Подготовка к экзамену по веб-разработке

Краткая самостоятельная шпаргалка: тема, суть простыми словами, ссылка на подробный раздел внутри этого же документа и пример экзаменационного вопроса.

Карта тем

1. Адресация, IPv4, IPv6

Кратко: IP-адрес нужен, чтобы доставить пакет до конкретного узла сети. IPv4 записывается как четыре числа и занимает 32 бита, поэтому адресов мало; IPv6 занимает 128 бит и записывается шестнадцатеричными блоками. Доменное имя само по себе не является сетевым адресом: перед подключением DNS превращает его в IP.

Подробнее: адресация и IP

Вопрос: Чем IPv4 отличается от IPv6?

2. TCP, порты, NAT и подсети

Кратко: TCP устанавливает соединение и контролирует доставку данных: порядок, подтверждения и повторную отправку. IP выбирает устройство, а порт выбирает приложение на этом устройстве. Маска подсети отделяет адрес сети от адреса хоста, а NAT позволяет приватным адресам локальной сети выходить наружу через публичный IP.

Подробнее: TCP, порты, NAT

Вопрос: Зачем нужен NAT и как он связан с портами?

3. DNS, URI, URL, URN

Кратко: DNS сопоставляет доменное имя с IP-адресом, чтобы браузер понял, к какому серверу подключаться. URI - общее понятие для идентификатора ресурса. URL является частным случаем URI: он указывает, где находится ресурс и как его получить; URN задаёт устойчивое имя без обязательного сетевого адреса.

Подробнее: DNS и идентификаторы

Вопрос: Почему URL является URI, но не каждый URI является URL?

4. HTTP, HTTPS, HTTP/3

Кратко: HTTP - прикладной протокол обмена сообщениями: клиент отправляет запрос, сервер возвращает ответ со статусом, заголовками и, возможно, телом. GET обычно получает данные, POST отправляет данные в теле запроса. HTTPS - это HTTP внутри TLS-соединения, а HTTP/3 сохраняет смысл HTTP, но использует QUIC поверх UDP.

Подробнее: HTTP-протоколы

Вопрос: Чем GET отличается от POST?

5. Идентификация, аутентификация, OAuth 2.0

Кратко: Идентификация отвечает на вопрос "кто пользователь?", аутентификация проверяет доказательство личности, например пароль или код. Авторизация решает, какие действия разрешены уже проверенному пользователю. OAuth 2.0 используется, чтобы стороннее приложение получило ограниченный access token без получения пароля пользователя.

Подробнее: доступ и OAuth

Вопрос: Почему OAuth 2.0 называют протоколом авторизации?

6. HTML-документ

Кратко: HTML описывает смысловую структуру страницы: заголовки, абзацы, ссылки, изображения, таблицы и разделы. В `head` находятся служебные данные: кодировка, viewport, заголовок вкладки, подключение CSS и скриптов. В `body` находится видимый контент; семантические теги помогают показать назначение частей страницы.

Подробнее: HTML-структура

Вопрос: Для чего нужен meta viewport?

7. HTML-формы

Кратко: HTML-форма собирает ввод пользователя и отправляет его на сервер по адресу из `action` методом `GET` или `POST`. Атрибут `name` определяет имя параметра в запросе, а `id` обычно нужен для связи поля с `label`. `required`, `type` и другие атрибуты дают базовую проверку в браузере.

Подробнее: HTML Forms

Вопрос: Чем radio отличается от checkbox?

8. CSS selectors и специфичность

Кратко: CSS-селектор выбирает элементы, к которым применяются свойства. При конфликте правил учитываются каскад, специфичность и порядок записи. ID сильнее класса, класс сильнее тега; если специфичность равна, побеждает правило, написанное ниже.

Подробнее: CSS selectors

Вопрос: Какой селектор сильнее: `body .content` или `div p`?

9. CSS box model и базовые свойства

Кратко: Каждый элемент в CSS рассматривается как прямоугольный блок: content, padding, border и margin. При `content-box` ширина задаёт только content, а padding и border добавляются сверху. При `border-box` заданная ширина уже включает content, padding и border, поэтому размеры проще контролировать.

Подробнее: блочная модель

Вопрос: Какая ширина у блока с `width:100px` и `box-sizing:border-box`?

10. Flexbox и Grid

Кратко: Flexbox удобен для одномерной раскладки: элементы распределяются вдоль главной оси, а поперечная ось используется для выравнивания. Grid решает двумерные задачи: одновременно управляет строками, столбцами, линиями, ячейками и областями. `fr`, `repeat()` и `minmax()` помогают задавать гибкую сетку.

Подробнее: Flexbox и Grid

Вопрос: Какое значение `align-items` по умолчанию?

11. Django models и ORM queries

Кратко: Модель Django описывает таблицу базы данных: класс - таблица, поле - колонка, объект - строка. ORM позволяет строить SQL-запросы через Python-методы вроде `filter()`, `get()`, `annotate()` и `aggregate()`. QuerySet ленивый: запрос обычно выполняется только тогда, когда реально нужны данные.

Подробнее: Django models и queries

Вопрос: Когда использовать `select_related`, а когда `prefetch_related`?

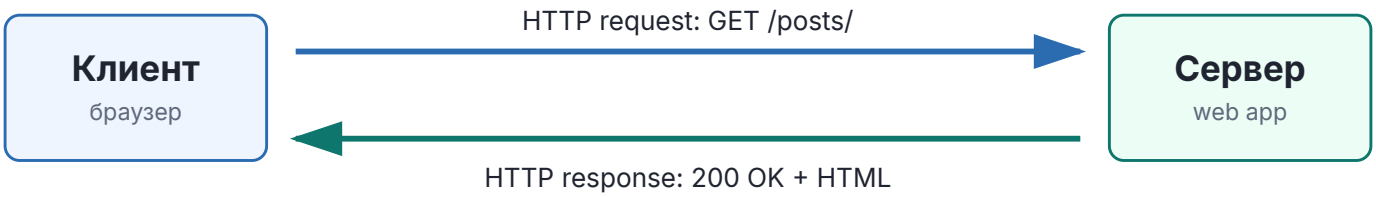
12. Django views, URLs, templates, forms

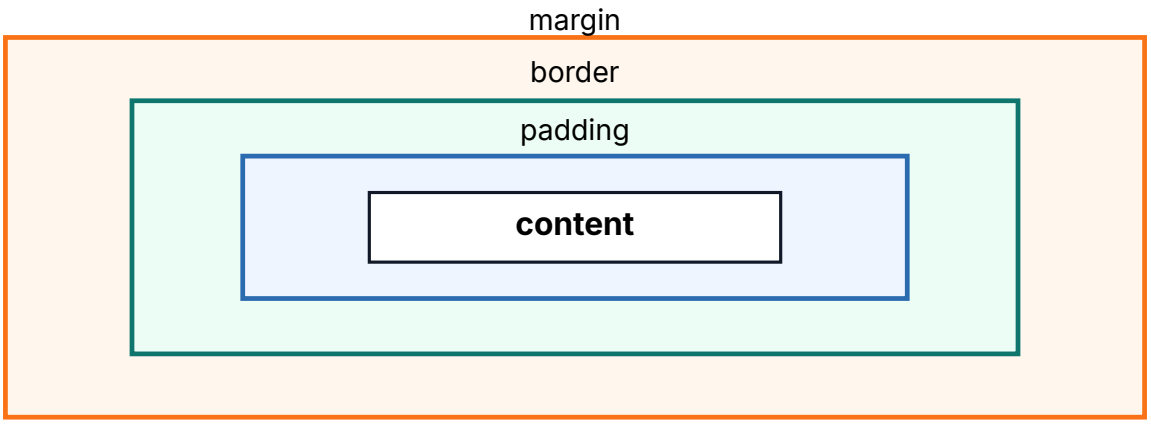
Кратко: Django view принимает `request` и возвращает HTTP-ответ: HTML, redirect, JSON или ошибку. URL-маршрут связывает путь с view, шаблон получает context и отображает данные, а форма описывает поля, валидирует ввод и отдаёт очищенные значения через `cleaned_data`. FBV проще как функция, CBV удобнее для типовых страниц.

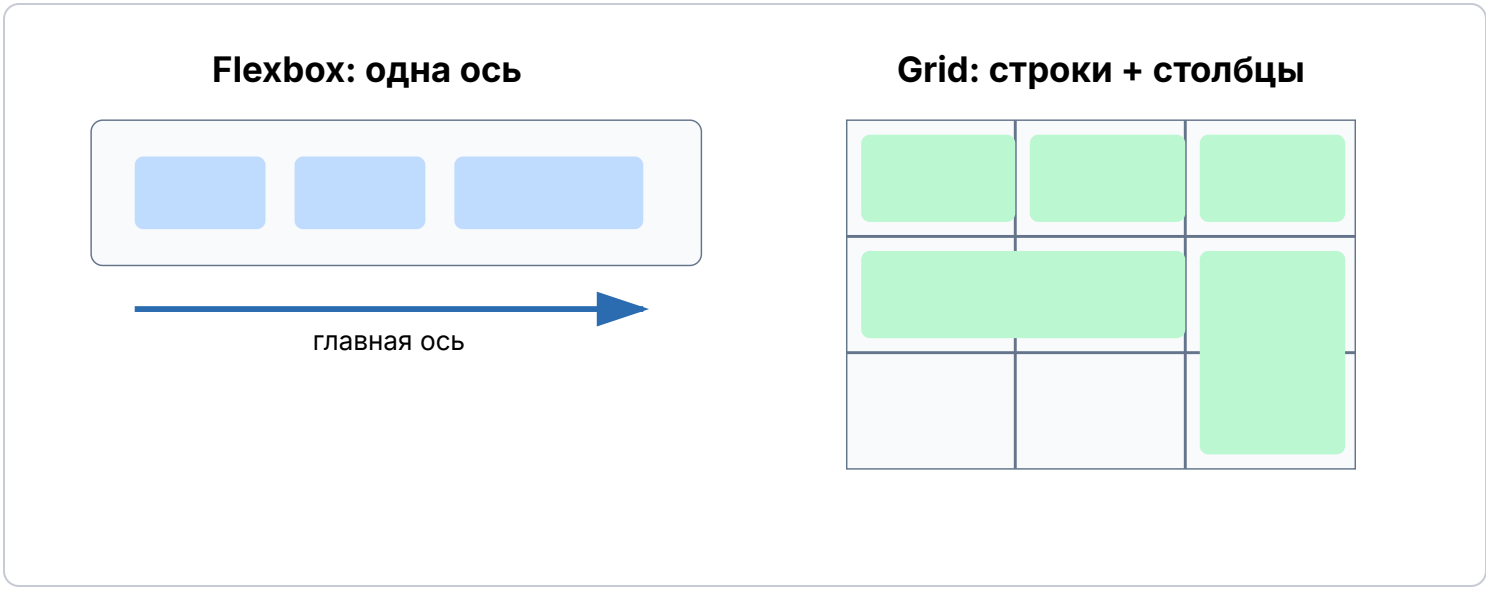
Подробнее: Django web layer

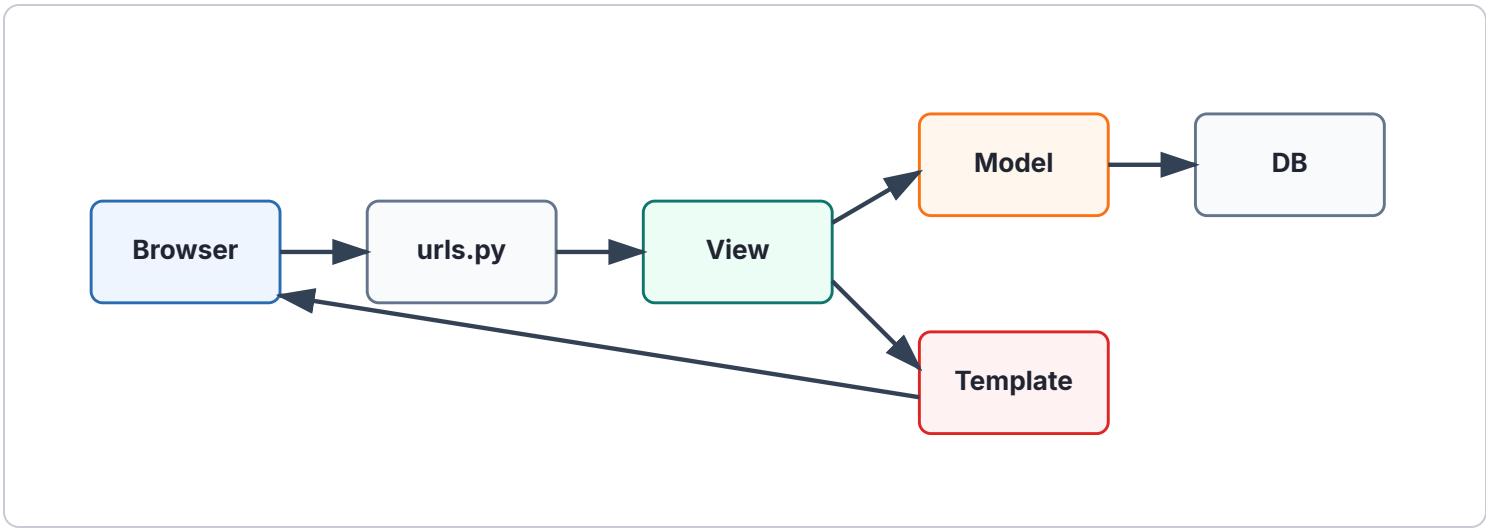
Вопрос: Что будет, если в `ListView` не указать ни `model`, ни `queryset`?

Схемы









Подробная теория

Адресация и IP

← к краткой карточке

IP-адрес нужен, чтобы доставить пакет до конкретного узла сети. Доменное имя удобно человеку, но сеть работает с IP.

IPv4 32 бита, запись вроде `192.168.1.10`. Адресов мало, поэтому используют приватные диапазоны и NAT.

IPv6 128 бит, запись вроде `2001:db8::1`. Адресов намного больше, запись можно сокращать через `::`.

Приватные IPv4-диапазоны: `10.0.0.0/8`, `172.16.0.0/12`, `192.168.0.0/16`. `127.0.0.1` - localhost, обращение к самому себе.

TCP, порты, NAT и подсети

← к краткой карточке

TCP устанавливает соединение через три шага: `SYN`, `SYN-ACK`, `ACK`. Он следит за порядком данных, потерями и повторной отправкой.

Порт выбирает приложение внутри одного IP-адреса. HTTP обычно работает на `80`, HTTPS - на `443`.

Маска подсети показывает, где в IP-адресе часть сети, а где часть хоста. В `192.168.1.10/24` сеть - `192.168.1.0`, хост - `.10`.

NAT переводит приватные адреса во внешний публичный адрес. PAT дополнительно использует порты, поэтому много внутренних устройств могут выходить через один публичный IP.

DNS и URI/URL/URN

← к краткой карточке

DNS - система имён. Она ищет IP-адрес по домену. Упрощённо: кэш браузера, кэш ОС, resolver, root, TLD, авторитативный DNS-сервер.

Запись	Смысл
`A`	домен в IPv4
`AAAA`	домен в IPv6
`CNAME`	псевдоним
`MX`	почтовый сервер
`TXT`	служебный текст, например SPF/DKIM

URI - общий идентификатор. URL - URI с адресом и способом получения: `https://example.com/path?x=1#top`. URN - устойчивое имя, например ISBN или UUID.

HTTP, HTTPS, HTTP/3

← к краткой карточке

HTTP-сообщение состоит из стартовой строки, заголовков, пустой строки и необязательного тела.

```
GET /posts/?page=2 HTTP/1.1
Host: example.com
```

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>...</html>
```

`GET` обычно получает данные, параметры видны в URL. `POST` обычно отправляет данные в теле запроса.

Коды: `2xx` успех, `3xx` редирект, `4xx` ошибка клиента, `5xx` ошибка сервера.

HTTPS - HTTP поверх TLS. HTTP/3 - HTTP поверх QUIC/UDP; смысл методов и кодов не меняется.

Идентификация, аутентификация, авторизация, OAuth 2.0

← к краткой карточке

Идентификация: пользователь сообщает, кто он. Аутентификация: система проверяет доказательство. Авторизация: система проверяет права.

OAuth 2.0 позволяет приложению получить access token без знания пароля пользователя. Права ограничиваются scope, сроком жизни токена и настройками сервиса.

1. Клиент отправляет пользователя на authorization server.
2. Пользователь подтверждает доступ.
3. Клиент получает authorization code.
4. Код меняется на access token.
5. Access token используется для API-запросов.

HTML-структура

← к краткой карточке

HTML задаёт смысловую структуру документа. Комментарий пишется так: `<!-- comment -->`.

`head` содержит `meta`, `title`, `link`, `script`. `meta viewport` нужен, чтобы мобильный браузер корректно рассчитывал ширину страницы.

Блочные элементы занимают строку: `div`, `p`, `section`, `form`. Строчные идут внутри текста: `span`, `a`, `strong`, `em`.

Семантические теги: `header`, `nav`, `main`, `section`, `article`, `aside`, `footer`.

HTML Forms

← к краткой карточке

`form` отправляет данные. `action` задаёт URL, `method` - способ отправки: `get` или `post`.

`id` уникален на странице и связывает поле с label. `name` - имя параметра, которое уйдёт на сервер.

```
<label for="email">Email</label>
<input id="email" name="email" type="email" required>
```

`radio` выбирает один вариант из группы с одинаковым `name`. `checkbox` - независимый флажок. `fieldset` и `legend` группируют поля.

`placeholder` - подсказка внутри поля. `required` - поле обязательно. У кнопки лучше явно писать `type="submit"`, `type="button"` или `type="reset"`.

CSS selectors и специфичность

← к краткой карточке

Селекторы: `*`, `p`, `.class`, `#id`, `div p`, `div > p`, `h2 + p`, `input[required]`, `:hover`, `:nth-child(2)`.

Специфичность: inline-style сильнее id, id сильнее class/attribute/pseudo-class, class сильнее tag.

`body .content` сильнее `div p`, потому что класс имеет больший вес, чем селектор тега.

Если специфичность одинаковая, побеждает правило ниже в CSS. Наследуются, например, `color` и `font-family`; не наследуются `margin`, `padding`, `border`.

CSS box model и базовые свойства

← к краткой карточке

Блок состоит из content, padding, border и margin. `content-box` считает `width` только как ширину контента. `border-box` включает padding и border в width.

```
width: 100px;
padding: 10px;
border: 5px solid;
box-sizing: border-box;
```

Реальная ширина такого блока - `100px`.

Шрифты: `font-family`, `font-size`, `font-weight`, `font-style`. Текст: `text-align`, `text-decoration`, `line-height`, `letter-spacing`, `text-shadow`.

Flexbox и Grid

← к краткой карточке

Flexbox создаётся через `display: flex` или `display: inline-flex`. Главная ось зависит от `flex-direction`.

`justify-content` выравнивает по главной оси, `align-items` - по поперечной. Значение `align-items` по умолчанию - `stretch`.

`flex-grow` отвечает за рост, `flex-shrink` - за сжатие, `flex-basis` - за базовый размер.

Grid создаётся через `display: grid`. `grid-template-columns` задаёт столбцы, `grid-template-rows` - строки. `fr` - доля свободного места. `repeat()` сокращает повтор, `minmax()` задаёт минимум и максимум.

Django models и ORM queries

← к краткой карточке

Модель - Python-класс таблицы. Поле - колонка. Объект - строка. Если не указать primary key, Django создаёт `id`.

`CharField` требует `max_length`. `null=True` разрешает NULL в БД, `blank=True` разрешает пустое значение в форме.

`ForeignKey` - многие к одному. `related_name` задаёт обратную связь. `ManyToManyField(..., through="Model")` задаёт промежуточную модель.

Метод	Что возвращает
`all()`	QuerySet всех объектов
`filter()`	QuerySet по условию
`get()`	один объект или ошибка
`aggregate()`	словарь с итогом по выборке
`annotate()`	QuerySet с вычисляемым полем у каждого объекта

`select_related` делает SQL JOIN для ForeignKey/OneToOne. `prefetch_related` делает отдельный запрос для ManyToMany и обратных связей.

Django views, URLs, templates, forms

← к краткой карточке

FBV - функция. Она принимает `request` и возвращает response. В `render()` первым аргументом должен быть `request`.

```
def home(request):
    return render(request, "index.html", {"title": "Home"})
```

CBV вызывается через `.as_view()`. `TemplateView` показывает шаблон, `ListView` показывает список, `DetailView` - один объект.

В `ListView` нужны `model` или `queryset`. Дополнительный context добавляют через `get_context_data()`, список меняют через `get_queryset()`.

`path(route, view, kwargs, name)` задаёт маршрут. `app_name` задаёт namespace. В шаблоне URL строится так: `{% url 'app:name' %}`.

В шаблонах переменные пишутся как `{{ value }}`, теги как `{% tag %}`, фильтры как `{{ value|lower }}`. Если переменной нет, обычно выводится пустая строка.

`forms.Form` - обычная форма. `forms.ModelForm` - форма на основе модели. Для связи между полями используют общий метод `clean()`. После `is_valid()` доступны `cleaned_data` и `errors`.