Лекция 2. Глубокое понимание структуры HTML-документа и взаимодействие его элементов

Добро пожаловать, уважаемые студенты, на вторую лекцию нашего курса "Основы веб-разработки". Сегодня мы взглянем глубже в структуру HTML-документа и выясним, почему это так важно для веб-разработчика.

Структура – это каркас, на котором держится вся ваша веб-страница. Это не просто набор тегов; это организация вашего контента, предоставляющая ясную логику и порядок. Правильная структура не только облегчает вам жизнь в процессе разработки, но также делает вашу страницу доступной и понятной для браузеров, поисковых систем и, самое главное, для ваших пользователей.

Давайте начнем разбираться в том, как взаимодействуют элементы HTML и как создать структуру, которая не только будет функциональной, но и визуально привлекательной.

Поговорим о важном аспекте создания сложных и структурированных веб-страниц – вложенности тегов. Это концепция, которая придает вашему коду гибкость, позволяя создавать разнообразные макеты и компоненты.

Вложенность – это использование одних тегов внутри других. Это позволяет создавать древовидные структуры, где элементы могут быть вложены друг в друга, образуя логические блоки.

Давайте рассмотрим примеры. Представим, что у нас есть блок текста, и мы хотим выделить одно слово особым образом:

```
<p>Это <strong>очень</strong> важный текст.</p>
```

Здесь тег `<strong>` используется для выделения слова "очень" внутри абзаца. Это слово браузер выделит жирным шрифтом. Также рассмотрим вложенность с использованием `<div>`:

```
<div class="container">
  <h2>Заголовок блока</h2>
  <p>Текст внутри блока</p>
</div>
```

Здесь `<h2>` и `<p>` вложены в тег `<div>`, создавая логический блок с заголовком и текстом. Тег `<div>` означает “блок”. По-умолчанию он никак визуально не отображается браузером, текст в нем не выделяется особым шрифтом, не появляются отступы и фон. Но, забегая несколько вперед, все эти аспекты могут быть добавлены к

этому блоку за счет каскадных таблиц стилей - CSS, которые мы будет изучать подробнее через пару лекций. Атрибут class в блоке со значением container как раз предназначен для привязки определенных стилей к этому блоку.

В коде современных сайтов обычно существует множество тегов div, вложенных друг в друга, которые задают структуру документа - посмотрите, для примера, на фрагмент кода главной страницы сайта НГТУ:

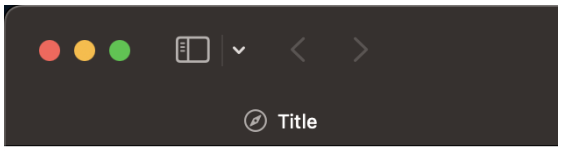
```
<div class="header__bottom">
    <div class="header__logo unhappy-new-year"><a href="/"><img
class="header__logo-full logo" src="/img/logo.png" alt="НЭТИ"
id="logo"><img class="header__logo-small logo_small" src="/img/logo-
mobile.png" alt="НЭТИ" id="logo_small"></a></div>
    <div class="header__menu">
        <div class="header__menu-wrapper">
            <nav class="header__menu-top">
                <div class="header__menu-item js-header-menu-item">
                    <a class="header__menu-item__link link-black "
href="/university"><span>Университет</span></a>
                        <div class="header__dropdown">
                            <div class="header__dropdown-wrapper">
                                <div class="container">
                                    <div class="header__dropdown-columns">
                                        <div class="header__dropdown-
group">
```

Переходим к еще одному стандартному инструменту для структурирования контента – спискам. Списки предоставляют удобный способ организации информации, делая ваш код более читаемым и вашу страницу более удобной для восприятия.

HTML предоставляет два основных тега для создания списков: `<ol>` (упорядоченный список) и `<ul>` (неупорядоченный список). Упорядоченные списки представляют собой элементы, которые упорядочены по порядку, а неупорядоченные списки – элементы, которые могут располагаться в произвольном порядке.

```
<!-- Пример упорядоченного списка -->
<ol>
  <li>Первый элемент</li>
  <li>Второй элемент</li>
  <li>Третий элемент</li>
</ol>

<!-- Пример неупорядоченного списка -->
<ul>
  <li>Первый элемент</li>
  <li>Второй элемент</li>
  <li>Третий элемент</li>
</ul>
```



1. Первый элемент
2. Второй элемент
3. Третий элемент

- Первый элемент
- Второй элемент
- Третий элемент

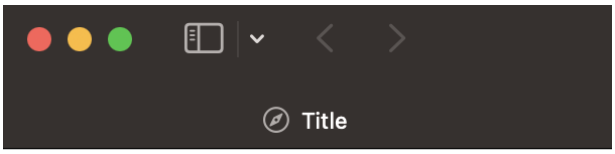
Элемент `<li>` (list item) используется для определения отдельных элементов списка. Этот тег следует писать внутри тегов `<ol>` или `<ul>`.

```
<ul>
  <li>Элемент 1</li>
  <li>Элемент 2</li>
  <li>Элемент 3</li>
</ul>
```

Атрибуты, такие как `type` для `<ol>` (который может быть `1`, `A`, `a`, `I`, `i`) и `start` (для начального значения упорядоченных списков), могут влиять на отображение списков, делая их более гибкими и подстраиваемыми под ваши потребности.

```
<ol type="A">
  <li>Первый элемент</li>
  <li>Второй элемент</li>
  <li>Третий элемент</li>
</ol>

<ol start="10">
  <li>Десятый элемент</li>
  <li>Одиннадцатый элемент</li>
  <li>Двенадцатый элемент</li>
</ol>
```



A. Первый элемент
B. Второй элемент
C. Третий элемент

10. Десятый элемент
11. Одиннадцатый элемент
12. Двенадцатый элемент

Переходим к еще одному увлекательному аспекту веб-разработки – работе с изображениями и ссылками. Вставка картинок и создание гиперссылок – это не только креативный процесс, но и важный элемент, делающий ваши страницы более интересными и функциональными. Это то, что отличает веб-документы от обычных текстовых, и то, из-за чего в названии языка HTML и присутствует в начале слово “Гипертекстовый”.

Тег `<img>` – это инструмент для добавления графики на веб-страницу. Просто укажите путь к файлу изображения в атрибуте `src`, и ваши посетители смогут наслаждаться визуальным контентом.

```
<img src="путь_к_изображению.jpg" alt="Описание изображения">
```

Атрибут `alt` также является обязательным и представляет собой текстовое описание изображения. Этот текст будет отображаться, если изображение не загрузится, и играет важную роль для людей с ограниченными возможностями и для поисковых систем.

Также у тега img есть необязательный атрибут title, который будет показываться пользователю, когда она наведет курсор мышки на изображение на веб-странице.

```
<img src="путь_к_изображению.jpg" alt="Красивый закат на пляже"
title="Нажмите для увеличения">
```

Тег `<a>` (anchor) предоставляет нам возможность создавать гиперссылки на другие веб-страницы, ресурсы или файлы. Укажите адрес в атрибуте `href`, и пользователи смогут перейти по этой ссылке.

```
<a href="https://www.example.com">Посетите наш веб-сайт</a>
```

Помимо ссылки на сайт в атрибуте href можно указать номер телефона или email (только нужно правильно использовать специальные префиксы tel и mailto) - тогда при клике на такую ссылку будет вызываться приложения для звонков или почтовый клиент, соответственно.

```
<a href="tel:+79991234567">+7 (999) 123-45-67</a>
<a href="mailto:info@example.com">info@example.com</a>
```

У ссылок могут быть также другие необязательные атрибуты, например, target и rel:

```
<a href="https://www.example.com" target="_blank" rel="noopener
noreferrer">Посетите официальный сайт</a>
```

Здесь мы использовали атрибут `target="_blank"`, чтобы открыть ссылку в новой вкладке, и добавили атрибуты `rel="noopener noreferrer"`, чтобы обеспечить безопасность при открытии в новой вкладке.

Когда мы вставляем изображения или создаем гиперссылки на веб-страницах, мы используем атрибуты src (для изображений) и href (для ссылок). Понимание различных типов путей к файлам важно для правильного отображения контента. Давайте поговорим подробнее о типах путей и их использовании.

Пути бывают относительными и абсолютными. Относительные пути начинается относительно текущего рабочего каталога (папки, в которой находится html-файл, код которого вы пишите). Если вы укажете, например, такой путь в теге img:

```
src="images/picture.jpg"
```

то браузер отобразит картинку из файла picture.jpg находящегося в папке images, которая находится в той же папке, что и файл с вашим кодом. И если вы, например, в ссылке в своем html-файле напишите:

```
href="detail.html"
```

то при клике на нее пользователем браузер откроет файл detail.html, лежащий в той же папке рядом с вашим файлом, в котором вы это написали.

Также есть возможность ссылаться на вышележащие папки, используя две точки, например, так:

```
src="../images/picture.jpg"
```

в этом случае браузер подключит файл picture.jpg из папки images, которая находится не внутри, а рядом с папкой, в которой лежит ваш html-файл:

```
about/
    your-file.html
images/
    picture.jpg
```

Абсолютные же пути к файлам обычно используются при размещении html-кода уже в интернете, на определенном домене. Для разработки на локальном компьютере и открытии файлов непосредственно с его диска они мало подходят. Абсолютный путь начинается со слеша, что означает, что файл и папка будет искаться от корня домена, например:

```
src="/images/picture.jpg"
```

В этом случае, если ваш html-код размещен на домене https://www.example.com, то браузером будет отображен файл https://www.example.com/images/picture.jpg независимо от того находится ли этот код в корне домена или где-то во внутренней папке. Указания путей вместе с протоколом и доменом также является абсолютными путями и это позволяет подключать изображения или ссылаться на страницы, находящиеся на других сайтах, например

```
src="https://www.another.com/images/picture.jpg"
```

Относительные пути удобны при работе с файлами в пределах проекта и хороши для поддержания порядка в структуре проекта.

Абсолютные пути обеспечивают стабильные ссылки, даже при изменении структуры папок, и используются, когда ресурсы расположены вне вашего проекта.

Практические советы:

1. Используйте относительные пути в пределах проекта:
    - Облегчит поддержание проекта и перенос на другие серверы.
2. Используйте абсолютные пути при внешних ресурсах:
    - Гарантирует правильную загрузку даже при изменении структуры проекта.
3. Старательно проверяйте регистр символов в путях:
    - На некоторых серверах и операционных системах регистр символов имеет значение.
4. Проверяйте правильность ссылок в браузере:
    - Открывайте консоль разработчика, чтобы увидеть ошибки загрузки файлов.

Наша лекция завершается. В ней мы освоили такие важные аспекты веб-разработки как вложенность тегов, работа с списками, изображениями и ссылками. Поняли

разницу между относительными и абсолютными путями для корректной работы с файлами. Теперь, вооружившись этими знаниями, важно практиковаться. Вперед, выполняйте практические задания, экспериментируйте и создавайте свои первые веб-проекты. Успехов в вашем творчестве!

Дополнительный материал (не включается в видеолекцию, идет только в файле)

Блочные и строчные элементы HTML:

Важным аспектом при работе с HTML является понимание разницы между блочными и строчными элементами. Эта классификация определяет, как элемент взаимодействует с другими элементами на странице и как он оформляется визуально.

Блочные элементы:

1. Характеристики:
   - Занимают всю доступную ширину, по умолчанию начинают новую строку.
   - Примеры: `<div>`, `<p>`, `<h1>-<h6>`, `<ul>`, `<ol>`, `<li>`.
2. Использование:
   - Идеальны для создания структуры веб-страницы и группировки содержимого.
   - Позволяют задавать ширину, высоту, внутренние и внешние отступы.

Строчные элементы:

1. Характеристики:
   - Занимают только необходимую ширину, не начинают новую строку.
   - Примеры: `<span>`, `<a>`, `<strong>`, `<em>`, `<img>`.
2. Использование:
   - Подходят для внедрения элементов в текст, изменения стиля части текста.
   - Не принимают значения ширины и высоты, не создают новую строку.

Как это влияет на веб-разработку:

Позиционирование:

Блочные элементы часто используются для создания структуры страницы, в то время как строчные элементы чаще применяются для встроенных элементов.

Визуальное оформление:

Блочные элементы могут содержать другие блочные и строчные элементы, что дает больше возможностей для стилизации.

Это важное понятие помогает разработчикам более точно управлять структурой и визуальным представлением элементов на веб-страницах. Знание различий между блочными и строчными элементами улучшит вашу способность создавать гибкий и красочный контент.