**Практическое занятие № 7**

**ТЕМА: Базовые правила создания веб-страницы. Элементы блочной верстки: CSS свойства display, float и position**

|  |  |  |
| --- | --- | --- |
| **7.1. Теоретический материал** | | |
| **Некоторые рекомендуемые элементы HTML документа**  Начинать HTML документ рекомендуется с объявления <!DOCTYPE html>. Это не тег, а обязательная инструкция объявления типа документа. Она нужна для того, чтобы сообщить браузеру в какой версии HTML написан документ. По доктайпу браузер определяет версию HTML и правильно отображает страницу. Объявление типа документа должно быть самым первым что видит браузер во время обработки HTML-документа. Современные страницы содержат <!DOCTYPE html>, дополнительные параметры указываются, если html-разметка соответствует устаревшим стандартам.  Далее рекомендуется начинать описание разметки страницы с тега **<html>,** дополненного атрибутом **lang.** Атрибут **lang** указывает на каком языке написан текст страницы.    Внутри секции <head> могут прописываться метаданные, т.е. информация предназначенная для браузера и поисковых систем: установка кодировки документа, передача информации поисковым системам и многое другое. Для указания метаданных используется тег **<meta>** Мета-тегов может быть несколько, потому что в зависимости от используемых атрибутов они несут различную информацию. Например, кодировку страницы необходимо указать для того, чтобы браузер корректно отобразил текст. Если этого не сделать, или задать неверную кодировку, вместо символов браузер может отобразить иероглифы. Чаще всего пишут следующее:  <meta charset="utf-8" />  Краткое описание содержания с помощью мета-тегов помогает поисковым системам лучше проиндексировать страницу:  <meta name="description" content="Изучение основ HTML5 для новичков" />  Подробнее о мета-тегах можно прочитать, например, здесь: <https://developers.google.com/search/docs/crawling-indexing/special-tags?visit_id=638096312071276814-3988895965&rd=1>  **Каркас HTML-страницы. Поток документа.**  HTML-документ состоит из «дерева» тегов. Пример простейшего дерева представлен на рисунке:    Когда мы открываем веб-страницу в браузере, он получает исходный текст HTML и разбирает (парсит) его для того, чтобы сформировать подобную древовидную (иерархическую) структуру веб-документа. Более того, браузер строит специальную модель структуры документа (в основе которой лежит дерево) и использует её, чтобы нарисовать страницу на экране. Это представление называется Document Object Model (объектная модель документа), или сокращённо DOM. Именно благодаря DOM-модели мы можем работать с элементами страницы через JavaScript код. (Мы можем получить доступ к этим объектам через глобальную переменную document).  В этой модели у элементов есть родительские и дочерние элементы. Например, на рисунке выше элемент **body** является родительским элементом для элементов **h1, p, ul**. Вместе с тем элемент **body** является дочерним для **html**, а элемент **p** родительским для **a**.  На рисунках ниже показан фрагмент HTML-кода и построенная на его основе DOM-модель:    Таким образом, DOM-модель (дерево) отображает структуру документа, а порядок отображения элементов на странице определяется **потоком документа**.  **Поток** — одно из важнейших базовых понятий в создании веб-интерфейсов (вёрске). Это принцип организации элементов на странице при отсутствии стилей: если мы напишем HTML и не напишем CSS, то отображение в браузере будет предсказуемо благодаря тому, что браузер однозначным образом располагает элементы в потоке. На самом деле, если мы не пропишем к элементам страницы стилевые правила, то к каждому элементу всё равно будут применяться CSS-правила – правила «по умолчанию», «зашитые» в браузер. Благодаря этим правилам «по умолчанию» заголовок **<h1>** крупнее заголовка **<h2>**, а гиперссылки — синие и подчёркнутые. Если вообще не применять никаких стилей, браузер формирует из элементов **нормальный поток.**  Поток — это вертикальный и горизонтальный порядок отображения элементов на странице. Вертикально поток идёт сверху вниз, и, по умолчанию, элементы отображаются на странице в том порядке, в котором они указаны в HTML-документе. Горизонтально поток идёт слева направо.  Все элементы страницы, без исключения, это прямоугольные области, занимающие определённое место в так называемых «строках», как слова предложения на листе в линейку. Есть два основных типа элементов - блочные и строчные. При этом расположение элементов в условиях нормального потока как раз и зависит от того, к какому из двух типов относится элемент – блочному (block) или строчному (inline).    **Блочный элемент** - занимает всю строку целиком (если css свойством width искусственно не ограничивается ширина элементов), вне зависимости от объёма его контента, поэтому несколько блочных элементов визуально идут друг за другом сверху вниз.  К блочным элементам относятся, например, теги <div>, <fieldset>, <form>, <h1>,...,<h6>, <hr>, <ol>, <p>, <pre>, <table>, <ul> и др.  Не все блочные теги допустимо вкладывать один в другой, поэтому при создании структуры кода активную роль выполняет **<div>** как универсальный кирпичик вёрстки. Тег **<div>** допустимо вкладывать один в другой, другие блочные элементы также можно помещать внутрь **<div>.** Существуют и некоторые другие правила:   * Блоки располагаются по вертикали друг под другом. * На прилегающих сторонах элементов действует эффект схлопывания отступов. * Запрещено вставлять блочный элемент внутрь строчного. Например, **<a><h1>**Заголовок**</h1></a>** не пройдёт валидацию, правильно вложить теги наоборот — **<h1><a>**Заголовок**</a></h1>**. * По ширине блочные элементы занимают всё допустимое пространство. * Если задана ширина контента (свойство width), то ширина блока складывается из значений width, полей, границ, отступов слева и справа. * Высота блочного элемента вычисляется браузером автоматически, исходя из содержимого блока. * Если задана высота контента (свойство height), то высота блока складывается из значения height, полей, границ, отступов сверху и снизу. При превышении указанной высоты контент отображается поверх блока. * На блочные элементы не действуют свойства, предназначенные для строчных элементов, вроде vertical-align. * Текст по умолчанию выравнивается по левому краю.   **Строчный элемент** (inline element) - занимает место по своему содержимому, поэтому несколько строчных элементов могут располагаться на одной строке. Когда в строке не хватает места для вмещения строчного элемента, он переносится на новую. К строчным элементам относятся теги <img>, <span>, <a> и др. В основном они используются для изменения вида текста или его логического выделения.  По аналогии с блочными элементами перечислим их характерные особенности.   * Внутрь строчных элементов допустимо помещать текст или другие строчные элементы. Вставлять блочные элементы внутрь строчных запрещено. * Эффект схлопывания отступов не действует. * Свойства, связанные с размерами (width, height) не применимы. * Ширина равна содержимому плюс значения отступов, полей и границ. * Несколько строчных элементов идущих подряд располагаются на одной строке и переносятся на другую строку при необходимости. * Можно выравнивать по вертикали с помощью свойства vertical-align.   Строчные элементы удобно использовать для изменения вида и стиля текста, в частности, отдельных символов и слов. Для этой цели обычно применяется универсальный тег **<span>**, который самостоятельно никак не модифицирует содержимое, но легко объединяется со стилями через классы или идентификаторы. За счёт чего с помощью этого тега можно легко управлять видом и положением отдельных фрагментов текста или рисунков.  Для вёрстки строчные элементы применяются реже, чем блочные. Это связано в основном с тем, что внутрь строчных элементов не допускается вкладывать контейнеры **<div>**, **<p>** и подобные широко распространённые теги. Тем не менее, блочные и строчные элементы удачно дополняют друг друга, поскольку позволяют на всех уровнях менять вид составляющих веб-страниц.  **CSS свойство display**  За восприятие браузером элементов как блочных или строчных отвечает CSS-свойство display. При этом у блочных по умолчанию элементов это свойство автоматически установлено в значение block, а у строчных – в значение inline. При этом у разработчиков остается возможность в ручном режиме настроить свойство display, сделав элемент блочным или строчным (существуют и другие значения CSS свойства display). Наиболее часто применяемые значения свойства display показаны в таблице ниже:   |  |  | | --- | --- | | block | Элемент показывается как блочный. Применение этого значения для встроенных элементов, например тега **<span>**, заставляет его вести подобно блокам — происходит перенос строк в начале и в конце содержимого. | | inline | Элемент отображается как встроенный. Использование блочных тегов, таких как **<div>** и **<p>**, автоматически создает перенос и показывает содержимое этих тегов с новой строки. Значение inline отменяет эту особенность, поэтому содержимое блочных элементов начинается с того места, где окончился предыдущий элемент. | | inline-block | Это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно встроенному элементу. Фактически такой элемент по своему действию похож на встраиваемые элементы (вроде тега **<img>**). При этом его внутренняя часть форматируется как блочный элемент, а сам элемент — как встроенный. | | inline-table | Определяет, что элемент является таблицей как при использовании тега **<table>**, но при этом таблица является встроенным элементом и происходит ее обтекание другими элементами, например, текстом. | | list-item | Элемент выводится как блочный и добавляется маркер списка. | | none | Временно удаляет элемент из документа. Занимаемое им место не резервируется и веб-страница формируется так, словно элемента и не было. Изменить значение и сделать вновь видимым элемент можно с помощью скриптов, обращаясь к свойствам через объектную модель. В этом случае происходит переформатирование данных на странице с учетом вновь добавленного элемента. |     Отдельного рассмотрения заслуживает значение **inline-block,** которое превращает элемент в гибрид строчного и блочного элементов. Если для формирования секций использовать тег <div>, как блочный элемент он будет каждый раз начинаться с новой строки. Для строчных элементов нельзя задать цвет фона всей секции и установить её размеры. Чтобы наделить элемент одновременно строчными и блочными свойствами, его можно превратить в строчно-блочный, который будет сочетать преимущества как строчных, так и блочных элементов.В HTML нет тега, который относится к строчно-блочным элементам, его можно определить, задав элементу свойство display со значением inline-block.    Характеристики этих элементов следующие:   * Внутрь строчно-блочных элементов допустимо помещать текст, строчные или блочные элементы. * Высота элемента вычисляется браузером автоматически, исходя из содержимого блока. * Ширина равна содержимому плюс значения отступов, полей и границ. * Несколько элементов идущих подряд располагаются на одной строке и переносятся на другую строку при необходимости. * Можно выравнивать по вертикали с помощью свойства vertical-align. * Разрешено задавать ширину и высоту. * Эффект схлопывания отступов не действует.   *В примерах 7.1 и 7.2 будут демонстрироваться строчно-блочные элементы.*  Кроме того, существуют значения flex и grid, которые подробно будут рассматриваться в курсе «Программирование фронтенда и бэкенда».  *В примере 7.1 демонстрируется, как можно поставить несколько элементов <div> в строку путем изменения стандартного значения свойства display.*  В примере показывается, как можно показать кнопочки соцсетей со ссылками на аккаунты. Обратите внимание, что внутри ссылок ничего не написано. Нам не нужно выводить название соцсети, а нужно вывести иконку с логотипом. Это сделано при помощи настройки фона элемента списка. При этом пункты списка (элементы <li>) с помощью ручной настройки CSS (класс .social) выстраиваются в ряд, а не друг под другом. Далее гиперссылки превращаются из строчных в блочные элементы (класс .social\_link), что позволяет задать высоту и ширину кнопочек соцсетей. Итак, по умолчанию ссылки — строчные. Это значит, что им нельзя задать размеры (width и height) и фоновую картинку. Пишем display: block, и строка превращается в условный прямоугольник, у которого могут быть и размеры, и фон. После этого мы можем менять размер на нужный нам и фоном выводить кнопочки каждой из соцсетей (используются изображения, которые хранятся в специальных файлах).  Помимо этого, по ходу решения задачи мы установили свойство display: inline-block для пунктов списка с классом social. За счёт этого элементы, которые по умолчанию блочные, приобретают внешние признаки строчных элементов. Вместо того чтобы выстраиваться друг под другом, пункты списка теперь стоят рядом, в строку.  *В примере 7.2 демонстрируется создание галереи изображений на основе строчных и блочных элементов*  **Float элементы (плавающие элементы)**  Некоторые элементы в CSS являются блочными и поэтому начинаются с новой строки. Например, если расположить два абзаца с тегами <P>, то они будут находиться друг под другом. Другие же элементы являются «строчными», т.е. отображаются на странице в одну строку. Один из способов переназначить элементам тип обтекания — это использование свойства **float**. Классический пример — использование **float** для выравнивания картинки по левому или правому краю. Вот простой HTML-код картинки и абзаца:    Они отображаются с новой строки, как и предполагает нормальный поток. Но нормальный поток можно «поломать», настроив CSS-свойство float у изображения:    Получается выравнивание по правому краю, а абзац обтекает картинку:    Свойство **float** может иметь следующие значения:    Элементы, которые идут после плавающего элемента, будут обтекать его. Свойство **clear** определяет, с каких сторон элемента другие плавающие элементы не допускаются.    **Позиционирование**  CSS свойство **position** задаёт способ позиционирования элемента в документе. Совместно с CSS-свойствами **left, right, top, bottom** элементу задаётся его местоположение на странице.  Иногда в процессе вёрстки требуется реализовать разные сложные идеи дизайнера про расположение элементов друг относительно друга. Например, расположить один элемент поверх другого или немного сместить отображение элемента относительно своего начального положения. Бывают и более сложные случаи, когда требуется зафиксировать элемент относительно окна браузера, а не относительно страницы. Базовым свойством, которое изменяет способ позиционирования, является свойство **position**.  **position: static**  Значение по умолчанию. Статичное позиционирование. Любой элемент, добавленный на страницу, будет иметь это значение и будет расположен в нормальном потоке документа  **position: relative**  Элемент позиционируется так же, как и статичный, но его отображение может быть смещено относительно своего начального положения при помощи CSS-свойств **left, right, top, bottom.** Это смещение чисто **визуальное** и не затрагивает положение соседних элементов. На странице элемент будет занимать столько же места, как если бы он имел статичное позиционирование.  **position: absolute**  При задании абсолютного позиционирования с помощью свойств left, right, top, bottom регулируется положение элемента. Есть ряд особенностей такого позиционирования:   * Элемент убирается из основного потока документа. То есть перестаёт влиять на положение окружающих элементов и на размер родителя. Можно представить себе, что элемент уходит на слой выше и перестаёт взаимодействовать со всеми элементами, кроме своих потомков. * Если элемент был частью строчного контекста форматирования, он приобретает блочный контекст форматирования. К нему становится применима блочная модель. * Если элемент был блочным и занимал всю ширину своего родителя, то теперь его ширина будет определяться шириной контента. * Отступы элемента с position: absolute не схлопываются с отступами соседних элементов.     **position: fixed**  Иногда требуется позиционировать элемент не относительно родителя, а относительно окна браузера вне зависимости от вложенности. Для решения подобной задачи подходит **position: fixed**. Свойство так же, как и предыдущее, работает с указанием смещения **left, right, top, bottom**.  Элемент убирается из основного потока документа. То есть перестаёт влиять на положение окружающих элементов и на размер родителя. Можно представить себе, что элемент уходит на слой выше и перестаёт взаимодействовать со всеми элементами, кроме своих потомков.  **position: sticky (липкое позиционирование)**  Элемент позиционируется в нормальном потоке так же, как и статичный, а затем смещается при помощи свойств left, right, top, bottom относительно ближайшего родителя, имеющего прокрутку. Это свойство применяется, когда нам нужно зафиксировать какой-либо элемент не сразу, а при прокрутке родителя до какого-то известного положения.  Например, в следующем примере элемент будет вести себя как обычно до тех пор, пока родитель не будет прокручен таким образом, что расстояние от верха родителя до верхней границы элемента не станет меньше 10 пикселей. Как только прокрутка достигнет такого значения, элемент зафиксируется в положении 10 пикселей от верха границы родителя:    *Использование «прилипшего элемента» продемонстрировано в примере 7.3* | | |
| **Пример 7.1** | | |
|  | ***Задача:*** | |
|  | Сформировать строку с иконками социальных сетей и гиперссылками на аккаунты |
| ***Решение:*** | |
|  | Файл index.html    Файл style.css |
| ***Ответ:*** | |
|  |  |
| **Пример 7.2.** | | |
|  | ***Задача:*** | |
|  | Создать галерею изображений из блочно-строчных элементов (исходные изображения в примере имеют одинаковый размер) |
| ***Решение:*** | |
|  | Файл index.html    Файл style.css |
| ***Ответ:*** | |
|  |  |
| **Пример 7.3.** | | |
|  | ***Задача:*** | |
|  | **Повтор** |
| ***Решение:*** | |
|  | Файл index.html    Файл style.css |
| ***Ответ:*** | |
|  |  |
| **Задание 7.1** | | |
|  | ***Задача:*** | |
|  | Создать галерею изображений известных ученых (например, химиков или экономистов) на основе блочно-строчных элементов. Настроить собственный стиль для галереи |
| ***Решение:*** | |
|  | Файл index.html    Файл style.css |
| ***Ответ:*** | |
|  |  |
| **Задание 7.2** | | |
|  | ***Задача:*** | |
|  | Повторить по образцу:    Текст и изображение взяты здесь: <https://www.mirea.ru/about/history-of-the-university/history-of-the-university/>  *Обтекание настроить с помощью свойства* ***float*** *для тега* ***<img>.***  *Отступы вокруг изображения настраиваются с помощью CSS свойства* ***margin.*** *Выравнивание текста по ширине для тега* ***<p>*** *в CSS настраивается так:*  ***text-align: justify;*** |
| ***Решение:*** | |
|  | Файл index.html    Файл style.css |
| ***Ответ:*** | |
|  |  |
| **Задание 7.3** | | |
|  | ***Задача:*** | |
|  | На основе примера 7.3 создайте страницу с «прилипшем элементом» про одного из ученых, помещенных в галерею в задании 7.1 |
| ***Решение:*** | |
|  | Файл index.html    Файл style.css |
| ***Ответ:*** | |
|  |  |