ВЕБ-ПРИЛОЖЕНИЕ

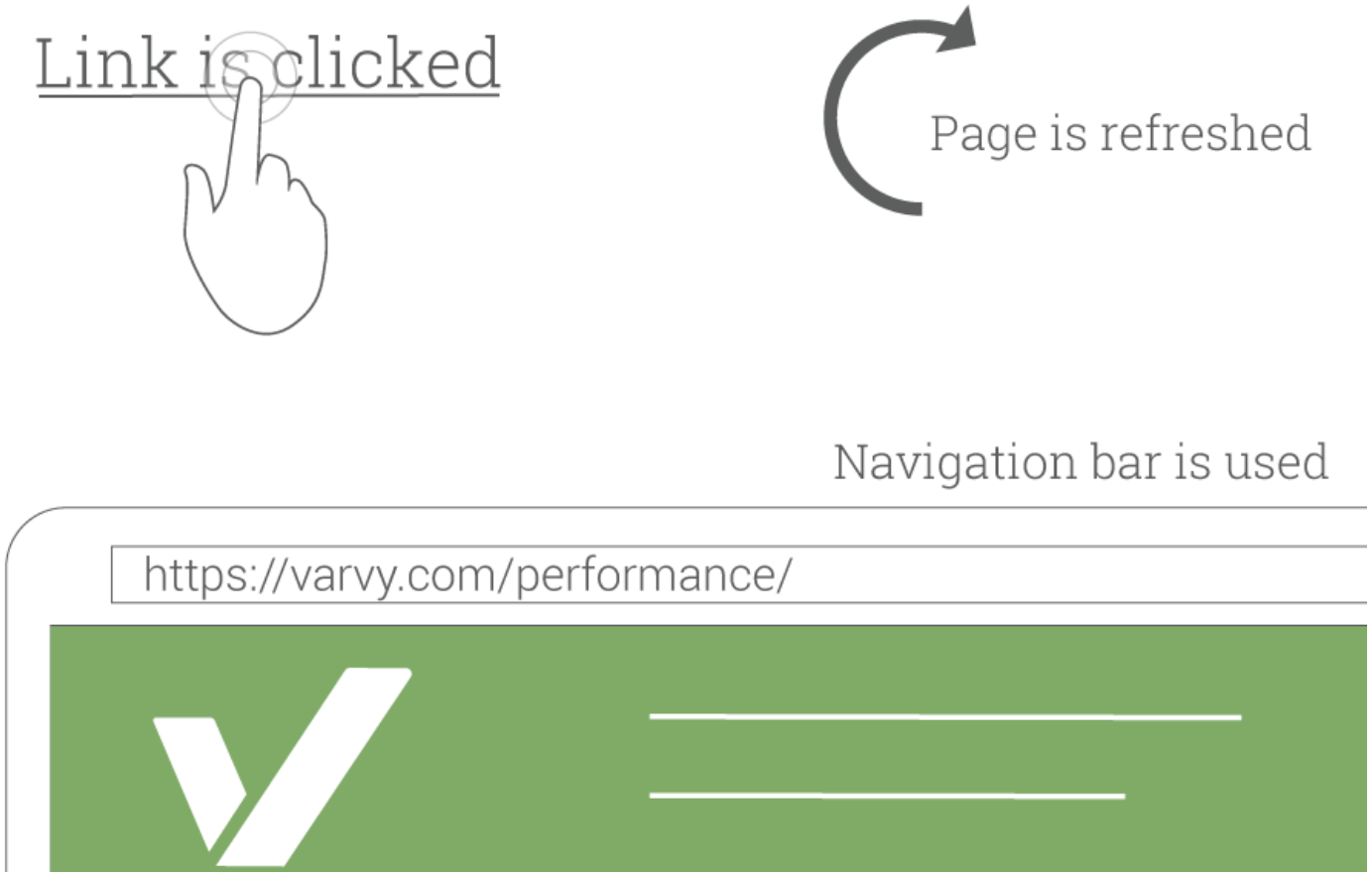
Содержание

- Основы Веб

- Веб-приложение с использованием Фреймворк веб
  - Веб-приложение с использованием Flask

Шаги отображаемой веб-страницы

1. Запрос выполняется при нажатии ссылки, обновляется страница или когда URL-адрес введен в браузер.

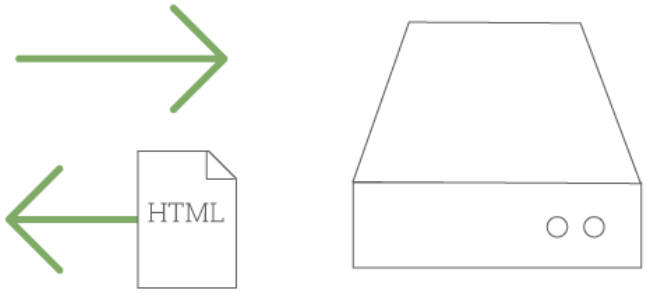


Шаги отображаемой веб-страницы

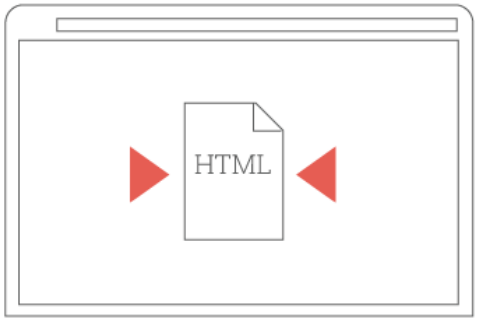
2. Загружается страница и ее ресурсы (файлы).

1. web browser requests document (html file)

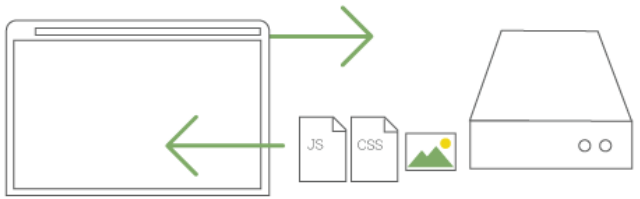
2. web server provides document (html file)



3. web browser reads html file to find resources

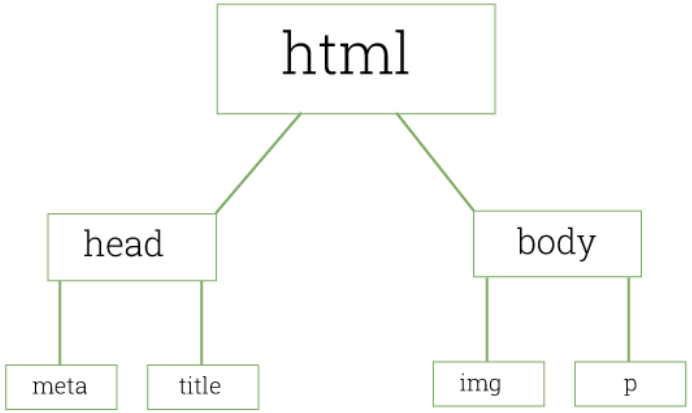


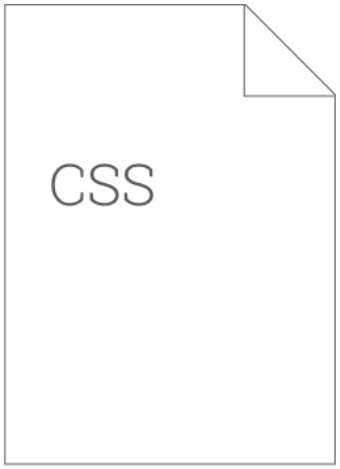
4. web browser requests found resources from server

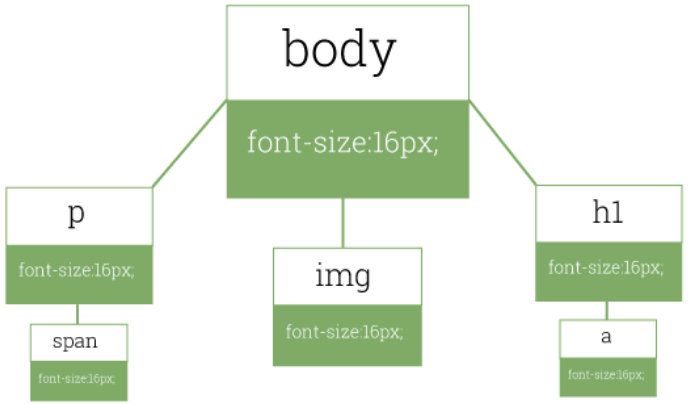


Шаги отображаемой веб-страницы

3. Веб-браузер использует ресурсы страницы для создания страницы.







Создание DOM

«Карта объектов докумен та» - это карта того, где в ещи отображаются на странице в соответствии с HTML.

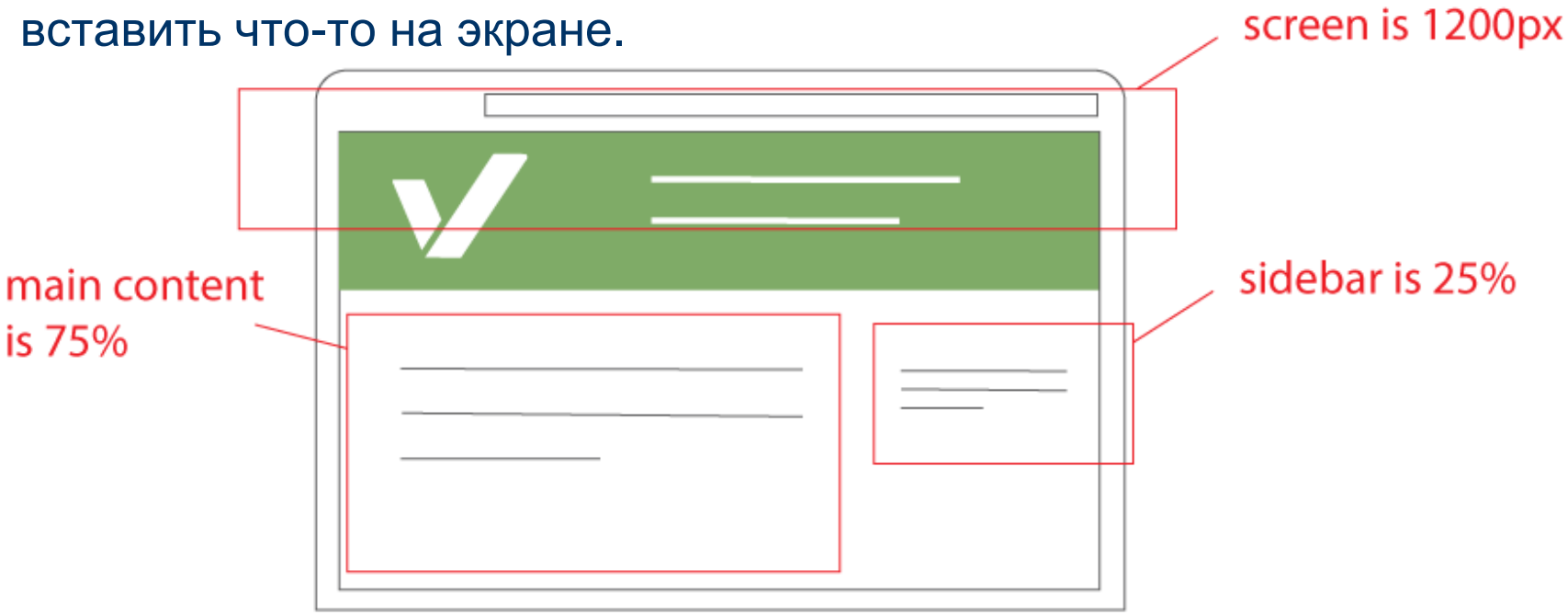
Содание CSSOM

"Карта объектов CSS" представляет собой карт у того, какие стили должн ы применяться к различн ым частям страницы в со ответствии с CSS.

Шаги отображаемой веб-страницы

4. Затем страница отображается (отображается) пользоват елю.

:После выполнения всех предыдущих шагов браузер может вставить что-то на экране.

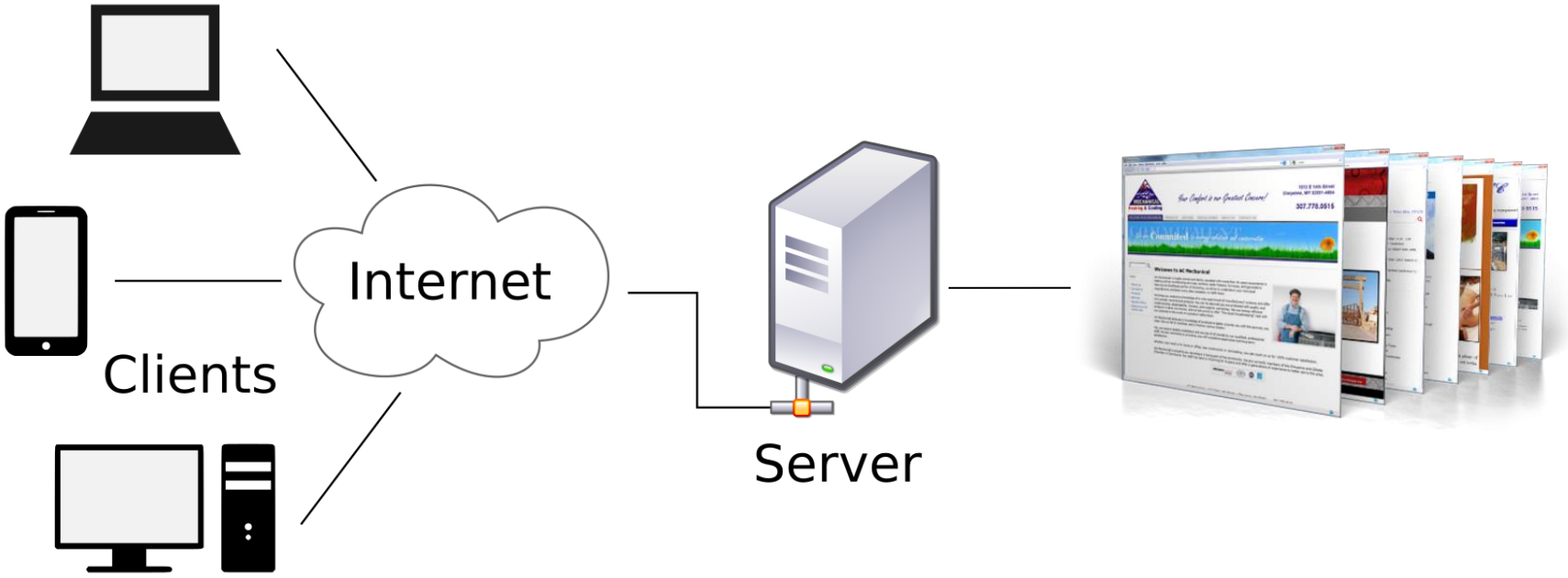


Макет / оглавление

Браузер определяет, насколько большой экран и как это повлияет на способ отображения страницы

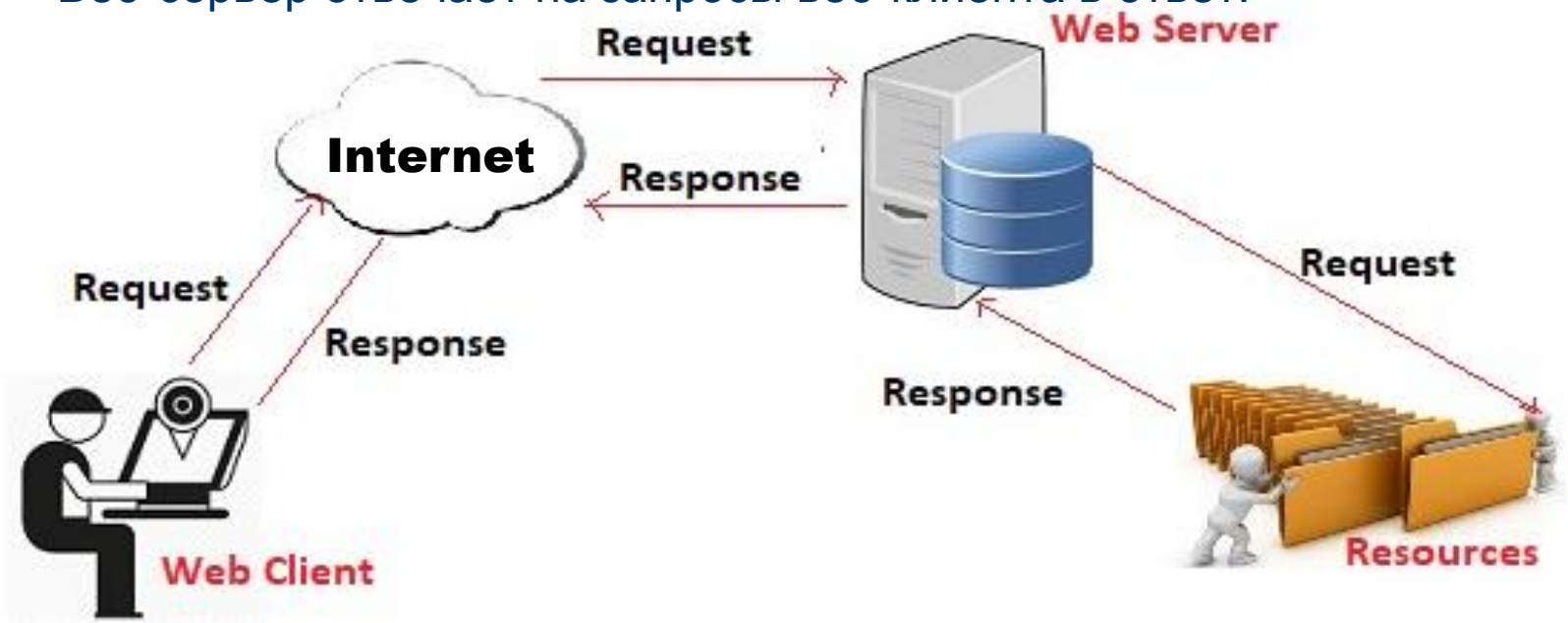
Веб-сервер

- Веб-сервер - это система, которая доставляет контент или услуги клиентам (пользователям) через Интернет.
  - Веб-сервер состоит из физического сервера, серверной операционной системы и программного обеспечения, используемого для облегчения связи по протоколу HTTP.
  - Веб-серверы используются для размещения веб-сайтов, игр, хранения данных или запуска бизнес-приложений.



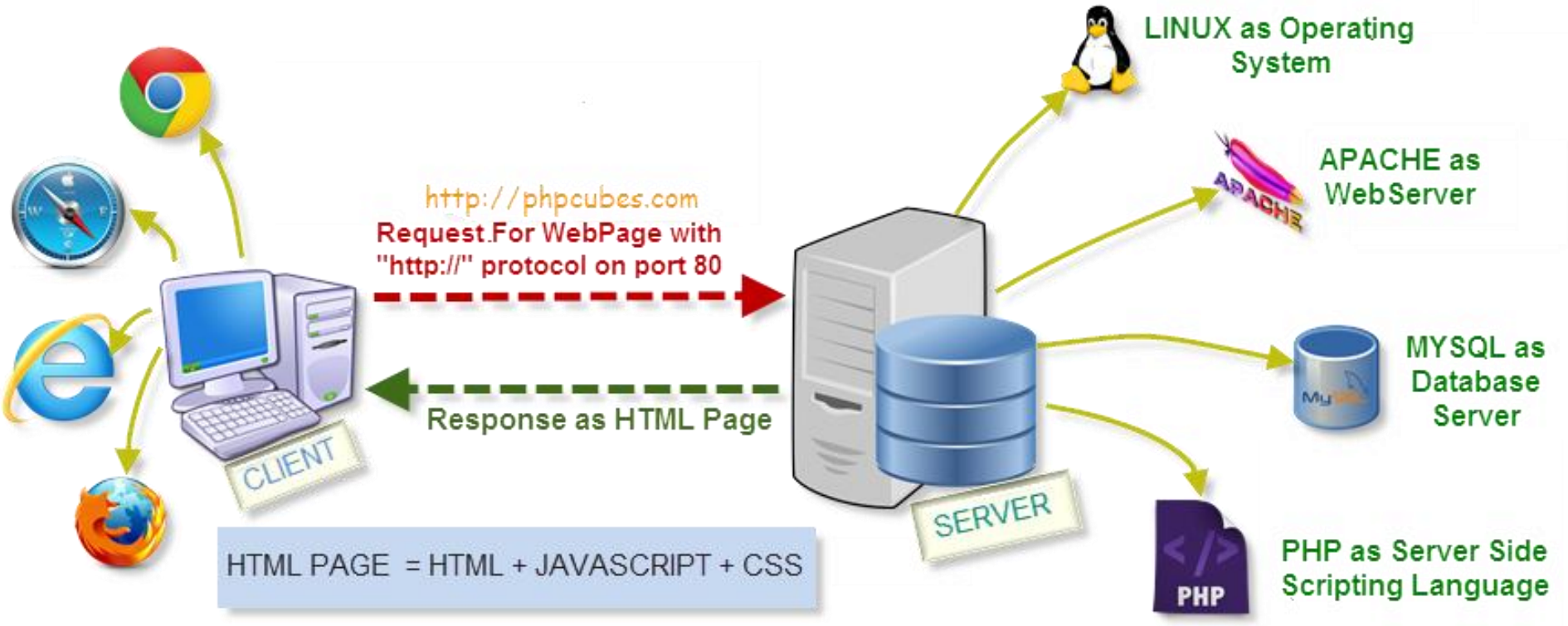
Веб клиент и сервер

- **Веб-клиент (пользователь, просматривающий веб-страницы в веб-браузере) и веб-сервер (компьютер) взаимодействуют при просмотре веб-сайта**
  - Веб-клиент получает доступ к веб-ресурсам с помощью веб-сервера.
  - Веб-сервер получает любой запрос от веб-клиента и находит запрашиваемые ресурсы
  - Веб-сервер отвечает на запросы веб-клиента в ответ.



Веб клиент и сервер

- Веб-сервер предоставляет веб-браузеру клиента соответствующий код HTML, CSS и JavaScript, необходимый для отображения интерактивной веб-страницы.



Две части веб-сервера

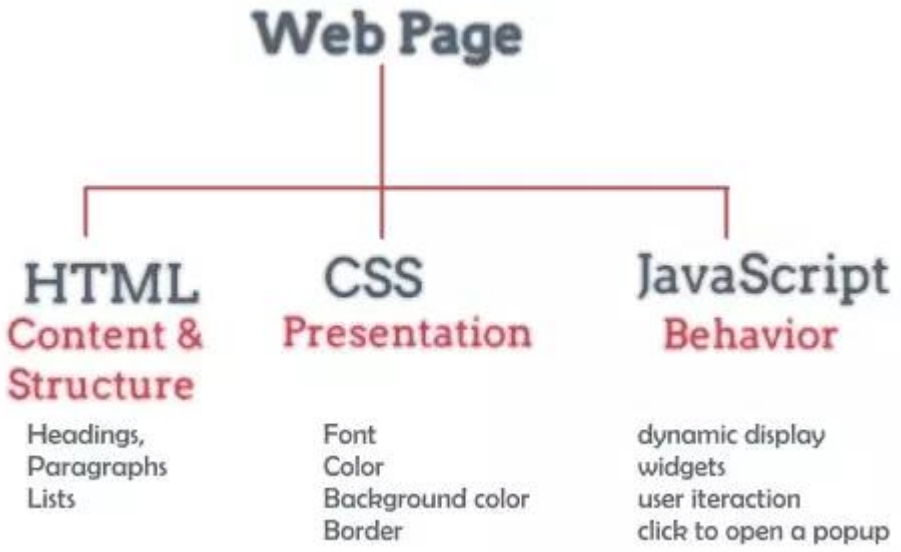
- На аппаратной стороне
  - компьютер, на котором хранятся программное обеспечение веб-сервера и файлы компонентов веб-сайта (например, документы HTML, изображения, таблицы стилей CSS и файлы JavaScript)
  - Он подключен к Интернету и поддерживает физический обмен данными с другими устройствами, подключенными к Интернету.
- Со стороны программного обеспечения
  - Контроль доступа веб-пользователей к размещенным файлам
  - Имеет как минимум HTTP-сервер

    : HTTP-сервер - это часть программного обеспечения, которая понимает URL-адреса (веб-адреса) и HTTP (протокол, используемый вашим браузером для просмотра веб-страниц)
  - Доступ к ним возможен через доменные имена (например, mozilla.org) веб-сайтов и доставляет их содержимое на устройство конечного пользователя.

Что делает веб сервер

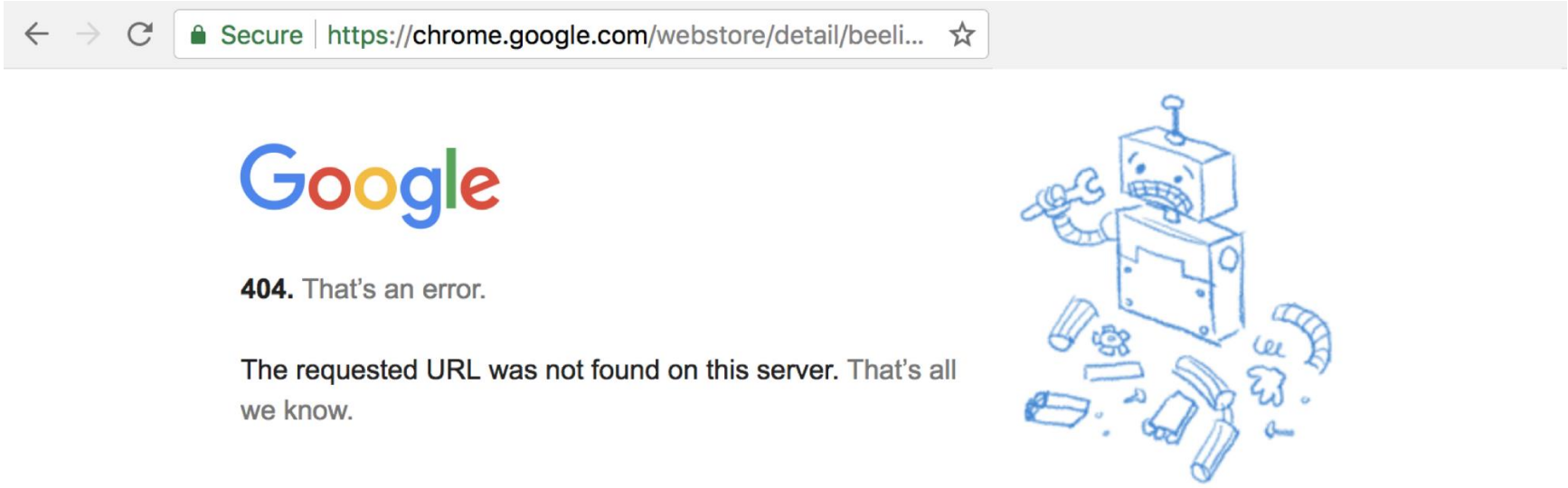
- Хостинг файлов
  - хранить и размещать файлы веб-сайта, а именно все документы HTML и связанные с ними активы, включая изображения, таблицы стилей CSS, файлы JavaScript, шрифты и видео.

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="styles.css">
  </head>

  <body>
    <!-- HTML elements go here -->
  </body>

  <script src="scripts.js"></script>
</html>
```



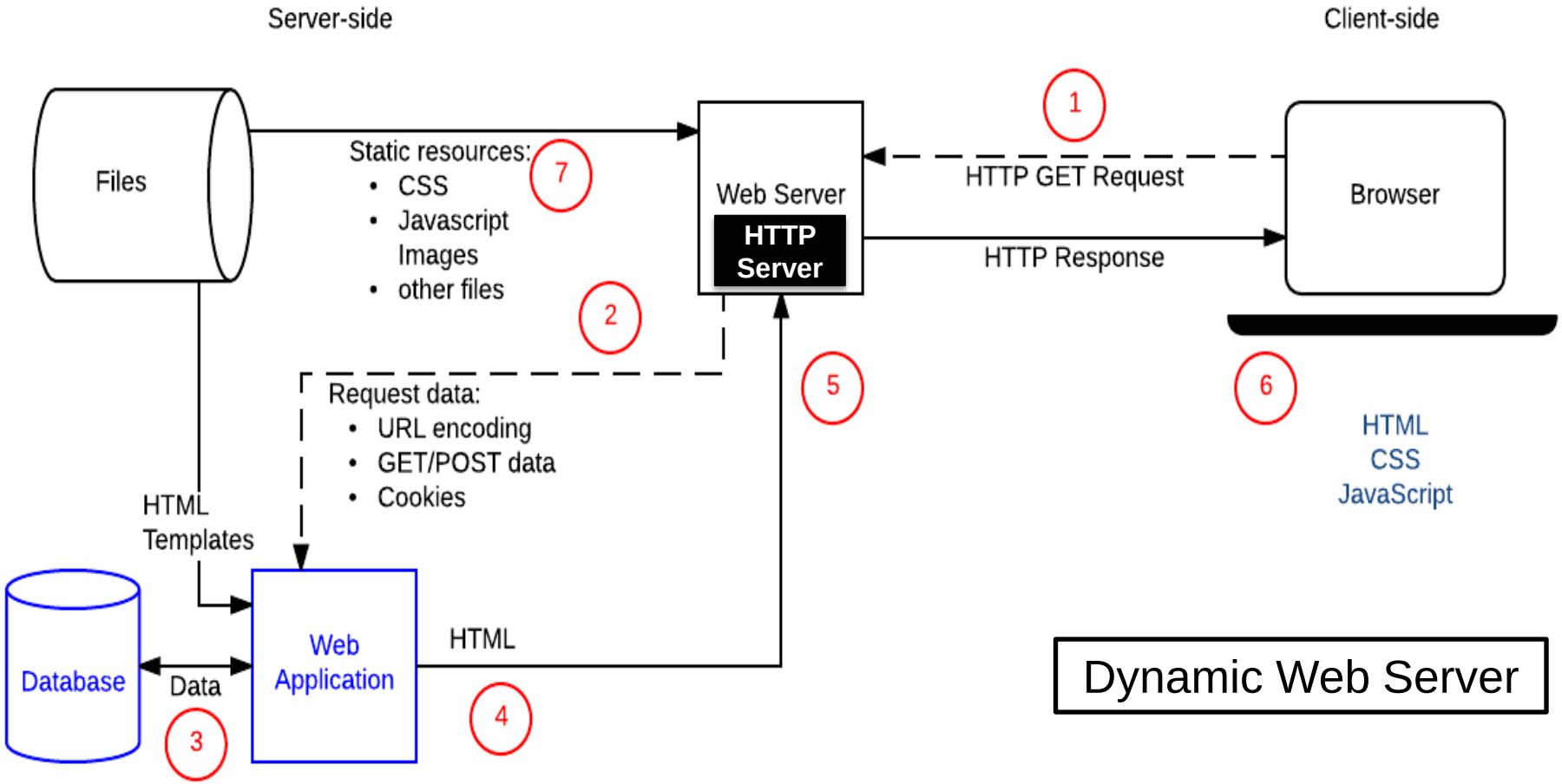
Что делает веб сервер

- Общение через HTTP
  - Всякий раз, когда браузеру нужен файл, размещенный на веб-сервере, браузер запрашивает файл через HTTP.
  - Когда запрос достигает правильного веб-сервера (аппаратного обеспечения), HTTP-сервер (программное обеспечение) принимает запрос и находит запрошенный документ (если он не возвращает ответ 404) и отправляет его обратно в браузер, также через HTTP



Веб-сервер с базой данных

- HTML-страницы обычно создаются путем вставки данных из базы данных в заполнители в шаблонах HTML

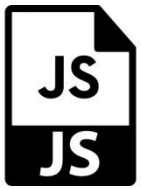


Веб технологии

- На стороне клиента
  - HTML(скелет страницы)

    - Язык разметки гипертекста для веб-страниц
    - Текст, смешанный с тегами разметки, для отображения различной информации
  - CSS(стиль страницы)

    - язык, который сообщает веб-браузеру, как стилизировать html-код
  - JavaScript (взаимодействие с пользователем)



    - язык программирования, который может динамически использовать данные HTML для «чего-то», без необходимости запрашивать у сервера дополнительную информацию.

Веб технологии

- На стороне сервера
  - LINUX(OS)
  - APACHE
    - Серверная программа, которая слушает сообщения из веб-браузера через Интернет и сообщает PHP, что это такое.
  - MYSQL
    - Память сервера / репозиторий данных (база данных).
    - Это хранилище данных в разных таблицах, которые можно получить с помощью языка SQL (структурированный язык запросов)
  - PHP (взаимодействие с сервером)
    - Мозг сервера.
    - Именно программный код определяет, как выглядит веб-страница.

Веб фреймворк (Web framework)

- Веб фреймворк или фреймворк веб-приложений это программная среда, предназначенная для поддержки разработки веб-приложений, включая веб-службы, веб-ресурсы и веб-APIs.
- Веб фреймворки обеспечивают стандартный способ создания и развертывания веб-приложений

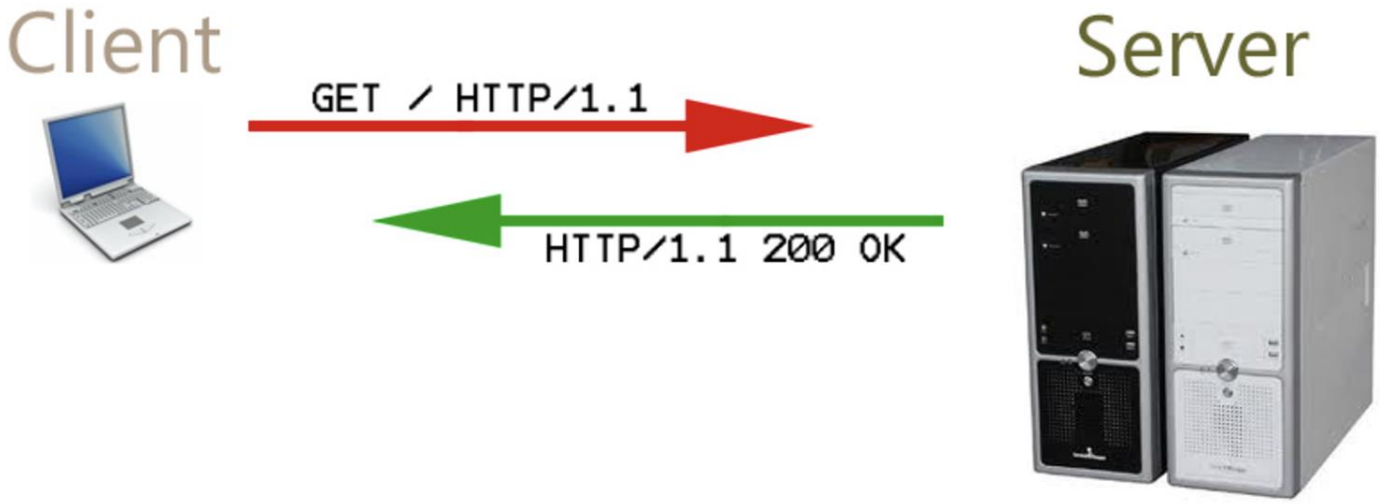
Веб фреймворк

- Веб-фреймворки представляют собой коллекции функций , объектов, правил и других кодовых конструкций, предназначенных для решения общих проблем, ускорения разработки и упрощения различных задач, стоящих перед конкретным доменом.

- Web Frameworks предоставляют инструменты и библиотеки, которые упрощают общие задачи веб-разработки, включая
  - Маршрутизация URL-адресов соответствующим обработчикам
  - Взаимодействие с базами данных
  - Поддержка сессий и авторизация пользователя
  - Форматирование вывода (например, HTML, JSON, XML)
  - Улучшение защиты от веб атак

Что делают Фреймворки

- Работайте напрямую с HTTP-запросами и ответами
  - Веб-серверы и браузеры обмениваются данными по протоколу HTTP
  - Веб-серверы ждут HTTP-запросов из браузера, а затем возвраща ют информацию в ответах HTTP
  - Веб-структуры позволяют писать упрощенный синтаксис, который будет генерировать серверный код для работы с этими запросам и и ответами



Что делают Фреймворки

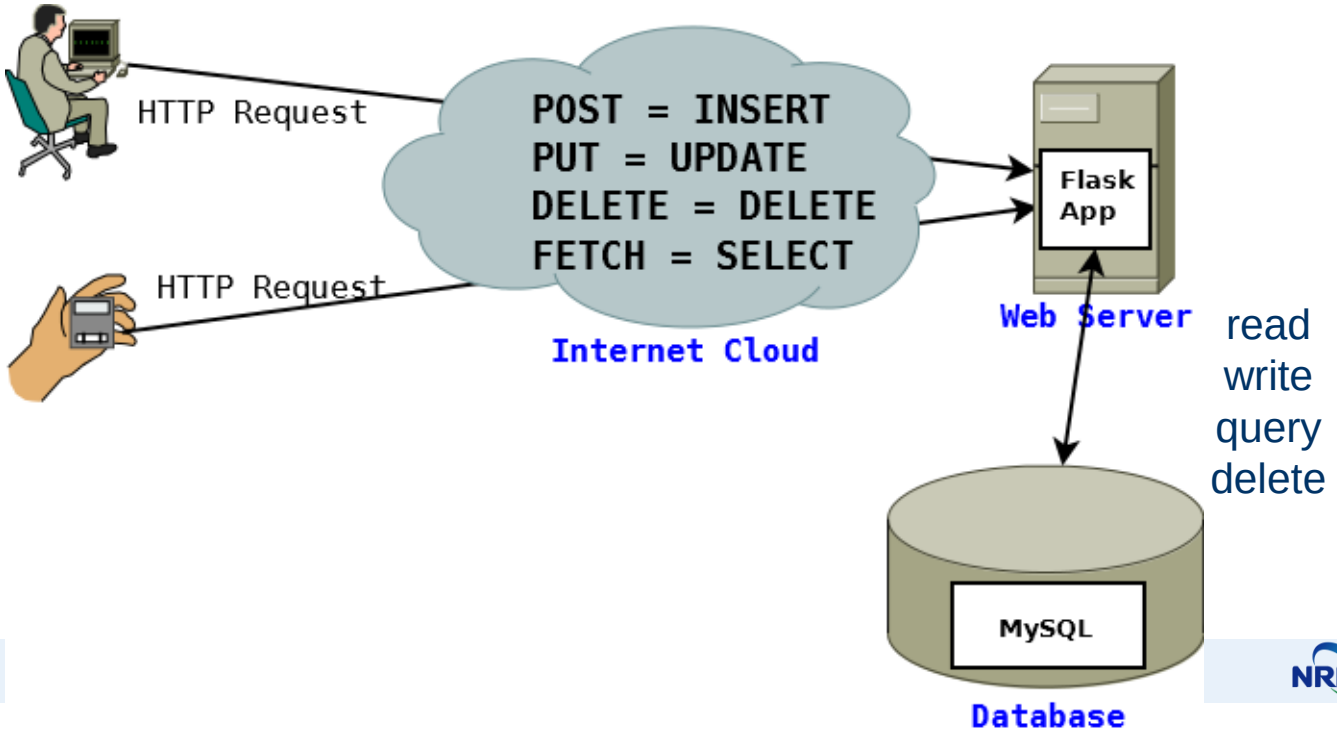
- Запросы маршрута к соответствующему обработчику
  - Большинство сайтов предоставят несколько различных ресурсов , доступных через отдельные URL-адреса.
  - Веб-структуры предоставляют простые механизмы для сопостав ления шаблонов URL-адресов определенным функциям обработ чика.

```
@app.route('/')
def index():
    return 'Home Page'

@app.route('/career/')
def career():
    return 'Career Page'

@app.route('/feedback/')
def feedback():
    return 'Feedback Page'
```

функции просмотра в Flask Фреймворк

Что делают Фреймворки

- Абстрагирует и упрощает доступ к базе данных
  - Сайты используют базы данных для хранения информации, которая будет использоваться совместно с пользователями и пользователями.
  - Веб-структуры часто предоставляют уровень базы данных, который абстрагирует операции чтения, записи, запроса и удаления базы данных



Что делают Фреймворки

- Предоставление данных
  - Веб-структуры предоставляют системам шаблонов, чтобы указать структуру выходного документа, используя заполнители для данных, которые будут добавляться при создании страницы.
  - Веб-структуры обеспечивают механизм, позволяющий легко генерировать другие форматы из сохраненных данных
  - (например, {{variable_name}}), который будет заменен на значения, переданные из функции просмотра при визуализации страницы.

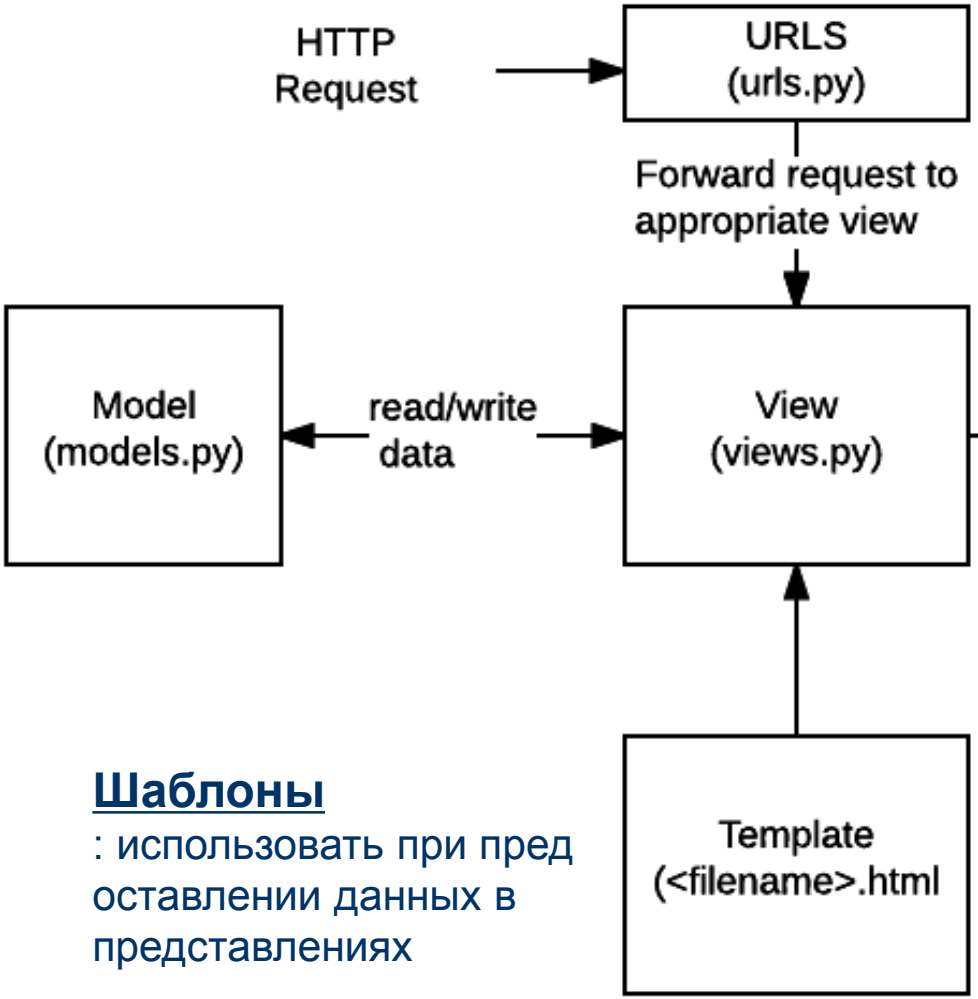
```
<!DOCTYPE html>
  <head>
    <title>{{ title }}</title>
  </head>

  <body>
    <h1>Hello, World!</h1>
    <h2>The time is: {{ time }}</h2>
  </body>
</html>
```

```
@app.route("/")
def hello():
   now = datetime.datetime.now()
   timeString = now.strftime("%Y-%m-%d %H:%M")
   templateData = {
      'title' : 'HELLO!',
      'time': timeString
      }
   return render_template('index.html', **templateData)
```

Python- код в Flask Фреймворк

Обзор Django

- Как создать главную страницу в Django Framework



URL-адреса

: переслать поддерживаемые URL-адреса в соответствующие функции просмотра

Функции просмотра

- получить запрошенные данные из моделей (базы данных)
- создавать HTML-страницы , отображающие данные
- вернуть страницы пользов ателю для просмотра в браузере

Шаблоны

: использовать при пред оставлении данных в представлениях

Типы веб фреймворков

Типы веб фреймворков

- Веб фреймворк Python

  - Flask отлично подходит для создания веб-приложений на встроенных устройствах

Содержание

- Основы Веб

- Веб-приложение с использованием Фреймворк веб
  - Веб-приложение с использованием Flask

Flask

- микро-веб-фреймворк, написанный на Python

  : Предоставляет только основной набор функций и полагается на расширен ия, чтобы сделать все остальное. (Нет набора инструментов или библиотек)

- отлично подходит для создания веб-приложений, работающих на вст роенных устройствах
- Веб-приложения с использованием Flask: Pinterest, LinkedIn, Reddit

Создание маршрутов - route

- route()
  - функция декоратора, которая решает, какой URL-адрес позволяет вызвать связанную функцию.
  - Пример

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello world‘
```

    - связать URL '/' с функцией просмотра hello ()
    - URL '/' указывает, где выполняется функция hello

Создание маршрутов - route

- Мы можем создать столько route, сколько потребуется нашим приложениям.

```
@app.route('/')
def index():
    return 'Home Page'

@app.route('/career/')
def career():
    return 'Career Page'

@app.route('/feedback/')
def feedback():
    return 'Feedback Page'
```

Запуск сервера

- run()
  - Запускает приложение на сервере разработки
  - Пример

```
if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```

    - debug: debug = True означает включение режима отладкиhost:
    - host: host = '0.0.0.0' означает, что веб-приложение будет доступно лю бому устройству в сети

Hello World во Flask

1

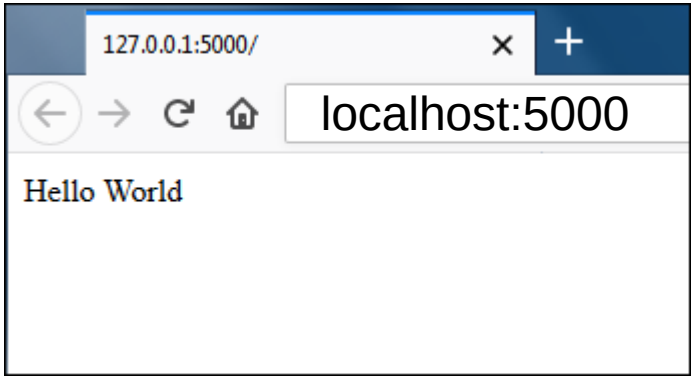
```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello World‘

if __name__ == '__main__':
    app.run()
```

2

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Командная строка

3



Веб броузер

#1 задача: Создание Routes

- Task(Задание)

  : Создание Routes как показано ниже

URL	функцией просмотра	Содержание веб-страницы
/	index()	‘My homepage’
/name	name()	‘Ваше имя(Daegeun Lee)’
/id	id()	Ваш ID(12345678)

- Use Tip(Подсказка)
  - Добавьте больше Routes , как показано ниже.

```
@app.route('/name')
def name():
    return ‘Daegeun Lee‘
```

Шаблон Flask

- Python Flask включает
  - **Werkzeug**– набор инструментов **WSGI**, реализующий запросы, ответы объекты и другие полезные функции

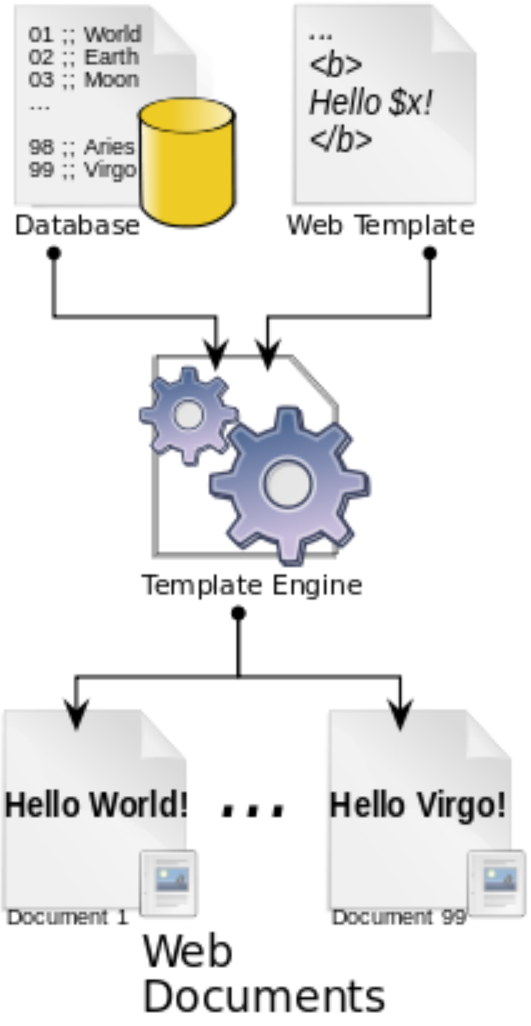
    : WSGI (интерфейс веб-сервера) - стандарт для разработки веб-прило жений Python
  - **Jinja 2**– Шаблонный движок Python

WSGI

Template Engine

Шаблоны и предоставление шаблонов

- Шаблон - это просто текстовый файл, содержащий статический HTML-код с некоторой специальной разметкой, обозначающей динамический контент
- Шаблон Rendering - это процесс, который заменяет динамическую разм етку и создает плоскую HTML-страницу
- Шаблон движок под названием Jinja, который выполняет фактическую работ у по разбору шаблонов и преобразован ию их в плоскую HTML-страницу

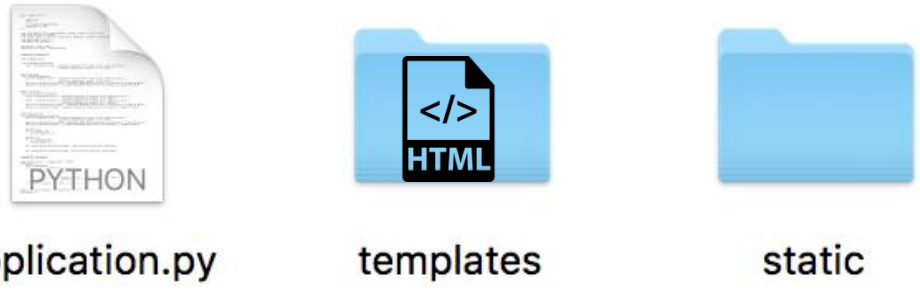


Предоставление шаблонов

- render_template()
  - Отображает шаблон и возвращает HTML как строку
  - Пример

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html‘)
```

  - Flask будет искать index.html в каталоге под названием templates, в том же каталоге, что и файл python
  - Будет отображен новый шаблон HTML (index.html).



Структура Flask

#2 задача: Предоставление шаблонов

- Task(Задание)

  : Создайте новый HTML-файл и напишите код python для предоставления HTML-файла с помощью render_template ()

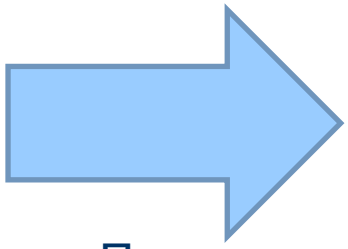
- Use Tip(Подсказка)
  - Сохраните файл как xxx.html в каталоге шаблонов

Шаблон рендеринга с данными

- Имя внутри двойных фигурных скобок {{ }} представляет собой переменную, значение которой будет указано во время визуал изации шаблона.

```
<html>
<head>
    <title>Title</title>
</head>
<body>
    <p>Name: {{ name }}</p>
</body>
</html>
```

index.html(шаблон)



После рендеринга

```
<html>
<head>
    <title>Title</title>
</head>
<body>
    <p>Name: LEE</p>
</body>
</html>
```

index.html

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html', name='LEE')
```

python file

#3 задача: Шаблон рендеринга с данными

❑ Task(Задание)

: Запишите код python, который передает следующие аргументы и значения ключевых слов в шаблон HTML и отображает HTML-файл

Ключевые аргументы	Значения
Title	‘Ваш университет’
name	‘Ваше имя’
ID	Ваш ID

```
<html>
<head>
    <title>{{Title}}</title>
</head>
<body>
    <h1>Hello {{ name }}</h1>
    <h1>Your ID is {{ ID }}</h1>
</body>
</html>
```

index.html(шаблон)

❑ Use Tip(Подсказка)

- Передайте аргументы ключевого слова, разделив их запятой (,)

Шаблон рендеринга с данными

- В случае, если достаточно много аргументов для перехода к render_template( ), создайте словарь и примените оператор ** на нем, чтобы доставить аргументы ключевого слова функции

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def index():
    name = “Lee”
    ID      = 12345
    templateData = {
        ‘name’ : name,
        ‘ID’      : ID,
    }
    return render_template('index.html‘, **templateData)
```

#4 задача: Шаблон рендеринга с данными

- Task(Задание)

  : Запишите код python, который передает следующие аргументы и значения ключевых слов в шаблон HTML и отображает HTML-файл, создавая словарь и применяя оператор **

Ключевой аргумент	Значения
Title	‘домашняя страница’
Name	‘имя’
Age	возраст
City	‘родной город’
Hobby	‘хобби’

```
<html>
<head>
   <title>{{Title}}</title>
</head>
<body>
   <h1>My name is {{ Name }}</h1>
   <h2>I’m {{ Age }} years old</h2>
   <h2>I live in {{ City }} </h2>
   <h2>My hobby is {{ Hobby }} </h2>
</body>
</html>
```

Обслуживание статических файлов с помощью Flask

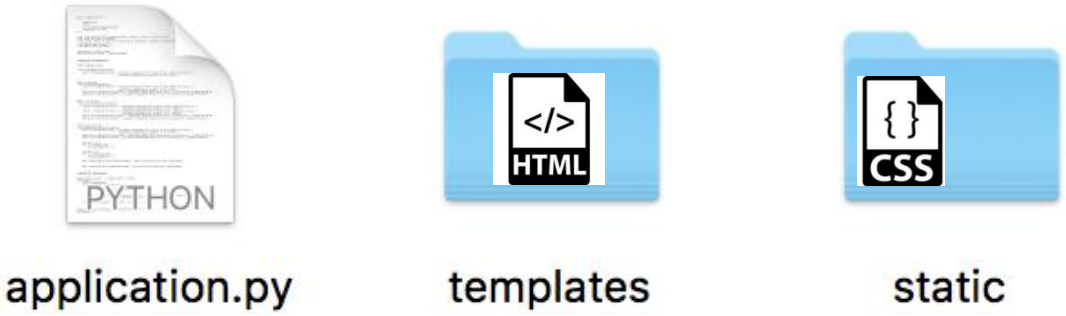
- Наиболее распространенные статические файлы, размещаемые на веб-странице
  1. CSS-файлы для стилизации страницы
  2. Файлы JavaScript, которые добавляют динамическое поведение на страницу

```
<!-- templates/index.html -->
<html>
<head>
    <title> IoRT </title>
    <link rel="stylesheet" href = “/static/style.css">
    <script src="/static/scripts.js" charset="utf-8"></script>
</head>
<body>
    <h1> Internet of Robotic Things </h1>
    <h2> IoRT Course </h2>
</body>
</html>
```

Стилизация страницы

- Стилизация с помощью CSS

```
/* static/style.css */
h1 {
    color: blue;
    font-size: 46px;
}
h2 {
    color: black;
    font-size: 36px;
}
body {
    background: lightblue;
}
```



Динамическое поведение на странице

- Добавление динамического поведения с JavaScript

```
// static/scripts.js
// JavaScript code to dynamically change the background color every second

var bg1 = ‘white';
var bg2 = ‘lightyellow';

var color = true;

setInterval(function () {
  document.body.style.backgroundColor = (color ? bg1 : bg2)
  color = !color;
}, 1000);
```