Основы языка оформления стилей документа CSS. Формирование блочной модели.

CSS (Cascading Style Sheets) — это каскадные листы стилей, которые применяются для описания внешнего вида веб-документа, написанного при помощи языка разметки HTML.

Другими словами, с помощью HTML появляется структура документа, а CSS — это уже его оформление. При помощи CSS можно менять цвета, шрифты у текста, изменять положение элементов на странице, их размеры, задавать элементам рамки, границы и отступы.

Синтаксис CSS

```
селектор {
   свойство: значение;
   свойство2: значение2;
}
```

Выше представлен самый популярный и удобный способ написания стилей. В нём можно увидеть наглядность и одновременно с этим, компактность. Поэтому его выбирает большая часть программистов.

Комментарии в CSS

```
        /* Внешний вид комментария */
p {
color: blue;
}
/*
Стили
для
параграфа
*/
```

В CSS можно оставлять комментарии для описания свойств элементов или для комментирования самих стилей при редактировании документа.

Примечание: если вы хотите быстро закомментировать несколько строк css, нужно выделить их и нажать ctrl + /.

Чтобы использовать стили CSS в веб-документе, нужно их сначала подключить. Рекомендуется использовать Внешний CSS-файл. Создаём файл с расширением .css.

style.css

```
body {
   background: #0f0;
}
h1 {
   text-align: center;
   color: blue;
}
```

В нужном HTML-файле этот файл подключаем, указав на него ссылку в теге <link>.

index.html

```
<head>
   <link rel="stylesheet" href="style.css">
</head>
```

Для подключения CSS-файла используется тег link, который помещается в раздел head нужного HTML-файла. Чтобы правильно подключить файл стилей, у тега link нужно указать несколько атрибутов.

В атрибуте rel указывается значение stylesheet, то есть лист стилей. Это нужно, чтобы браузер понимал, что подключается файл стилей CSS. В атрибуте href указывается путь к CSS-файлу. Причём, как и в случае с гиперссылками, этот путь может быть относительным и абсолютным.

Селекторы в CSS

1. Селекторы тегов

html	css

Для всех заголовков первого уровня цвет текста будет синим	h1 {
	color: blue;
	}

При использовании селекторов тегов стиль будет применяться ко всем указанным тегам. В качестве селектора указывается название любого HTML-тега.

Этот способ обращения к html используется крайне редко. Вы никогда не знаете, когда ещё потребуется создать заголовок или параграф. Например, вы создали стили для всего проекта. В дальнейшем большая часть заголовков будет в другой стилистике. В этом случае придётся каждый раз менять стили, которые вы уже задали заранее. Поэтому рассмотрим другие способы.

2. Селекторы классов (class)

html	css

<h1 class=”border”>Заголовок с рамкой</h1>
 Параграф с рамкой
	.border {
	border: 1px solid black;
	}

Для задания классу некоторого стиля необходимо указать точку его (класса) название. Этот способ используется чаще всего, так как не обладает недостатками.

3. Селекторы атрибутов

В качестве селекторов можно указывать атрибуты HTML-тегов. Есть разные способы указывать селекторы атрибутов.

html	css

<img src="pic.jpg" alt="Фото">
<input type="text">
	img[alt] {
	  width: 100px;
	}
	input[type="text"] {
	  font-size: 10px;
	}

В этом примере сначала указывается стиль для всех картинок, у которых присутствует атрибут alt. Для этого название атрибута выдаётся в квадратных скобках, сразу после названия тега.

Второй пример. Стиль будет применяться для всех тегов <input>, в значении атрибута type которого присутствует значение text, то есть для всех обычных текстовых полей ввода.

Свойства стилей CSS

Ширина и высота: width и height

```
img {
    height: 200px;
    width: 300px;
}
```

Можно задавать ширину и высоту в любых единицах измерения CSS (px, %). Если содержимое блока превышает указанную высоту, высота элемента останется неизменной, а содержимое будет отображаться поверх него. Например, width="100%" означает, что рисунок будет растянут на всю ширину веб-страницы. Примеры некоторых селекторов для задания *фона элемента – background:*

```
background-color: #ff0;
background-image: url(img/photo.jpg);
background-position: top; (bottom | left | right)
background-repeat: repeat-x; (repeat-y | no-repeat)
background-size: cover;
```

- background-color задаёт цвет фона.
- background-image используется, чтобы в качестве фона можно было установить изображение.
- background-position указывает, где будет располагаться фоновое изображение. Может иметь значения: top, bottom, left, right.

- background-repeat определяет, нужно ли повторять фоновое изображение: repeat-x — изображение повторяется по горизонтали, repeat-y — по вертикали, no-repeat — изображение не повторяется. По умолчанию у этого свойства установлено значение repeat. Это означает, что изображение будет повторяться по горизонтали и по вертикали.
- background-size — размер изображения. Возможно, вы не хотите, чтобы изображение занимало стандартные параметры. Поэтому можно выставить процентное значение или универсальное cover, которое будет занимать всё доступное пространство, contain — доступное пространство по высоте или ширине, чтобы фоновое изображение поместилось полностью.

border — рамка вокруг элемента

```
border-color: red; (#f00 | RGB(255, 0, 0))
border-style: solid; (dotted | dashed | groove | ridge | solid | double | inset | outset)
border-width: 2px;
```

- border-color — цвет рамки.
- border-style — стиль рамки, которая, может быть, разных значений: dotted, dashed, solid, double, groove, ridge, inset, outset.
- border-width задаёт толщину рамки, причём её можно задать для каждой из 4 сторон отдельно: (1px 2px) — 1px: верхняя и нижняя, 2px: левая и правая; (1px 2px 3px) — 1px: верхняя, 2px: левая и правая, 3 нижняя; (1px 2px 3px 4px) — 1px: верхняя, 2px: правая, 3px: нижняя, 4px: левая.

Можно перечислять свойства в одну строку, разделяя их пробелом. В этом случае не важен порядок следования свойств.

```
border: 1px solid black;
```

Цвет текста — color

```
color: red;
color: #78fa2e;
```

Шрифт — font

```
font-style: italic; (oblique | normal)
font-variant: small-caps;
font-weight: bold; (bolder | lighter| 100 | 200);
font-size: 20px; (small | medium | large);
```

- font-style — стиль шрифта. Предустановлен шрифт в значении normal. italic — это курсивное начертание, которое имитирует рукописный текст. oblique — наклонное начертание.
- font-variant имеет только 2 значения. По умолчанию установлено значение normal и small-caps, которое у строчных букв имитирует заглавные буквы, только уменьшенного размера.
- font-weight задаёт насыщенность шрифта. Можно указывать значения предопределёнными словами, например, bold — полужирный, bolder — жирный, lighter — светлый. Насыщенность определяется цифрами от 100 до 900.
- font-size определяет размер шрифта. Можно указывать в любых единицах измерения или предопределёнными словами. Определять стиль шрифта можно сокращённой записью. В этом случае важен порядок следования значений.

Оформление списков – list-style

```
list-style-type: circle; (disc | square | armenian | decimal)
list-style-position: inside;
list-style-image: url(img/list.png);
```

- list-style-type — тип маркера, который может быть, у разных видов. В примере приводятся только некоторые из них.
- list-style-position определяет, где располагается маркер. По умолчанию у него значение outside. В этом случае маркеры будут располагаться за пределами текстового блока. При значении inside — наоборот, внутри текстовых блоков.
- list-style-image позволяет вместо маркера установить изображение, для этого нужно указать к нему путь в скобках url.

Работа с текстом

```
text-align: center; (justify | left | right)
text-decoration: none; (line-through | overline | underline | none)
text-transform: capitalize; (lowercase | uppercase)
```

- text-align — выравнивание содержимого блока по горизонтали. Принимает 4 значения: left, right, center и justify. Выравнивание происходит по ширине, то есть одновременно по левому и по правому краю.
- text-decoration применяется для следующего оформления текста: line-through — перечёркивает текст, overline — задаёт линию над текстом, underline — под текстом (подчёркивает текст), none (по умолчанию) — отменяет все эффекты.

- text-transform используется для изменения регистра символов. capitalize — каждое слово в предложении будет начинаться с заглавной буквы. При значении lowercase все символы будут строчными, а при uppercase — заглавными.

Псевдоклассы

С псевдоклассами добавляются особые классы к элементам. Вы наверняка замечали на сайтах: когда наводите мышкой на конкретный пункт меню, он меняет свой вид. У него изменяется цвет фона, цвет ссылки, даже шрифт или его размер. Это происходит благодаря псевдоклассам. Рассмотрим их синтаксис.

```
Селектор:псевдокласс {
    свойство1: значение1;
}
a:hover {
   color: #ccc;
}
```

После селектора ставится двоеточие. Сразу после него без пробела указывается название псевдокласса.

Псевдоклассы, определяющие состояние элементов

1) a:link — ссылается на непосещённую ссылку.
2) a:visited — ссылается на уже посещённую ссылку.
3) a:hover — ссылается на любой элемент, по которому проводят курсором мыши.
4) a:focus — ссылается на любой элемент, над которым находится курсор мыши.
5) a:active — ссылается на активизированный пользователем элемент.

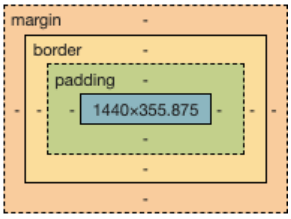
Основы позиционирования

В вёрстке сайта с помощью слоёв самый часто используемый HTML-тег — . Он и формирует слой на веб-странице. Это блочный тег. Сам по себе тег ничего на экране не отображает и оформляется стилями CSS.

```
<div>Блочный элемент</div>
```

Блочные элементы содержат элементы любого типа.

Ширина блока складывается из свойств	Высота блока складывается из свойств
margin-left margin-right padding-right padding-left width	margin-top margin-bottom padding-top padding-bottom height



Внутренний отступ или поле элемента (padding) добавляет отступы внутри элемента, между его основным содержимым и границей. Внешний отступ (margin) добавляет отступы за границами элемента, создавая тем самым промежутки между элементами. Значения padding и margin задаются в следующем порядке: верхнее, правое, нижнее и левое. Внешние, внутренние отступы и рамка элемента — необязательные свойства. По умолчанию их значение равно нулю. Но некоторые браузеры добавляют к ним положительные значения автоматически на основе собственных таблиц стилей.

Адаптивная вёрстка

Свойства flex-контейнера

Обязательно прописываем display: flex;

1. justify-content

Свойство выравнивает flex-элементы по ширине flex-контейнера, распределяя оставшееся свободное пространство. Для выравнивания элементов по вертикали используется свойство align-content.

- flex-start – значение по умолчанию. Flex-элементы позиционируются от начала flex-контейнера.
- flex-end – flex-элементы позиционируются относительно правой границы flex-контейнера.
- center – flex-элементы выравниваются по центру flex-контейнера.
- space-between – flex-элементы выравниваются по главной оси, свободное место между ними распределяется следующим образом: первый блок располагается в начале flex-контейнера, последний блок – в конце, все остальные блоки равномерно распределены в оставшемся пространстве, а свободное пространство равномерно распределяется между элементами.
- space-around – flex-элементы выравниваются по главной оси, а свободное место делится поровну, добавляя отступы справа и слева.

2. align-items

Свойство выравнивает flex-элементы, в том числе анонимные flex-элементы, по перпендикулярной оси (по высоте).

- stretch – значение по умолчанию. Flex-элементы растягиваются, занимая все пространство по высоте.
- flex-start – flex-элементы выравниваются по левому краю flex-контейнера относительно верхнего края блока-контейнера.

- flex-end – flex-элементы выравниваются по левому краю flex-контейнера относительно нижнего края блока-контейнера.
- center – flex-элементы выравниваются по центру flex-контейнера.
- baseline – flex-элементы выравниваются по базовой линии.

3. flex-direction

Свойство определяет, каким образом flex-элементы укладываются во flex-контейнере, задавая направление главной оси flex-контейнера. Они могут располагаться в двух главных направлениях: горизонтально, как строки, и вертикально, как колонки. Главная ось по умолчанию идет слева направо. Поперечная – сверху вниз.

- row – значение по умолчанию, слева направо (в rtl – справа налево). Flex-элементы выкладываются в строку. Начало (main-start) и конец (main-end) направления главной оси соответствуют началу (inline-start) и концу (inline-end) инлайн оси (inline-axis).
- row-reverse – направление справа налево. Flex-элементы выкладываются в строку относительно правого края контейнера.
- column – направление сверху вниз. Flex-элементы выкладываются в колонку.
- column-reverse – колонка с элементами в обратном порядке, снизу вверх.

4. flex-wrap

Свойство управляет тем, как flex-контейнер будет выкладывать flex-элементы – в одну строку или в несколько, и направлением, в котором будут укладываться новые строки. По умолчанию flex-элементы укладываются в одну строку. При переполнении контейнера их содержимое будет выходить за границы flex-элементов.

- nowrap – значение по умолчанию. Flex-элементы не переносятся, а располагаются в одну линию слева направо (в rtl – справа налево).
- wrap – flex-элементы переносятся, располагаясь в несколько горизонтальных рядов (если не помещаются в один ряд) в направлении слева направо (в rtl справа налево).
- wrap-reverse – flex-элементы переносятся, располагаясь в обратном порядке слева направо, при этом перенос происходит снизу вверх.

Задание

1. Создать файл стилей и подключить к каждой странице.
2. Задать блочную структуру страниц сайта:

```
<body>
<div class="wrapper">
        <div class="content">
```

```
                    <div class="menu"> //главное меню сайта
                    </div>
                //контент страницы
                </div>
                <div class="footer"> //подвал
                </div>
    </div>
    </body>
```

3. Прижать «подвал» к низу страницы, задав следующие стили:

```
/*Прижатие футера*/
* {
        margin: 0;
        padding: 0;
}

html, body {
        height: 100%;
}

.wrapper {
        display: flex;
        flex-direction: column;
        min-height: 100%;
}

.content {
        flex: 1 0 auto;
}

.footer {
        flex: 0 0 auto;
}
```

4. Для всего сайта задать разновидность шрифта:
    a. font-family: sans-serif.
5. Расположить горизонтально меню сайта и убрать маркеры. При наведении на каждый пункт меню должна появляться подсветка текста.
6. Для всех гиперссылок убрать подчеркивающую линию и задать любой цвет текста.
7. Для всех параграфов присвоить класс и задать стили:
    a. font-size: 16px;
    b. line-height: 26px;
    c. color: #1F3F68.

8. Для заголовка h1 создать класс и присвоить стили:
    d. font-size: 64px;
    e. color: #1F3F68.
9. Для заголовка h2 (на странице регистрация подзаголовок «Регистрация») создать класс и присвоить стили:
    a. насыщенность шрифта (500);
    b. Размер шрифта: 44px;
    c. цвет: #1F3F68;
10. Для формы добавить отступ с левой стороны (50px), отодвинув от границы страницы.
11. Для всех полей ввода (input) задать общий класс и к нему стили:
    a. border: 1px solid #356EAD;
    b. box-sizing: border-box;
    c. border-radius: 10px;
    d. прозрачность текста в полях: 0.4;
    e. ширина поля: 250px;
    f. высота поля: 30px;
    g. внешний отступ сверху и слева (10px), а справа и снизу (15px).
12. Кнопкам формы присвоить класс и к нему стиль:
    a. background: cornflowerblue;
    b. box-shadow: 5px 20px 50px rgba(16, 112, 177, 0.2);
    c. border-radius: 10px;
    d. font-weight: 500.
    e. font-size: 16px;
    f. line-height: 26px;
    g. text-align: center;
    h. color: #FFFFFF;
    i. text-transform: uppercase.
13. Пройти все уровни в игре Flexbox Froggy: https://flexboxfroggy.com/#ru