**Конспект «Наследование и каскадирование»**

**Иерархическое дерево**

HTML-документ представляет собой иерархическое дерево. У каждого элемента (кроме корневого) есть только один родитель, то есть элемент, внутри которого он располагается. У корневого раздела родитель отсутствует. Рассмотрим простейшую страницу:

<html>

<head></head>

<body>

<p>Текст документа</p>

<p class="text">Выделенная <span>строка</span></p>

</body>

</html>

Для этой страницы можно нарисовать такое иерархическое дерево:

Оно схематически отображает структуру вложенности элементов.

Иерархическая структура документа определяет основы концепции наследования.

**Наследование**

Наследование в CSS — механизм, с помощью которого значения свойств элемента-родителя передаются его элементам-потомкам. Стили, присвоенные некоторому элементу, наследуются всеми потомками (вложенными элементами), если они не переопределены явно.

Рассмотрим пример:

<p class="text">Строка c выделенным <span>словом</span></p>

Представим, что нам нужно установить красный цвет текста для всего текста. Зададим CSS-свойства следующим образом:

.text {

color: red;

}

Благодаря наследованию цвет текста в теге span автоматически станет красным:

Строка c выделенным словом

А так бы выглядел результат, если бы наследование не работало:

Cтрока c выделенным словом

**Наследуемые и ненаследуемые свойства**

Не все свойства наследуются тегами-потомками от их родителей.

К наследуемым свойствам относятся в первую очередь свойства, определяющие параметры отображения текста:

font-size, font-family, font-style, font-weight, color, text-align, text-transform, text-indent, line-height, letter-spacing, word-spacing, white-space, direction и т. д.

Также к наследуемым свойствам относятся list-style, cursor, visibility, border-collapse и некоторые другие.

Весь список наследуемых свойств смотрите в [стандарте CSS](https://www.w3.org/TR/CSS21/propidx.html). Значение yes в колонке Inherited?.

Наследуемые свойства можно и нужно задавать через предков, следуя семантике документа.

Все остальные свойства относятся к ненаследуемым. Это параметры позиционирования, размеров, отступов, фона, рамок и т. д. А именно: background, border, padding, margin, width, height, position и др.

**Принудительное наследование**

Для каждого свойства может быть задано значение inherit. Оно означает, что данное свойство принимает такое же значение, как и у родительского элемента.

Запись выглядит следующим образом:

p {

background: inherit;

}

**Каскадирование**

CSS расшифровывается как «*Cascading* Style Sheets» или «*каскадные* таблицы стилей».

Каскадность обозначает, что к одному и тому же элементу может применяться несколько CSS-правил (наборов CSS-свойств). Среди этих свойств могут быть и конфликтующие, поэтому существуют инструкции, которые определяют, каким будет финальный набор свойств элемента.

Каскадирование определяет, какие именно свойства из всех возможных источников будут применены к элементу.

Имеется три основные концепции, управляющие порядком, в котором применяются CSS-свойства:

1. важность;
2. специфичность;
3. порядок исходного кода.

**Специфичность**

В случае, если элемент обладает несколькими классами и селекторы по этим классам задают одно и то же свойство с разными значениями, более высоким приоритетом обладает то правило, которое расположено в CSS-коде *ниже*.

Разберём пример. Вот HTML-код, в котором есть абзац с двумя классами:

<p class="red blue">Синий или красный?</p>

А вот CSS-код c двумя правилами для этих классов:

.blue {

color: blue;

}

.red {

color: red;

}

Абзац будет красного цвета, так как второе правило расположено ниже и является более приоритетным.

Простое объяснение специфичности звучит так:

Чем меньшее количество элементов потенциально может выбрать селектор, тем он специфичнее.

Селектор .red выберет *все теги* с нужным классом, а селектор p.red выберет *только абзацы* с нужным классом. Поэтому селектор p.red является более специфичным, чем селектор .red.

Селектор по id может выбрать только один элемент. И поэтому он на порядок специфичнее селекторов по тегам, классам, а также комбинаций этих селекторов.

CSS-правила, которые прописаны в style обладают наивысшим приоритетом. Такой способ задания стилей не приветствуется в профессиональной вёрстке сайтов и годится только для создания быстрых прототипов.

Существует способ переопределить из подключаемых CSS-файлов даже стили, заданные в атрибуте style. Для этого нужно использовать ключевое слово !important. Оно задаёт CSS-свойству усиленный приоритет. Вот пример:

HTML:

<p style="color: red;" class="blue">Синий или красный?</p>

CSS:

.blue {

color: blue !important;

}

Цвет текста в этом примере будет синим.

При вёрстке не рекомендуется часто использовать !important.

**Расчёт значения специфичности**

Специфичность селектора разбивается на 4 группы — a, b, c, d:

* если стиль встроенный, то есть определён как style="...", то а=1, иначе a=0;
* значение b равно количеству идентификаторов (тех, которые начинаются с #) в селекторе;
* значение c равно количеству классов, псевдоклассов и селекторов атрибутов;
* значение d равно количеству селекторов по тегу и псевдо-элементов.

После этого полученное значение приводится к числу (обычно в десятичной системе счисления). Селектор, обладающий большим значением специфичности, обладает и большим приоритетом.

Посчитаем специфичность в нашем примере:

|  |  |  |
| --- | --- | --- |
| Селектор | a, b, c, d | Число |
| span | 0, 0, 0, 1 | 1 |
| div.cat-in-box | 0, 0, 1, 1 | 11 |
| **#floor.cat-in-box** | **0, 1, 1, 0** | **110** |
| div span | 0, 0, 0, 2 | 2 |
| .cat-in-box | 0, 0, 1, 0 | 10 |
| #floor span | 0, 1, 0, 1 | 101 |

Отсюда сразу видно, что в нашем примере самым приоритетным является селектор #floor.cat-in-box.

**Перекрёстное наследование**

При создании стилей для сходных по внешнему виду или функциональности элементов, которые могут использоваться на странице неоднократно, очень удобно пользоваться перекрёстным наследованием.

Приём этот заключается в следующем:

1. создаётся базовый стиль для таких элементов;
2. определяются вспомогательные стили, которые применяются к элементам по мере надобности;
3. элемент наследует базовый стиль и один или несколько вспомогательных.