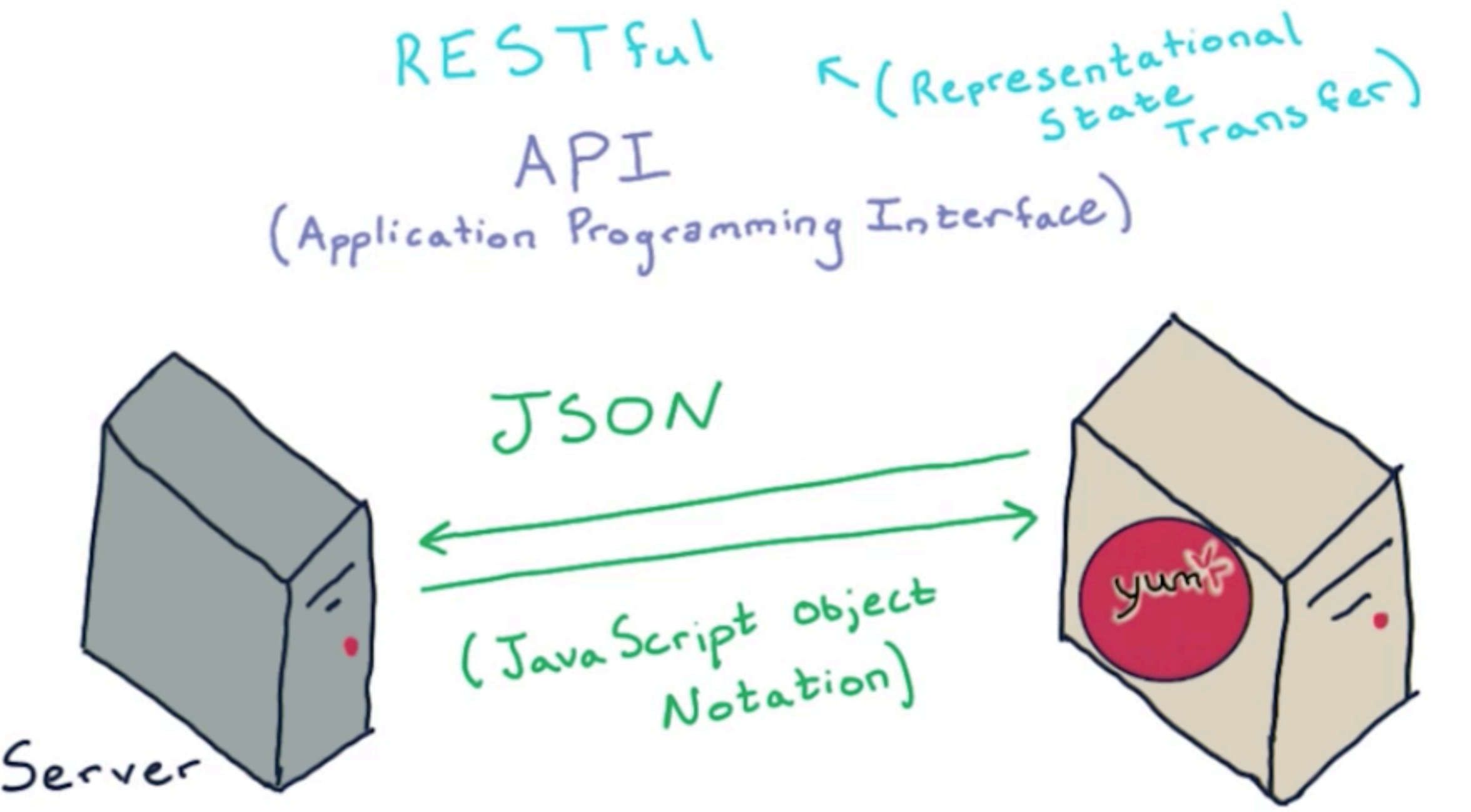
http://flask.pocoo.org/docs/0.10/quickstart/#url-building

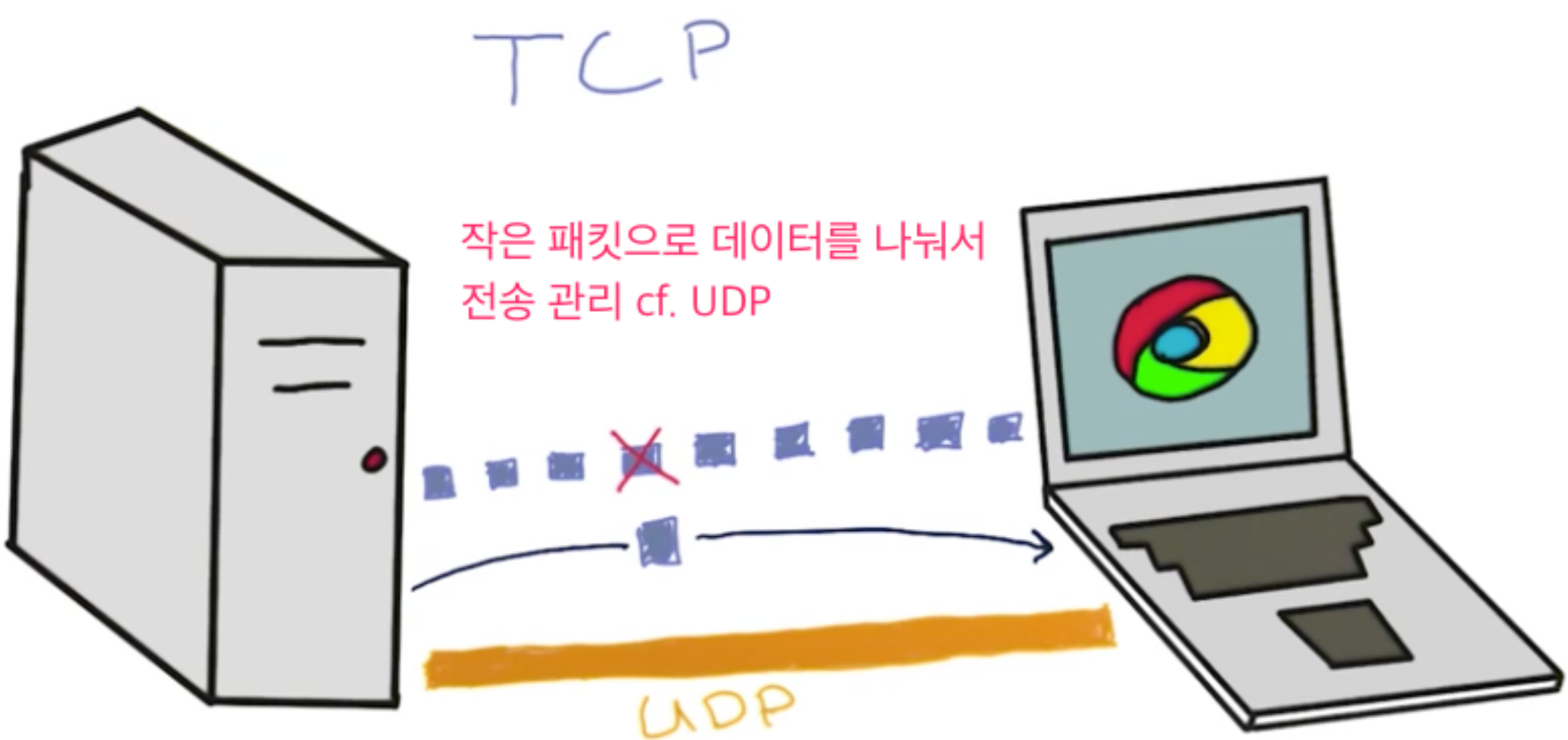
— sessions —

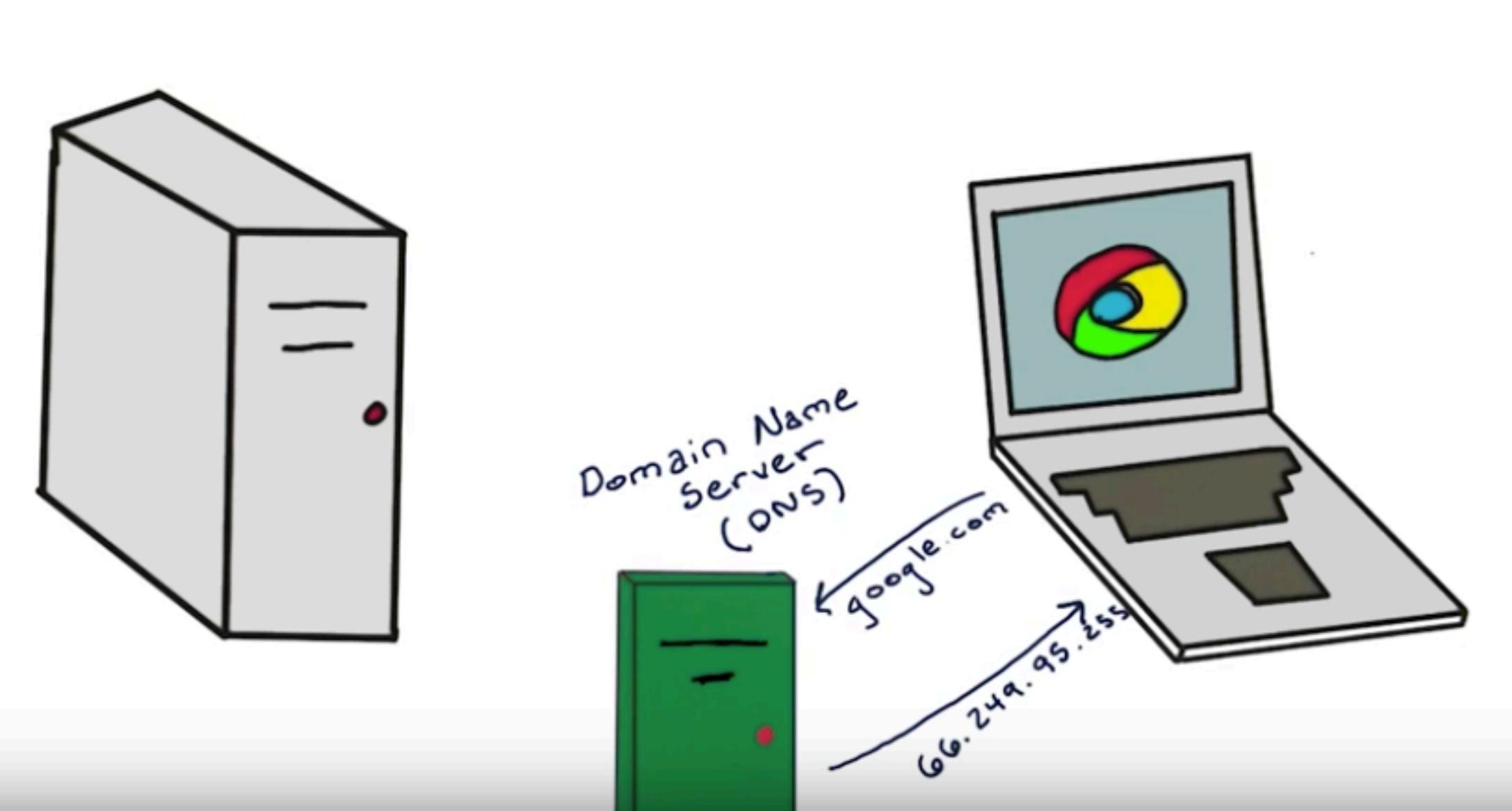
```
flash("insert message here")
```

```
get_flashed_messages()
```

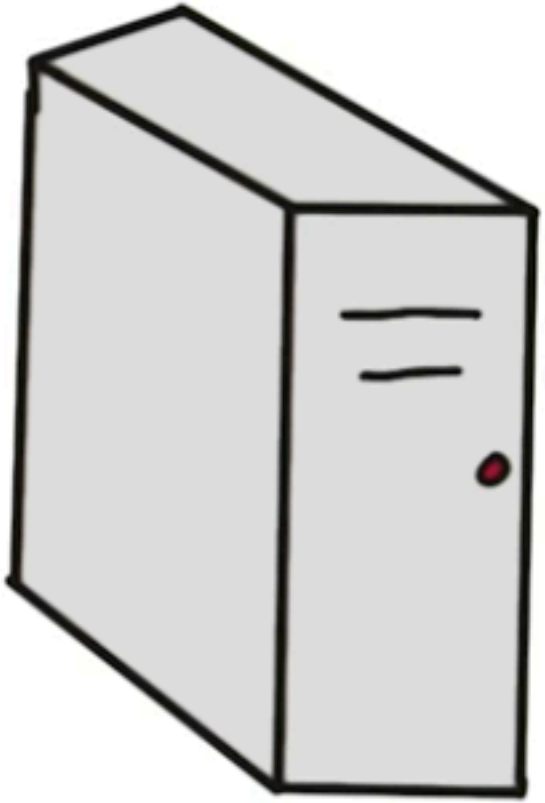

.css
.html
Server
Client
data
yum


RESTful
API
(Application Programming Interface)
(Representational State Transfer)
JSON
(JavaScript object Notation)
Server
yum


TCP
작은 패킷으로 데이터를 나눠서
전송 관리 cf. UDP
UDP


Domain Name
Server
(DNS)
google.com
66.249.95.255

IP



Ports

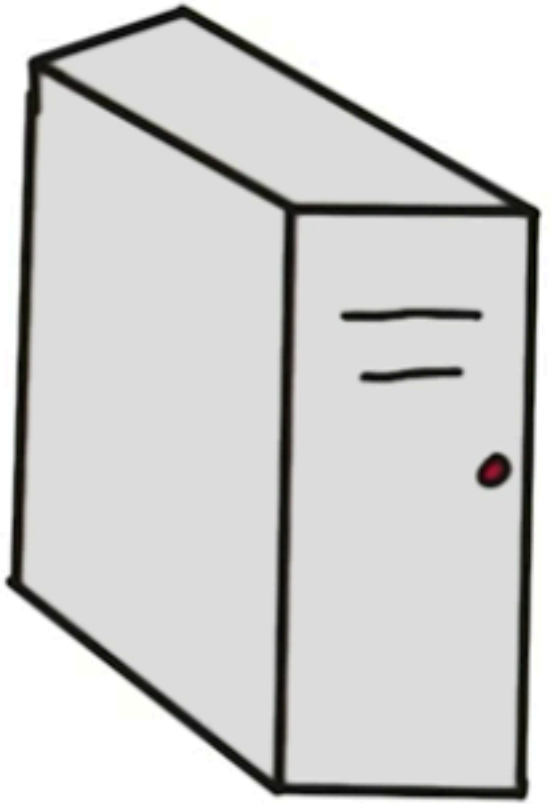
0 - 65,536

0 - 10,000

66.249.95.255:8080

217.99.202

IP



Ports

0 - 65,536

0 - 10,000

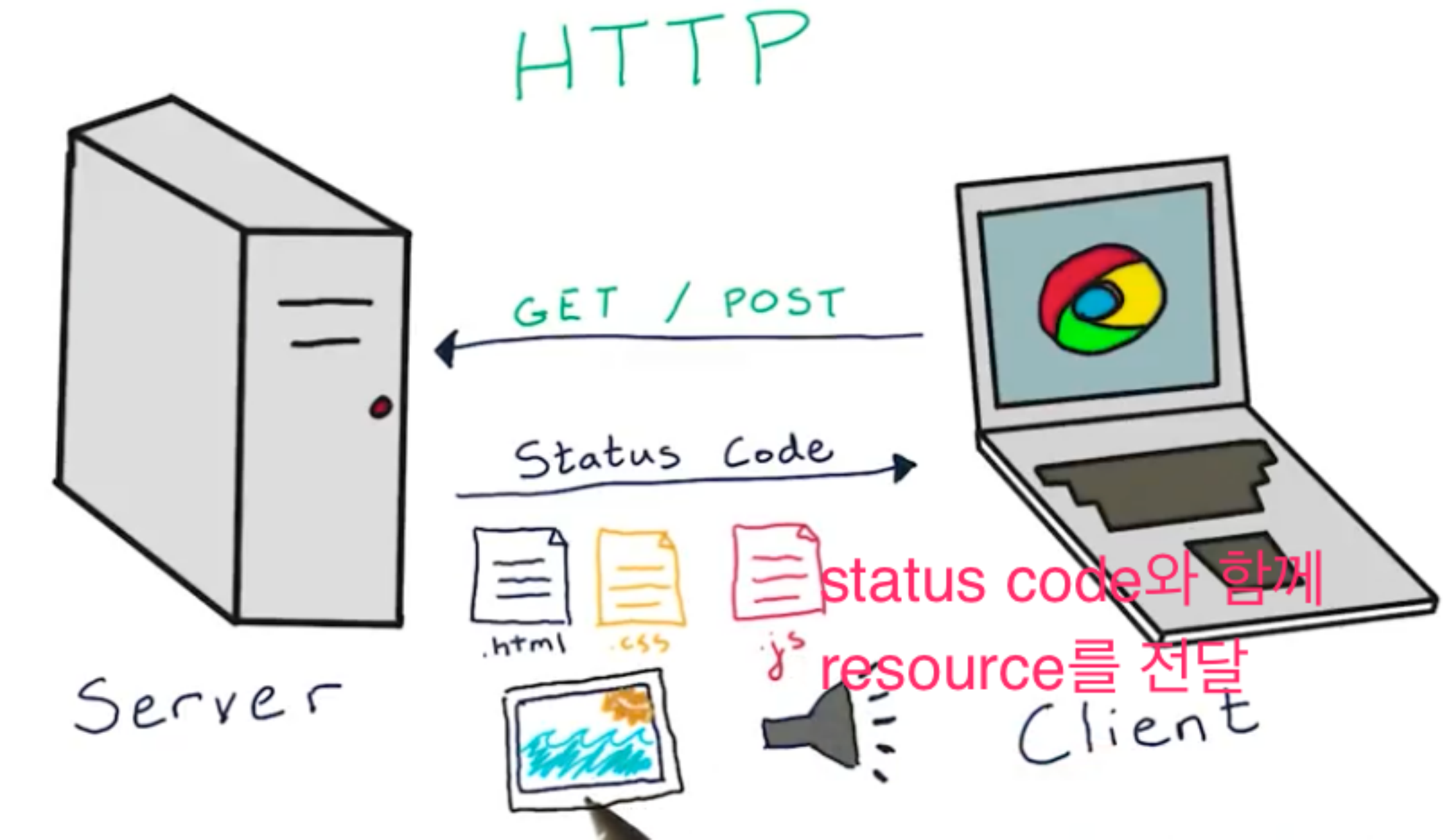
80

66.249.95.255:8080

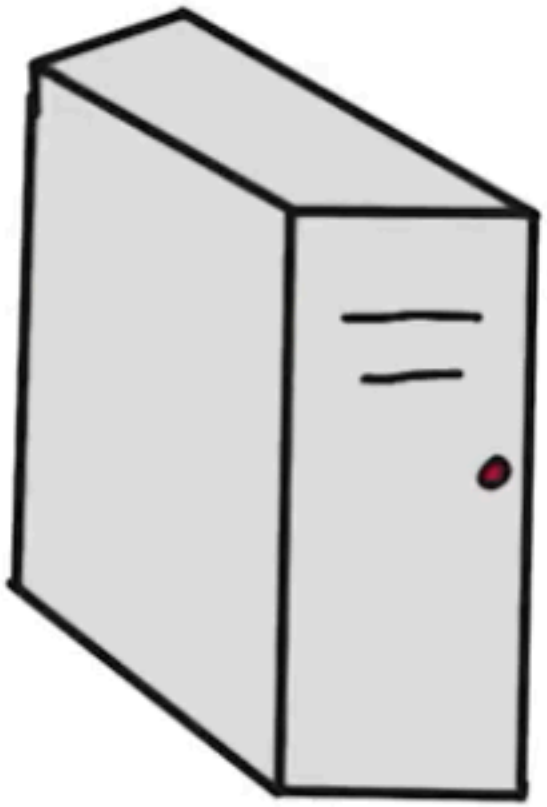
67.217.99.202

IP

localhost
127.0.0.1


HTTP
GET / POST
Status Code
.html
.css
.js
Server
Client
status code와 함께
resource를 전달

HTTP



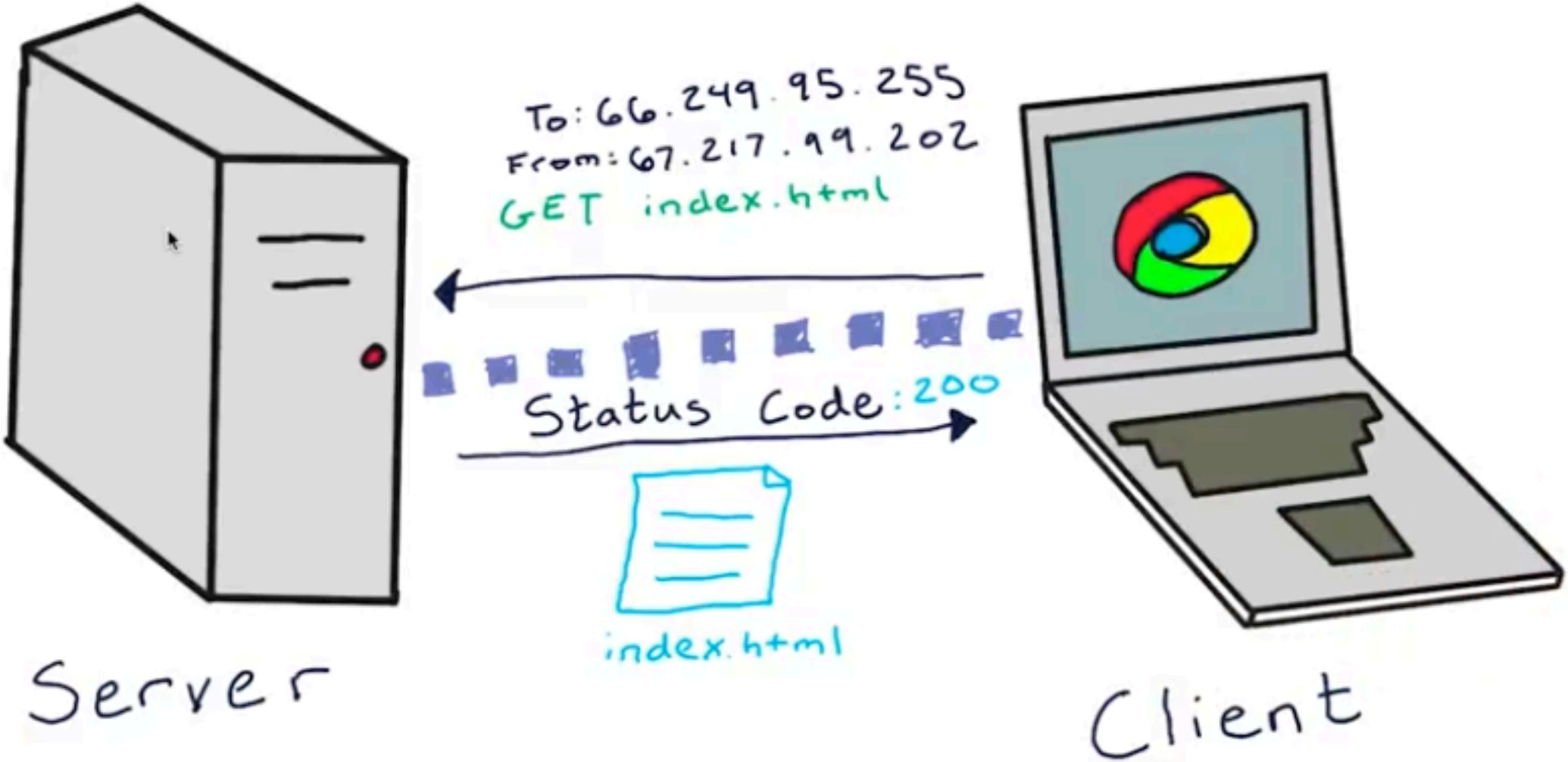
Server

GET / POST

Status Code

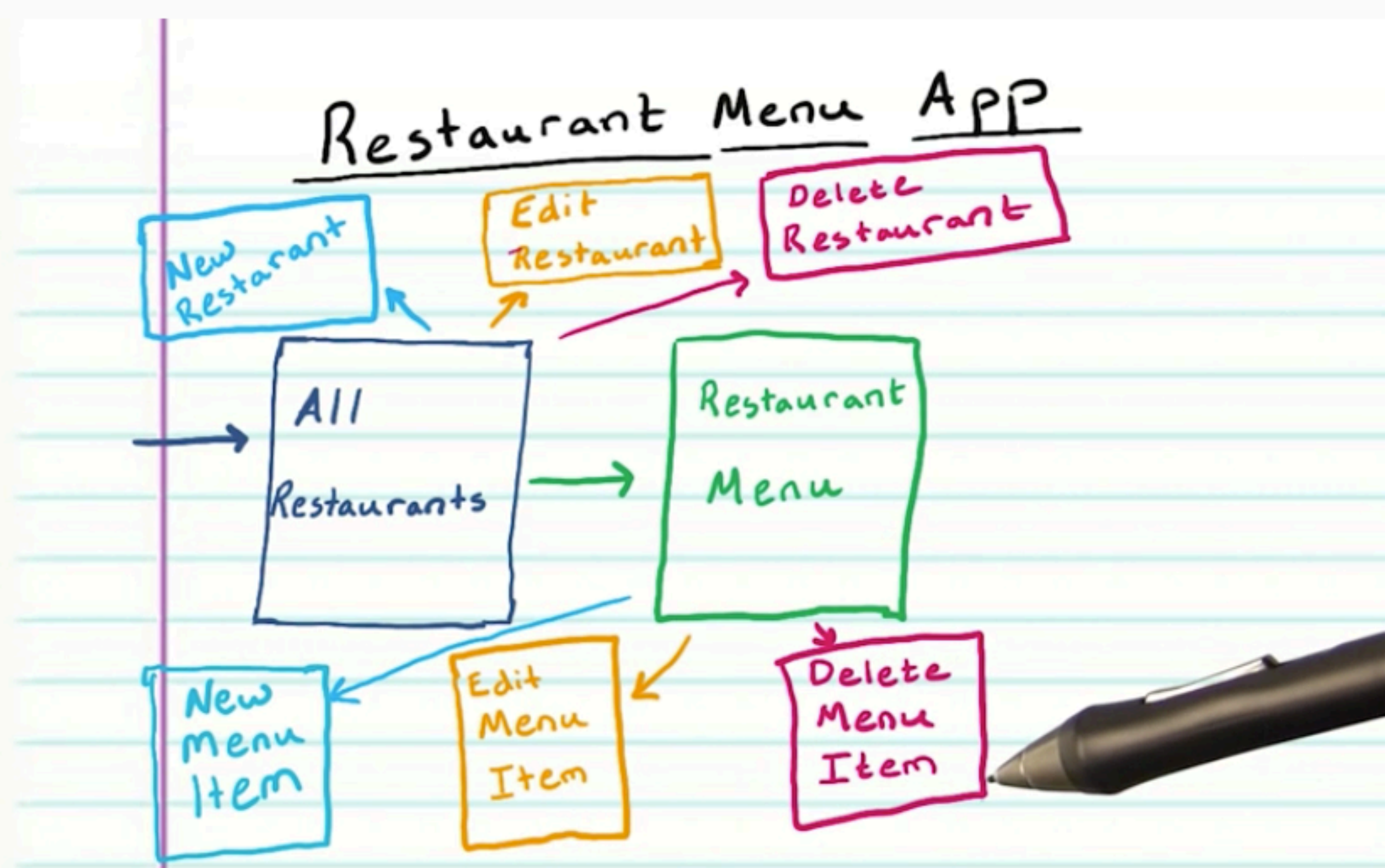
200 : Successful GET
301 : Successful POST
404 : File not found

Client


To: 66.249.95.255
From: 67.217.99.202
GET index.html
Status Code: 200
index.html
Server
Client

Checklist

- [ ] Mock-ups
- [ ] Routing
- [ ] Templates & Forms
- [ ] CRUD Functionality
- [ ] API Endpoints
- [ ] Styling & Message Flashing


Restaurant Menu App
New Restaurant
Edit Restaurant
Delete Restaurant
All Restaurants
Restaurant Menu
New Menu Item
Edit Menu Item
Delete Menu Item

Show all restaurants

/restaurants (and '/')

Show a restaurant menu

/restaurant/<int:restaurant_id/menu

(and /restaurant/<int:restaurant_id>)

Create a new restaurant

/restaurant/new

Create a new menu item

restaurant/<int:restaurant_id>/menu/new

Edit a restaurant

restaurant/<int:restaurant_id>/edit

Edit a menu item

/restaurant/<int:
menu/<int:menu_id>/edit

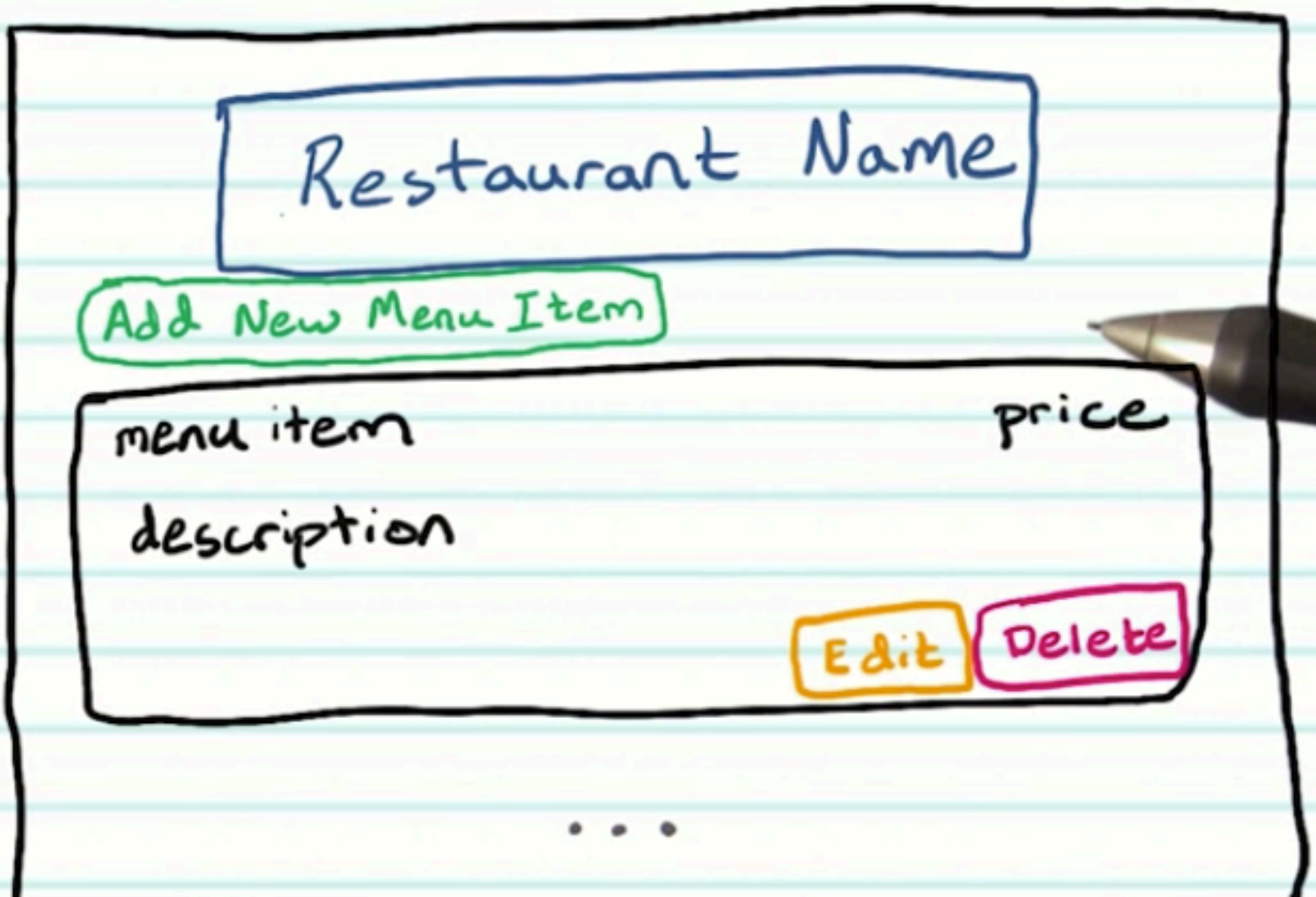
Delete a restaurant

/restaurant/<int:restaurant_id>/delete

Delete a menu item

/restaurant/<int:restaurant_id>/
menu/<int:menu_id>/delete

restaurant/<int: restaurant_id>/ menu



Show all restaurants

/restaurants (and '/')

Show a restaurant menu

/restaurant/<int:restaurant_id>/menu

(and /restaurant/<int:restaurant_id>)

Create a new restaurant

/restaurant/new

Create a new menu item

restaurant/<int:restaurant_id>/menu/new

Edit a restaurant

restaurant/<int:restaurant_id>/edit

Edit a menu item

/restaurant/<int:restaurant_id>/
menu/<int:menu_id>/edit

Delete a restaurant

/restaurant/<int:restaurant_id>/delete

Delete a menu item

/restaurant/<int:restaurant_id>/
menu/<int:menu_id>/delete

URL	Method	Message
/ /restaurants	showRestaurants()	"This page will show all my restaurants"
/restaurant/new	newRestaurant()	"This page will be for making a new restaurant"
/restaurant/restaurant_id/edit	editRestaurant()	"This page will be for editing restaurant %s" % restaurant_id
/restaurant/restaurant_id/delete	deleteRestaurant()	"This page will be for deleting restaurant %s" % restaurant_id
/restaurant/restaurant_id /restaurant/restaurant_id/menu	showMenu()	"This page is the menu for restaurant %s" restaurant_id
/restaurant/restaurant_id/menu/new	newMenuItem()	"This page is for menu item for restaurant_id
/restaurant/restaurant_id/menu/menu_id /edit	editMenuItem()	"This page is for editing menu item %s " menu_id
/restaurant/restaurant_id/menu/menu_id /delete	deleteMenuItem()	"This page is for deleting menu item %s" menu_id

Adding Templates

restaurants.html

newRestaurant.html

editRestaurant.html

deleteRestaurant.html

menu.html

newMenuItem.html

editMenuItem.html

deleteMenuItem.html

API Endpoints

/restaurants/JSON

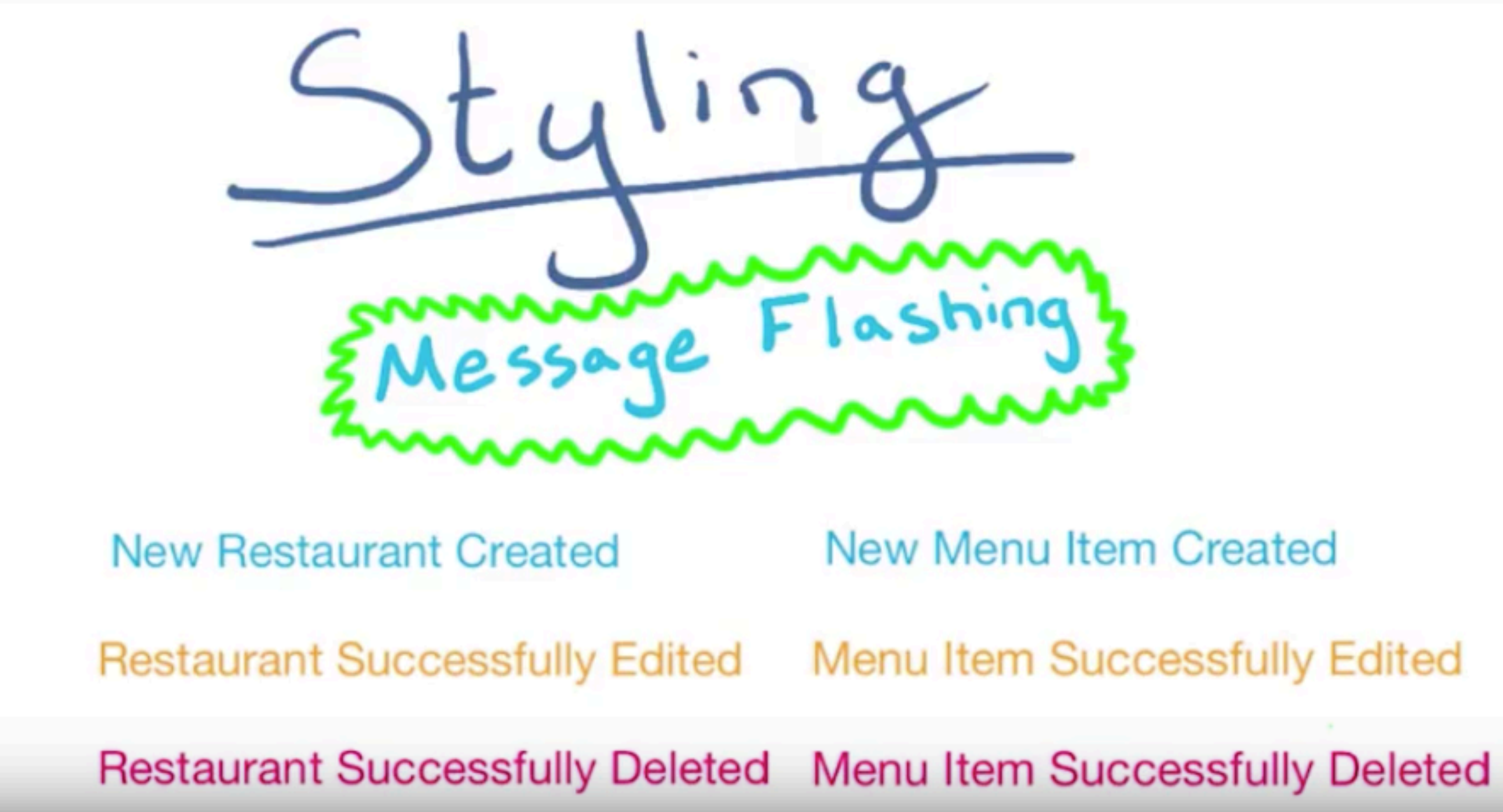
restaurants/restaurant_id/menu/JSON

restaurants/restaurant_id/menu/menu_id/JSON

API Endpoints

jsonify

serialize


Styling
Message Flashing
New Restaurant Created
New Menu Item Created
Restaurant Successfully Edited
Menu Item Successfully Edited
Restaurant Successfully Deleted
Menu Item Successfully Deleted

DB가 없는 상태면 fake DB를 만들어서 사용

```
3
4  #Fake Restaurants
5  restaurant = {'name': 'The CRUDdy Crab', 'id': '1'}
6
7  restaurants = [{'name': 'The CRUDdy Crab', 'id': '1'}, {'name
       :'Blue Burgers', 'id':'2'},{'name':'Taco Hut', 'id':'3'}]
8
9
10 #Fake Menu Items
11 items = [ {'name':'Cheese Pizza', 'description':'made with
       fresh cheese', 'price':'$5.99','course' :'Entree', 'id':'1
       '}, {'name':'Chocolate Cake','description':'made with
       Dutch Chocolate', 'price':'$3.99', 'course':'Dessert','id
       :'2'},{'name':'Caesar Salad', 'description':'with fresh
       organic vegetables','price':'$5.99', 'course':'Entree','id
```