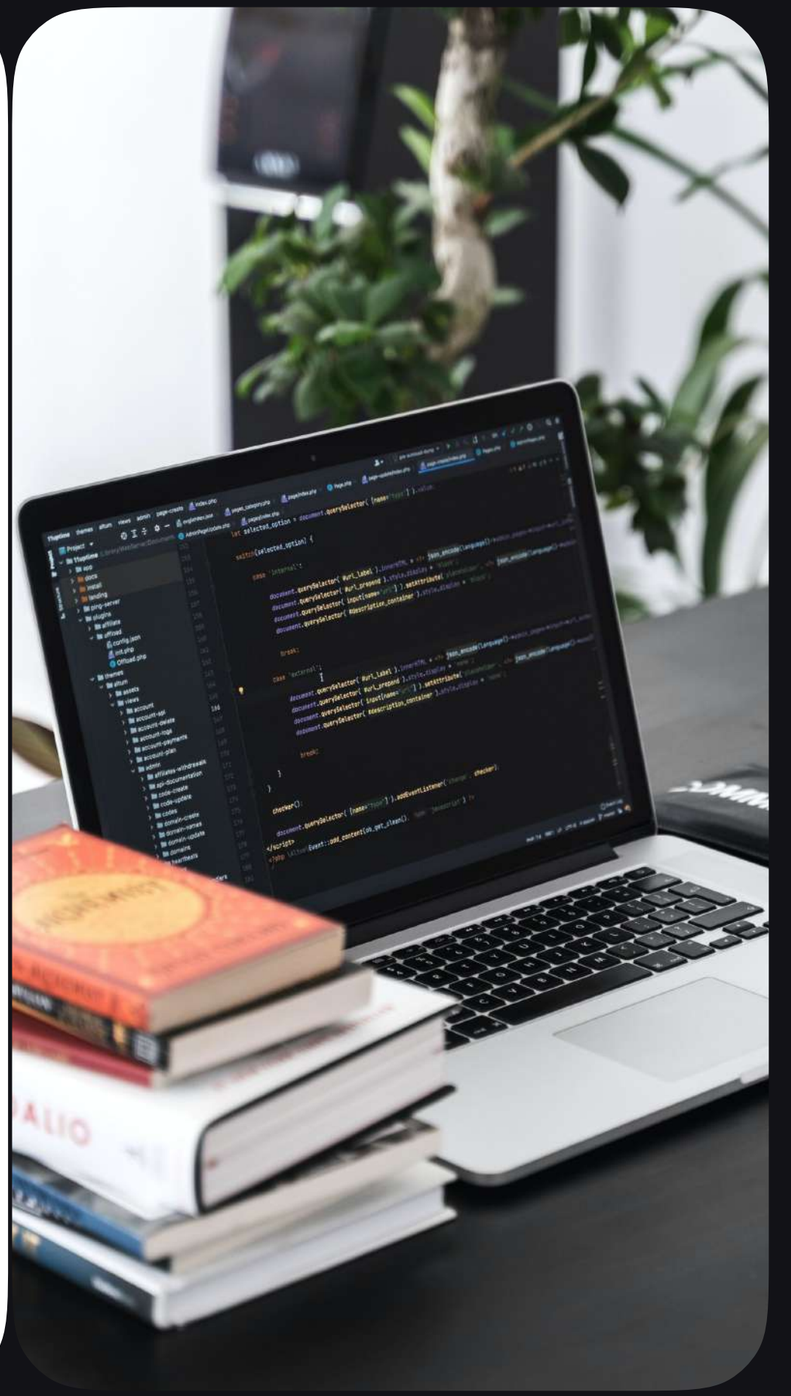
Модуль 4 Занятие 3

Рендеринг HTML



Шаблоны

HTML

CSS

Bootstrap

\+

Продолжаем создавать новостной портал

Шаблоны

Шаблон — это HTML файл, который также может содержать динамическое содержимое. Процесс, во время которого динамическое содержимое встраивается в страницу в HTML-страницу, называется **рендеринг** или **отрисовка.**

Flask использует **библиотеку Jinja2** для рендеринга шаблонов.

render_template()

Для рендеринга шаблонов используется функция render_template(). Функция принимает имя шаблона и переменные, которые будут встроены в шаблон в качестве аргументов ключевого слова.

На данном этапе вызовем функцию без дополнительных аргументов.

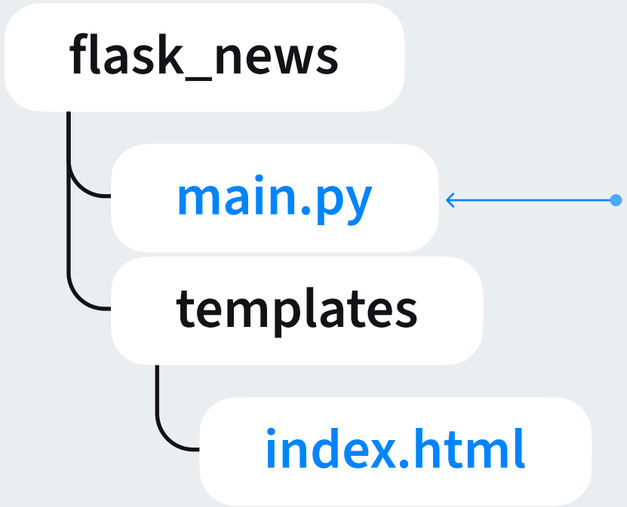
```
from flask import Flask, render_template

app = Flask(__name__)


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

Использование шаблонов

По умолчанию Flask ищет шаблоны в папке **templates**, создайте его и создайте в нем файл **index.html**, а также измените содержимое файлов.




```python
from flask import Flask,
render_template

app = Flask(__name__)


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

```html
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <title>Новостной сайт</title>
 </head>
 <body>
   <h1>Скоро тут будут новости!</h1>
   <p>Следите за обновлениями.</p>
 </body>
</html>
```

Шаблонизатор Jinja

Шаблонизатор — программа, которая объединяет отдельные шаблоны в конечную итоговую страницу.

Jinja — шаблонизатор для языка программирования Python.

В шаблоне можно использовать следующие выражения:

- ✓ `{% ... %}` — для управляющих структур, например, циклов и условий;
- ✓ `{{ ... }}` — для вывода значений переменных, вычисления выражений, вызова функций;
- ✓ `{# ... #}` — для комментариев.

Вывод значений переменных

main.py

```
from flask import Flask,
render_template

app = Flask(__name__)


@app.route('/')
def index():
    return render_template(
        'index.html',
        title='Новостной сайт',
        text='Скоро тут будут новости!'
    )
```

index.html

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <title>{{ title }}</title>
 </head>
 <body>
   <h1>{{ title }}</h1>
   <p>{{ text }}</p>
 </body>
</html>
```

Использования словаря

Если функции render_template() необходимо передать большое количество аргументов, то можно использовать словарь и оператор ** для распаковки. Такой словарь обычно называют словарь **context.**

Словарь можно и не распаковывать, тогда обращаться к аргументу можно, например, через **context['title'].**

main.py

```
from flask import Flask,
render_template

app = Flask(__name__)


@app.route('/')
def index():
 context = {
       title='Новостной сайт',
       text='Скоро тут будут новости!'
  }
   return render_template(
       'index.html',
       **context
   )
```

Практика

1

В проекте flask_news создайте новый шаблон news_detail.html:

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <title>{{ title }}</title>
 </head>
 <body>
   <h1>Новость: {{ title }}</h1>
   <p>{{ text }}</p>
 </body>
</html>
```

2

Измените функцию представление, связанную с маршрутом «/news_detail/<id>» и передавайте функции render_template() переменные title и text. Значения этих переменных возьмите из списка news.

Теги

Теги бывают одиночные и парные (контейнеры).

Парные

состоят из двух тегов — открывающего и закрывающего, например:

```
<h1>Скоро тут будут новости!</h1>
<p>Следите за обновлениями.</p>
```

Одиночные

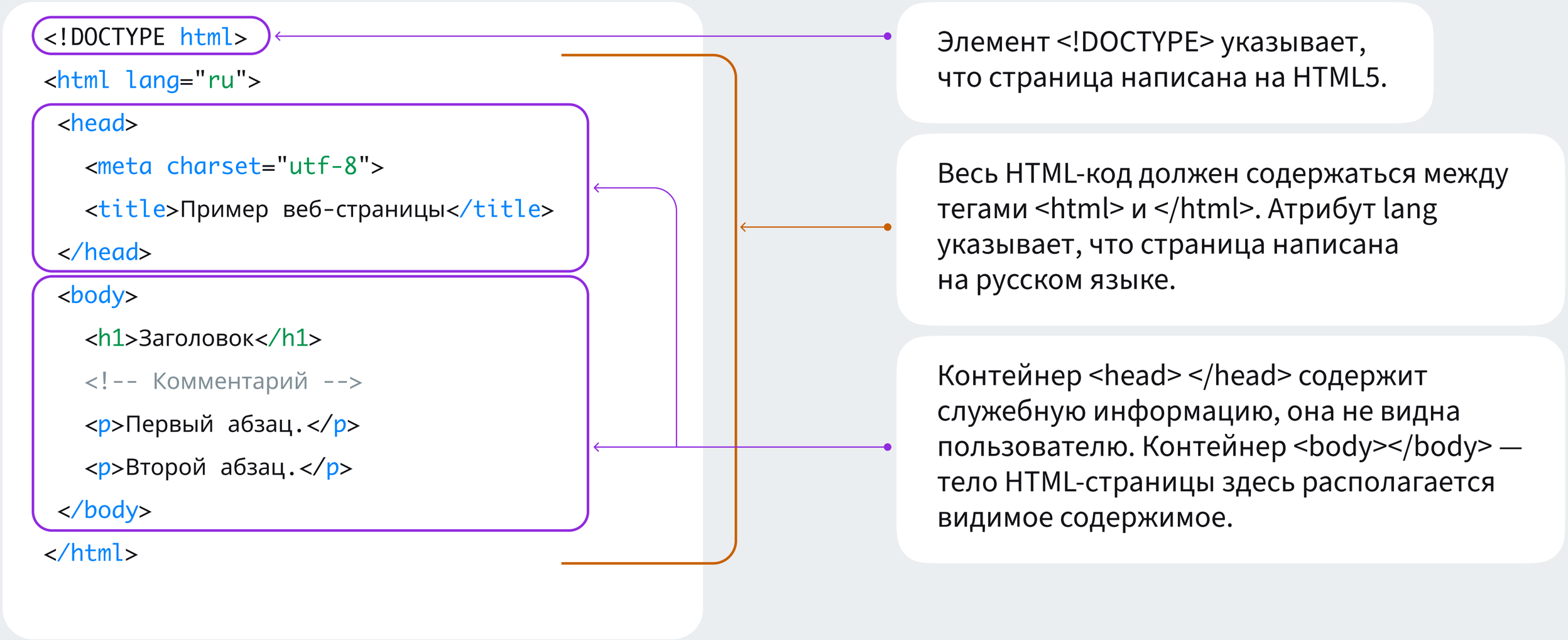
закрывающий тег не нужен.

```
<br>
```

У тегов есть **атрибуты** — с их помощью можно расширить возможности отдельных тегов. Например:

```
<a href="/news.html">Новости</a>
```

Общая структура HTML



Общая структура HTML

```
<!DOCTYPE html>
<html lang="ru">
 <head>
  <meta charset="utf-8">
  <title>Пример веб-страницы</title>
 </head>
 <body>
   <h1>Заголовок</h1>
   <!-- Комментарий -->
   <p>Первый абзац.</p>
   <p>Второй абзац.</p>
 </body>
</html>
```

С помощью тега <meta> можно изменять кодировку страницы, добавлять ключевые слова, описание документа и многое другое. Без указания кодировки браузер может неверно отображать русский текст.

Тег <title> определяет заголовок веб-страницы. Именно этот заголовок отображается во вкладке браузера.

Таким образом в HTML оформляются комментарии.

Заголовки и абзацы

```
<!DOCTYPE html>
<html lang="ru">
<head>
 <meta charset="utf-8">
 <title>Пример веб-страницы</title>
</head>
 <body>
  <h1>Заголовок 1</h1>
  <h2>Заголовок 2</h2>
  <h3>Заголовок 3</h3>
  <h4>Заголовок 4</h4>
  <h5>Заголовок 5</h5>
  <h6>Заголовок 6</h6>
  <p>Абзац</p>
 </body>
</html>
```

Теги <h1> … <h6> служат для создания заголовков разного уровня, от наиболее важного до наименее.

Тег <p> определяет абзац или параграф текста.

Ссылки

```
<!DOCTYPE html>
<html lang="ru">
<head>
 <meta charset="utf-8">
 <title>Пример веб-страницы</title>
</head>
 <body>
  <a href="https://uchi.ru/">Ссылка</a>
 </body>
</html>
```

Для создания ссылки используется тег <a>.

Атрибут href определяет адрес ссылки, а содержимое контейнера <a> является текстом ссылки.

Изображения

```
<!DOCTYPE html>
<html lang="ru">
<head>
 <meta charset="utf-8">
 <title>Пример веб-страницы</title>
</head>
 <body>
  <img src="images/cat.gif">
 </body>
</html>
```

Тег <img> используется для встраивания изображения на страницу.

Путь к изображению указывается с помощью атрибута src.

Списки

```
<!DOCTYPE html>
<html lang="ru">
<head>
 <meta charset="utf-8">
 <title>Пример веб-страницы</title>
</head>
 <body>
  <ul>
    <li>Красный</li>
    <li>Оранжевый</li>
    <li>Желтый</li>
  </ul>
  <ol>
    <li>Красный</li>
    <li>Оранжевый</li>
    <li>Желтый</li>
  </ol>
 </body>
</html>
```

Маркированный список создается с помощью тегов <ul> и , а нумерованный с помощью <ol> и .

Для создания элементов списка используются теги .

Оформление текста

```
<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="utf-8">
<title>Пример веб-страницы</title>
</head>
<body>
 <p>
   <b>Жирный</b><br>
   <i>Курсив</i><br>
   <u>Подчеркнутый</u><br>
 </p>
 </body>
</html>
```

Сделать текст жирным можно с помощью тегов <b>, выделить курсивом — тег <i>, подчеркнутым — тег <u>.

Переносы строк и последовательности из нескольких пробелов будут игнорироваться браузером, поэтому для принудительного переноса можно использовать тег
.

Блоки

Тег ** ** используется для того, чтобы сгруппировать элементы документа в структурные блоки, с целью изменения вида содержимого данного блока кода: положение, цвет, выравнивание и многое другое.

Свойства для блоков описываются с помощью CSS стилей. Без использования стилей блок никак не будет менять внешний вид.

Промежуточные итоги

- **Шаблон** — это HTML файл, который может содержать динамическое содержимое.
- Процесс, во время которого динамическое содержимое встраивается в страницу в HTML-страницу, называется **рендеринг** или **отрисовка.**
- Для рендеринга шаблонов используется функция **render_template().**
- **HTML** — это язык разметки, который определяет, как и какие элементы должны располагаться на странице, которую мы видим в браузере.

HTML теги

Структура HTML

<!DOCTYPE html>	Версия HTML
<html></html>	Содержимое HTML
<head></head>	Информация о странице
<body></body>	Содержимое страницы
<!-- -->	Комментарий

Дополнительная информация

<meta>	Метаданные
<title></title>	Заголовок страницы в браузере
<link>	Ссылка на внешний ресурс
<style></style>	Определение стилей
<script></script>	Описание скриптов

Основные теги

** ... ... **	Оформление заголовков
** **	Абзац текста
<a href=""><a>	Ссылка
<img src="">	Изображение
** или **	Списки (маркированный и нумерованный)
** **	Элемент списка
** **	Блоки

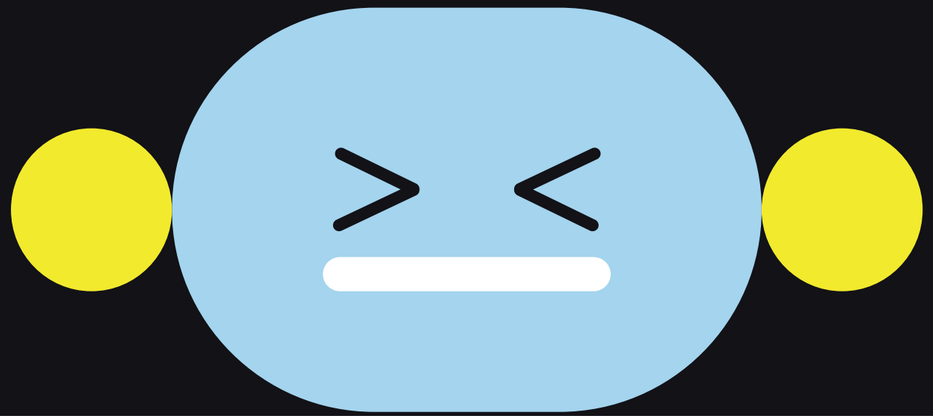
Оформление текста

****	жирный
****	курсив
<u></u>	зачеркнутый

Перерыв физкультминутка

Смотрим вверх–вниз, вправо–влево

Вращаем по кругу туда–обратно



Крепко зажмуриваемся

Быстро моргаем

Рассмотрим пример

```
<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="utf-8">
<title>Пример веб-страницы</title>
</head>
<body>
  <nav>
    <ul>
      <li>
        <a href="/">Главная</a>
      </li>
      <li>
        <a href="news/">Новости</a>
      </li>
    </ul>
  </nav>
</body>
</html>
```

Напишем небольшую HTML страницу и добавим ссылки для будущего меню: для этого воспользуемся маркированным списком ul, а в каждом пункте списка будет находиться ссылка.

Практика

- Вернемся к проекту flask_news. Скачайте шаблоны index.html и news_detail.html, а также файлы css и js и добавьте в свой проект.
- Доработайте шаблон news_detail.html, который возвращает функция представление связанная с маршрутом «/news_detail/<id>» для вывода переменных title и text. Значения этих переменных также возьмите из списка news.
- Шаблон index.html и функцию index дорабатывать пока не нужно!

Вопросы