**Конспект: основы CSS**

**CSS-правила**

CSS — это язык для оформления структурированных документов, например, HTML- документов. Синтаксис — это плоский список CSS-правил. CSS-правило состоит из селектора и перечня свойств и их значений:

селектор {

свойство: значение;

свойство: значение;

}

Для комментариев в CSS используются символы /\* и \*/.

**Селекторы**

Селектор находится в начале CSS-правила, до фигурных скобок, и определяет, к каким HTML-элементам применятся свойства и значения из правила.

.feature-kitten {

padding-top: 60px;

}

Простейшие (и самые популярные) селекторы — это селекторы по тегам и по классам. Селекторы по тегам содержат имя тега без символов < и > и применяются ко всем подходящим тегам. Селекторы по классам начинаются с точки, за которой идёт имя класса, и применяются ко всем тегам с подходящим атрибутом class.

h1 { color: red; }

.info { color: blue; }

На странице может быть несколько списков, и стили применятся ко всем спискам, даже к тем, которые вы менять не хотели. Чтобы избежать таких ситуаций, лучше не использовать селекторы по тегам или использовать их как можно реже.

Если у CSS-правил отличаются только селекторы, а свойства и значения одинаковые, то их можно сгруппировать через запятую.

Также можно комбинировать любые типы селекторов через пробел. Такие селекторы называются вложенными или контекстными и читаются справа налево. Например:

nav a {…}

.menu ul {…}

.post .title {…}

**Свойства и значения**

Список свойств и значений находится внутри фигурных скобок CSS-правила. Свойство определяет, какую характеристику внешнего вида мы хотим изменить, а значение — как именно.

.feature-kitten {

padding-top: 60px;

}

Каждый раз, когда мы добавляем новое свойство или изменяем его значение, мы меняем что-то на странице.

**Наследование**

Наследование в CSS — это механизм, с помощью которого значения свойств элемента-родителя передаются его элементам-потомкам. Стили, присвоенные одному элементу, наследуются всеми потомками (вложенными элементами), но только в том случае, если они где-то явно не переопределены.

**Составные свойства**

В CSS есть обычные свойства, управляющие одним параметром отображения, и есть составные свойства, управляющие одновременно несколькими параметрами. Например, свойство font. Оно задаёт сразу шесть параметров: размер и название шрифта, высоту строки и некоторые другие.

font: 16px/26px "Arial", sans-serif;

Если значение обычного свойства не было задано в составном, то браузер при «расшифровке» использует исходное значение этого свойства.

**Типы значений: абсолютные и относительные**

Абсолютные единицы измерения привязаны к настоящим физическим размерам и связаны между собой жёсткими пропорциями. Пиксели, px, используют чаще всего, остальные абсолютные единицы почти не применяют. Примеры абсолютных единиц измерения:

font-size: 1cm;

font-size: 10mm;

font-size: 38px;

Относительные единицы измерения описывают значения, которые зависят от других значений. Например, ширина элемента в процентах зависит от ширины родительского элемента, а ширина элемента в em зависит от размера шрифта самого элемента. К относительным единицам относятся em, rem, vh, vw и некоторые другие, ну и, конечно же, проценты.

**Стили по умолчанию**

Некоторым элементам можно не задавать никаких стилей, но у них всё равно будет какое-то оформление. Например, у списка <ul> есть отступы и маркеры. Такие стили называются стилями по умолчанию и задаются внутри браузерных стилей изначально. Их можно переопределить или сбросить, задав другие значения свойств элементу.

**Каскадирование**

Когда браузер отрисовывает страницу, он должен определить итоговый вид каждого HTML-элемента. Для этого он собирает все CSS-правила, которые относятся к каждому элементу, ведь на элемент могут влиять сразу несколько CSS-правил. Механизм комбинирования стилей из разных источников в итоговый набор свойств и значений для каждого тега называется каскадностью. Например, есть такой элемент в разметке:

<p class="beloved-color">Зелёный - мой любимый цвет</p>

Заданные стили:

.beloved-color { color: green; }

Браузерные стили:

margin: 1em 0;

Итоговые стили:

color: green;

margin: 1em 0;

**Конфликт свойств**

На один элемент могут действовать несколько CSS-правил. Если в этих правилах есть одинаковые свойства с разными значениями, то возникает конфликт. Например:

ul { list-style: disc; }

.blog-navigation ul { list-style: none; }

Браузеру нужно как-то решать, какими будут итоговые значения конфликтующих свойств. Конфликт разрешается максимум за три шага. Если на текущем шаге определиться не удалось, то выполняется следующий шаг. Вот эти шаги:

1. Сравниваются приоритеты стилевых файлов, в которых находятся конфликтующие свойства. Например, авторские (то есть наши) стили приоритетнее браузерных.
2. Сравнивается специфичность селекторов у CSS-правил с конфликтующими свойствами. Например, селектор по классу более специфичен, чем селектор по тегу.
3. Побеждает то свойство, которое находится ниже в коде.

Каскад работает и внутри CSS-правил.

**Встраивание и подключение внешних стилей**

Внешние стили подключаются через тег <link>

<link rel="stylesheet" href="style.css">

Встраивание стилей в тег <style>. Его обычно размещают внутри <head>:

<head>

<style>

CSS-код

</style>

</head>

Такой способ используется для оптимизации загрузки страницы, ведь в таком случае браузер не будет отправлять дополнительных запросов на сервер.

Встраивание в атрибут style:

<div style="width: 50%;"></div>

Свойства и значения, прописанные таким образом, применятся точечно к одному элементу.

Обычно использование этого способа считается плохой практикой. Но иногда в виде исключения бывает удобнее воспользоваться встраиванием стилей в атрибут style, чем писать отдельные CSS-правила. Например, когда нужно управлять стилями именно из разметки, и создавать отдельные классы при этом будет излишне. Так бывает, когда какие-то стилевые параметры устанавливаются с помощью сторонних программ или другими людьми, например, через CMS.