# 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 указывает узел сети. IPv4 - 32 бита, IPv6 - 128 бит. Домен через DNS превращается в IP.

**Подробнее:** адресация и IP

**Вопрос:** Чем IPv4 отличается от IPv6?

### 2. TCP, порты, NAT и подсети

**Кратко:** TCP отвечает за надёжную передачу, порт выбирает приложение, маска делит сеть, NAT переводит приватные адреса в публичные.

**Подробнее:** TCP, порты, NAT

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

### 3. DNS, URI, URL, URN

**Кратко:** DNS находит IP по имени. URI - общий идентификатор, URL - адрес получения ресурса, URN - устойчивое имя.

**Подробнее:** DNS и идентификаторы

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

## 4. HTTP, HTTPS, HTTP/3

**Кратко:** HTTP - запрос и ответ. HTTPS добавляет TLS. HTTP/3 переносит HTTP на QUIC поверх UDP.

**Подробнее:** HTTP-протоколы

**Вопрос:** Чем GET отличается от POST?

## 5. Идентификация, аутентификация, OAuth 2.0

**Кратко:** Идентификация - кто ты, аутентификация - докажи, авторизация - что тебе можно. OAuth 2.0 выдаёт ограниченный доступ без передачи пароля.

**Подробнее:** доступ и OAuth

**Вопрос:** Почему OAuth 2.0 называют протоколом авторизации?

## 6. HTML-документ

**Кратко:** HTML описывает структуру страницы. `head` хранит служебные данные, `body` - видимое содержимое.

**Подробнее:** HTML-структура

**Вопрос:** Для чего нужен meta viewport?

## 7. HTML-формы

**Кратко:** Форма отправляет данные на сервер. `name` уходит в запросе, `id` связывает поле с `label`.

**Подробнее:** HTML Forms

**Вопрос:** Чем radio отличается от checkbox?

## 8. CSS selectors и специфичность

**Кратко:** Селектор выбирает элементы. Специфичность решает, какое CSS-правило победит при конфликте.

**Подробнее:** CSS selectors

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

## 9. CSS box model и базовые свойства

**Кратко:** Блок состоит из content, padding, border и margin. `border-box` включает padding и border в заданную ширину.

**Подробнее:** блочная модель

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

## 10. Flexbox и Grid

**Кратко:** Flexbox раскладывает по одной оси. Grid работает со строками и столбцами одновременно.

**Подробнее:** Flexbox и Grid

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

## 11. Django models и ORM queries

**Кратко:** Модель описывает таблицу. ORM строит запросы через Python. QuerySet ленивый и поддерживает цепочки методов.

**Подробнее:** Django models и queries

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

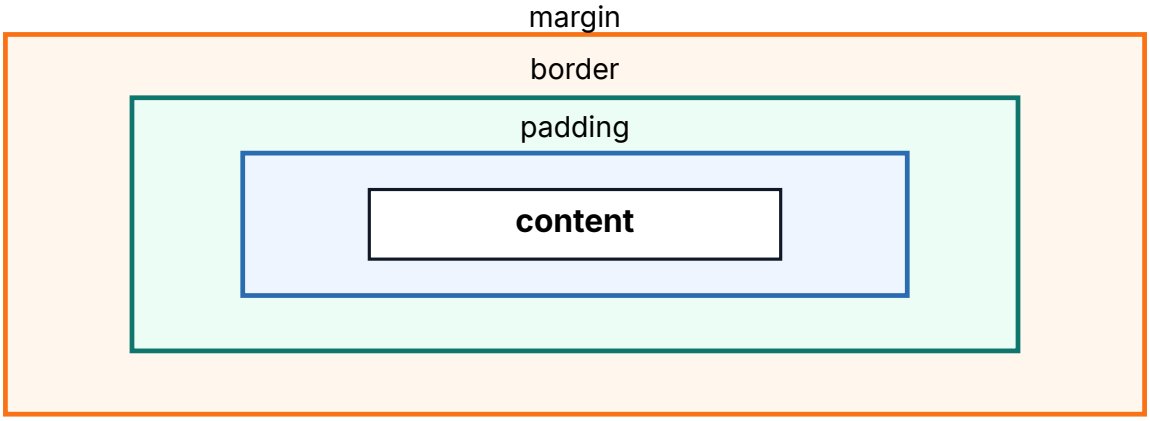
### 12. Django views, URLs, templates, forms

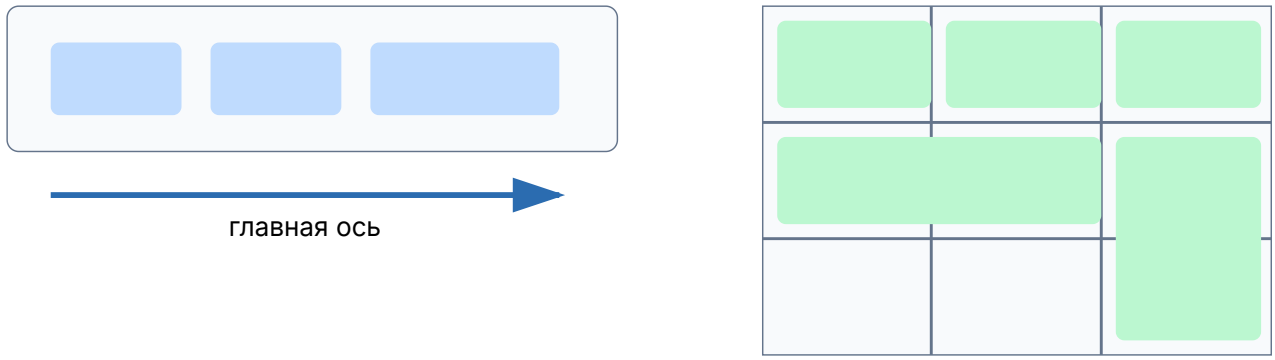
**Кратко:** View принимает request и возвращает response. URL связывает путь и view. Template получает context. Form валидирует данные.

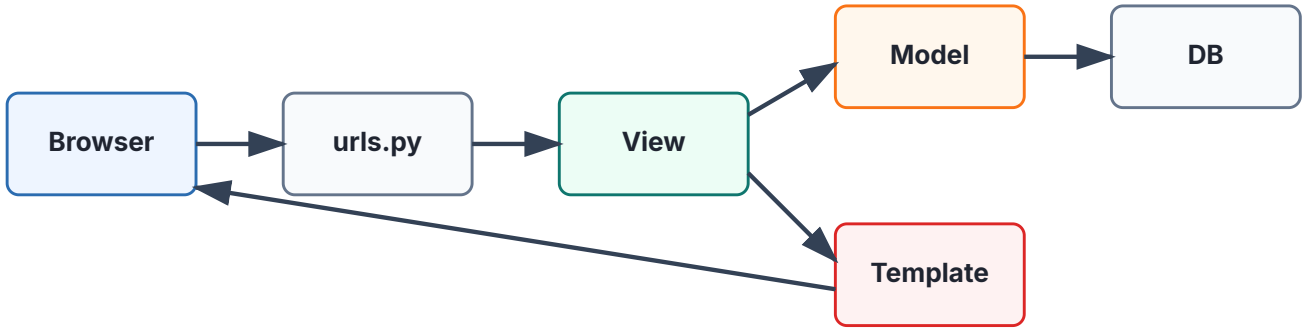
**Подробнее:** Django web layer

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

## Схемы




# Подробная теория

### Адресация и IP

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

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

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

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

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

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

`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

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

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

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

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

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`.