Django

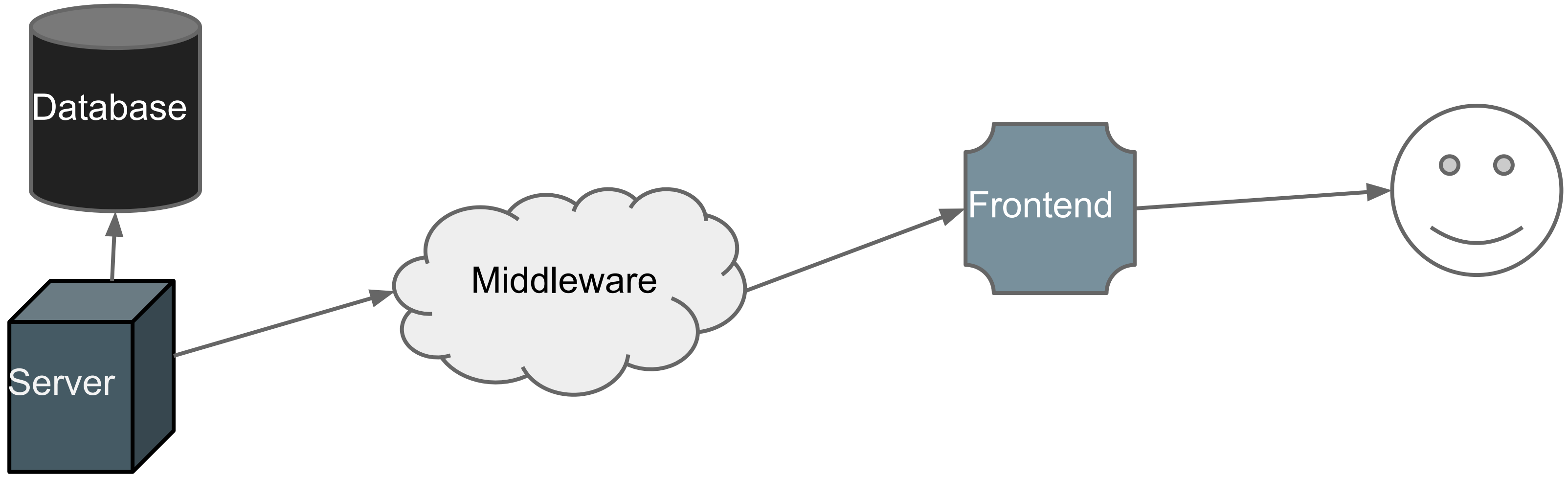
The last piece

Today's Agenda

- Finish JS assignment
- Do pass of reviews
- Server Lecture
- Everyone get Django

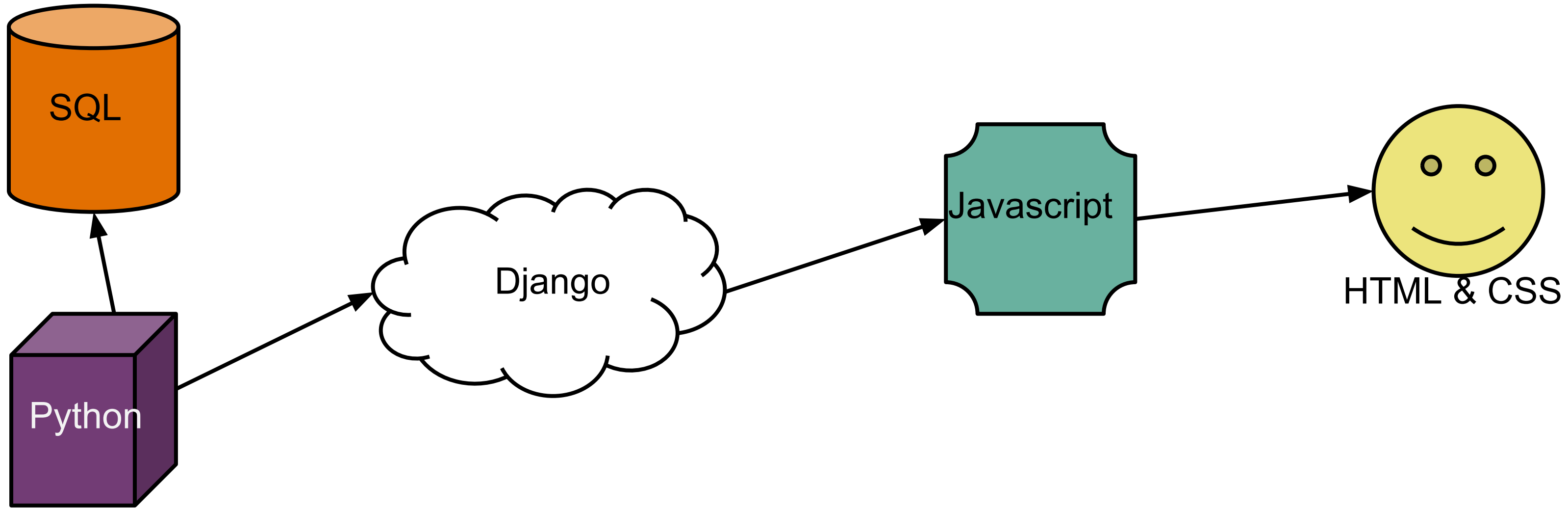
The Stack

Develop a web application from front to back, with all the sticky bits in the middle as well.



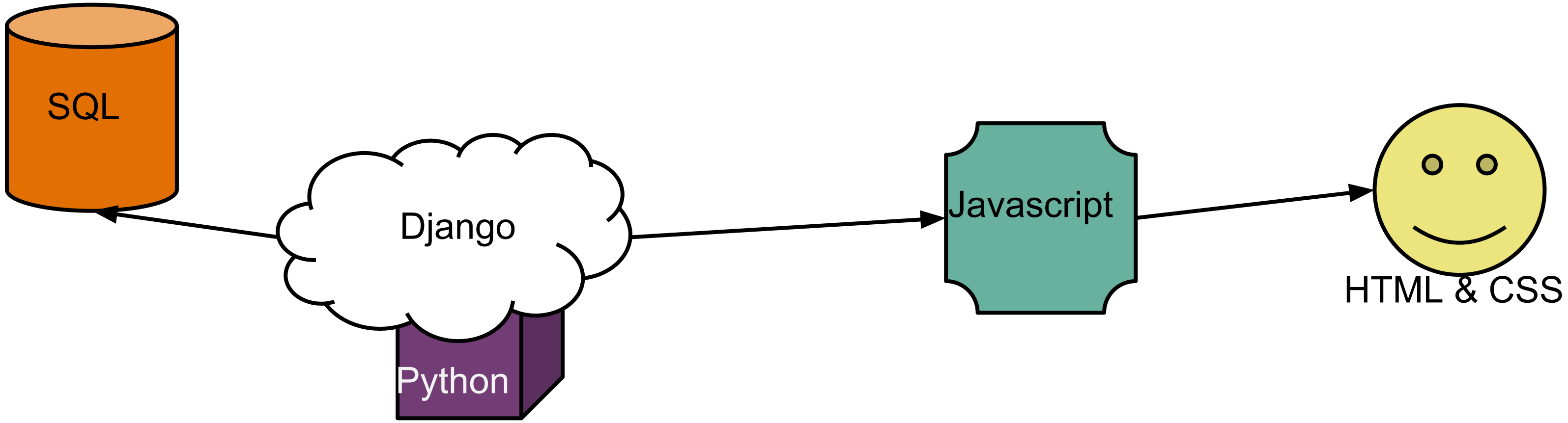
The Stack with languages

Each layer is best suited for a different language. This is what we'll be using:



What is the cloud?

Spoilers: It’s just someone else’s computer.



What Does the Server-Side and Client-side Mean?

The Server:

- Responsible for **serving** up the pages to people who access your url

The Client:

- the part that *requests* pages and data from the **server**
- this is generally the web browser

What Does the Server-Side and Client-side Mean?

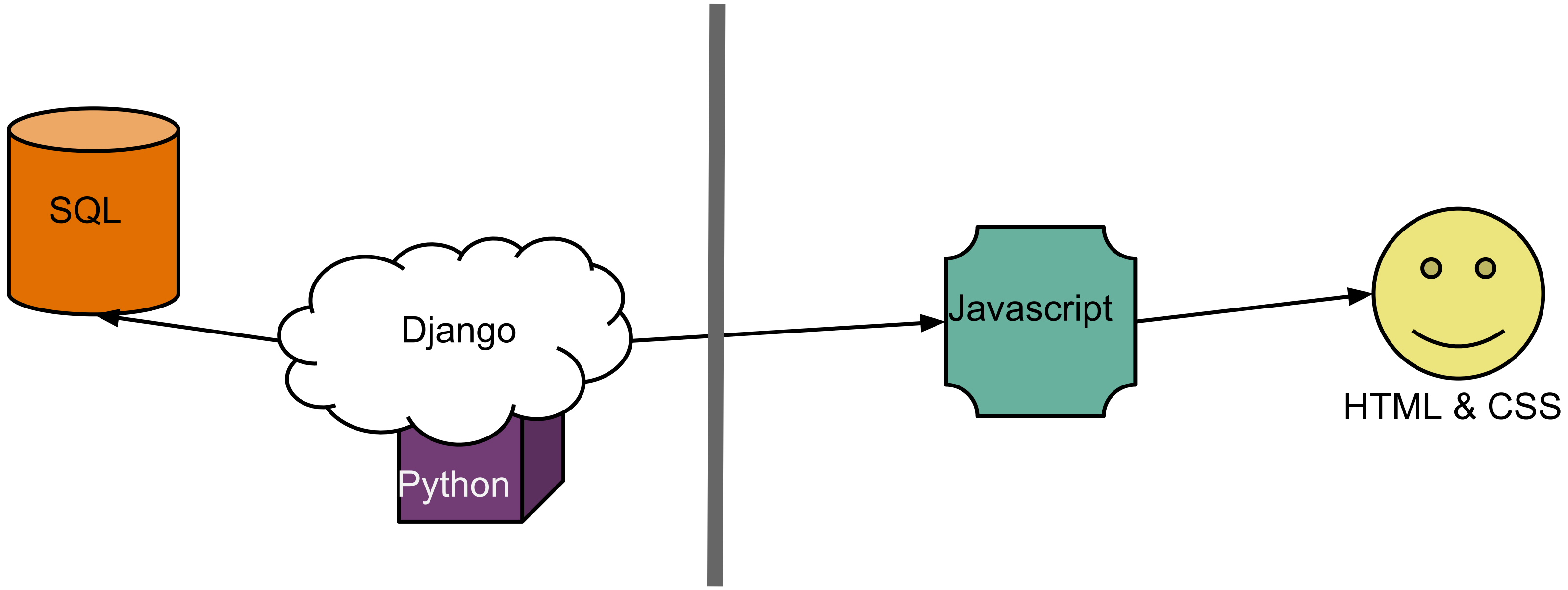
Server-side:

- code and logic that resides, runs, and is processed by the **server**

Client-side:

- code and logic that is run and processed by the **client**

Server-side & Client-side



Server-Side Programming

Uses:

- process user input
- display the correct pages and data
- structure web applications
- interact with permanent storage (the database)

Example Languages:

- PHP, ASP.Net in C#, C++, Java, Node.js, Python, Rails

Our language: Python, through the Django Framework

What About the Database?

A Database is where non-static data is stored for the server to access, process, and serve to the client.

Things you store in the database:

- any data you cannot store directly into your code

Data in a Database Examples

- usernames & password
- profile photo information
- forum posts (our spreadsheet is our database)
- emails
- tweets
- board games
- reddit
- songs
- movies
- t-shirt designs
- parking info
- game save files

Client-Side

Uses:

- makes the webpage interactive
- responds dynamically to users
- interacts with temporary storage (cookies, localStorage)
- requests information from the server
- provides remote service for client-side apps

Example Languages:

- HTML*, CSS*, Javascript, Ruby, Actionscript, Python

Our language: Javascript, with HTML & CSS

A Typical Website Flow

1. The **User** opens his web browser (the **Client**).
2. The **User** browses to http://google.com.
3. The **Client** (on the behalf of the **User**), sends a request to http://google.com (the **Server**), for their home page.
4. The **Server** then acknowledges the request, and replies the client with some meta-data (called *headers*), followed by the page's source.
5. The **Client** then receives the page's source, and *renders* it into a human viewable website.
6. The **User** types `Stack Overflow` into the search bar, and presses `Enter`
7. The **Client** submits that data to the **Server**.
8. The **Server** processes that data, and replies with a page matching the search results.
9. The **Client**, once again, renders that page for the **User** to view.

our Language Stacks' Flow

1. User requests http://corgisRawesome.com
2. **Client** requests info from http://corgisRawesome.com's **server**
3. **Server** acknowledges the request, returns to the **client** the homepage for corgisRawesome
4. **Client** receives the page's source and renders it to a human-readable format, based on the page's HTML and CSS
5. **Client** runs the Javascript that triggers on the page's load event, and sends a request to the **server**
6. **Server** receives the request, processes it in Python, fetches from the database, and then returns the desired data to the **client**
7. **Client's** Javascript AJAX success call is triggered, and it injects the latest corgi news into the HTML

MVC on the Web

Views: HTML pages with Javascript to handle dynamic interaction

Controllers: Server that serves data and views to the client

Models: structures stored on the database

Django

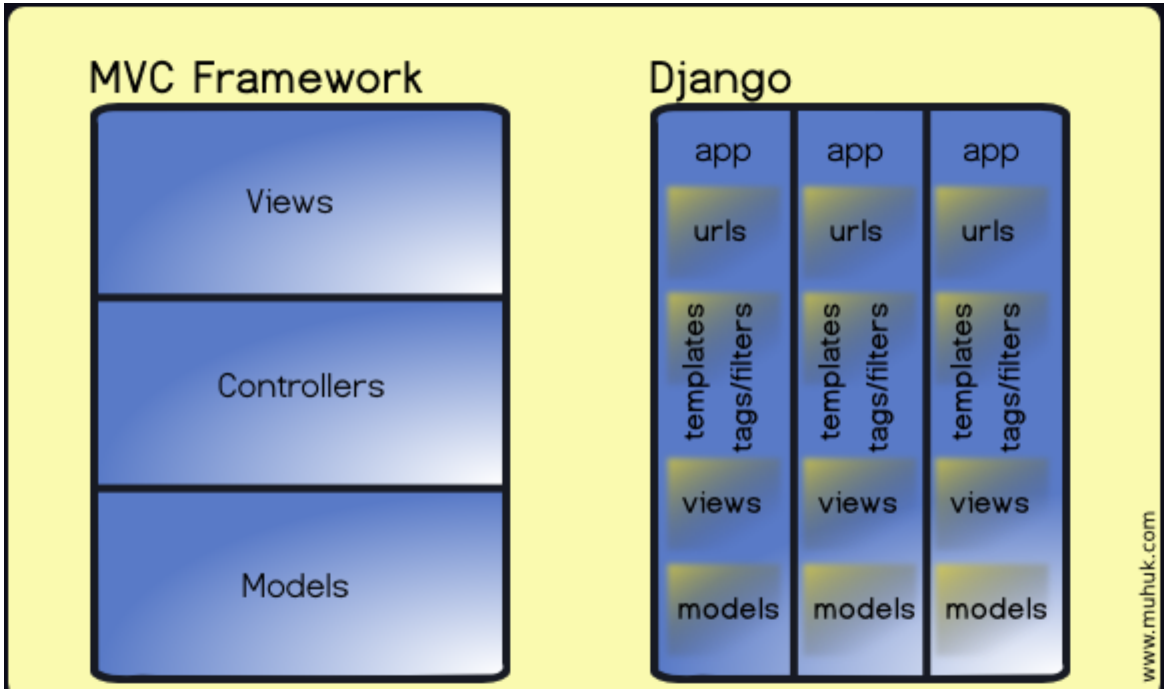
Django is a web framework, built in Python, that allows you to make a web server application quickly, with little fuss.

Official Website:

http://djangoproject.com

Django's Structure is MTV

Django uses a modified MVC framework to serve and process its' pages. It uses urls, templates, views, and models.


MVC Framework
Views
Controllers
Models
Django
app
urls
templates
tags/filters
views
models
app
urls
templates
tags/filters
views
models
app
urls
templates
tags/filters
views
models
www.muhuk.com