# Конспект

# «Наследование и каскадирование»

**Иерархическое дерево**

HTML-документ представляет собой иерархическое дерево. Это означает, что 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>

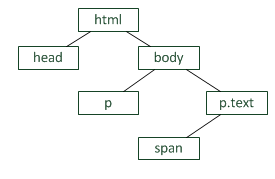
<p>Текст документа</p>

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

</body>

</html>

Для этой страницы можно нарисовать такое иерархическое дерево:



Оно схематически отображает структуру вложенности элементов. В данном примере видно, что у элемента span родителем является p.text, а у p.text родитель — body.

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. Например, размер шрифта и его цвет достаточно применить к body, чтобы все элементы внутри имели те же свойства.

Наследование позволяет сократить размер таблицы стилей, но если стилей много, то отследить какой родительский элемент установил некоторое свойство, становится сложнее.

**Наследование «на пальцах»**

Давайте на простых примерах подробнее разберёмся, в чём же преимущество наследования.

Рассмотрим пример:

<p class="text">Cтрока 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:

Cтрока c выделенным словом

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

Cтрока c выделенным словом

Нам пришлось бы отдельно прописывать цвет текста для тега span. И тогда установка таких простых свойств как стиль шрифта стала бы большой проблемой: нужно было бы задавать свойства для всех возможных вложенных тегов.

**Ещё немного про наследование**

Наверняка вы обращали внимание, что не все свойства наследуются тегами-потомками от их родителей.

Действительно, было бы странно, если бы свойство border автоматически устанавливалось для всех вложенных элементов.

Например, для этого куска кода:

<p class="bordered">Cтрока c выделенным <span>словом</span></p>

Установим CSS-свойство:

.bordered {

border: 1px solid green;

}

Если бы наследовались все свойства, то результат бы выглядел так:

Cтрока c выделенным словом

На самом деле граница будет нарисована только у тега p.

**Наследуемые свойства**

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. Но они используются значительно реже.

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

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

Например, параметры текста зачастую не меняются в пределах отдельных блоков страницы: меню, основного содержания, информационных панелей. Поэтому общие параметры текста (цвет, размер, гарнитура) обычно указывают в стилях самих блоков.

**Ненаследуемые свойства**

В предыдущем задании мы перечислили основные наследуемые свойства. Все остальные относятся к ненаследуемым. Это параметры позиционирования, размеров, отступов, фона, рамок и т. д.

А именно: background, border, padding, margin, width, height, position и др.

Весь список ненаследуемых свойств смотрите в [стандарте CSS](https://www.w3.org/TR/CSS21/propidx.html). Значение no в колонке Inherited?.

Не наследуются они из соображений здравого смысла. Например: если для какого-либо блока установлен внутренний отступ, автоматически выставлять такой же отступ каждому вложенному элементу нет никакой надобности. Эти параметры чаще всего уникальны для каждого отдельного блока.

**Принудительное наследование**

Для каждого свойства может быть задано значение inherit.

Оно означает, что данное свойство принимает такое же значение, как и у родительского элемента. Значение inherit может быть использовано для усиления наследуемых значений, а также в свойствах, которые обычно не наследуются.

Запись выглядит следующим образом:

p {

background: inherit;

}

В данном случае у тегов p свойство background будет таким же, как и у их родительских тегов.

**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 между собой. Поэтому существуют инструкции, которые определяют, каким будет финальный набор свойств элемента.

Например, для элемента:

<p class="text" style="color: red;"></p>

CSS-правила существуют как минимум в трёх разных местах:

1. в подключаемом файле style.css для селекторов p или .text;
2. в атрибуте style;
3. в стандартных стилях отображения, встроенных в браузер.

Каскадирование как раз и определяет, какие именно свойства из этих источников применятся к данному абзацу.

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

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

Самыми важными для нас являются последние две и подробнее о них мы поговорим отдельно.

Как вы уже знаете, существуют селекторы не только по классам, но и по id. Они начинаются с решётки #. HTML:

<div id="experiment-1" class="experiment">

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

</div>

CSS:

#experiment-1 .blue {

color: blue;

}

.experiment .red {

color: red;

}

Особенность атрибута id заключается в том, что его значение должно быть уникальным в пределах страницы. То есть, может существовать только один тег с определённым значением id.

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

**Запрещённый приём**

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. Поэтому мы и назвали этот приём запрещённым.

Однако 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. По возможности старайтесь обходиться без него.

**Расчёт значения специфичности**

Вы вдоволь наигрались со специфичностью, а теперь пришло время изучить полные правила её вычисления.

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 типов элементов 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.

**Пример:**

На странице используются кнопки разного назначения: для отправки форм, для сброса информации в полях формы, как элементы навигации и т. д.

Можно вынести общее оформление (размеры, отступы и так далее) для всех кнопок в отдельное CSS-правило для класса, например, .button.

А затем создать дополнительные CSS-правила, в которых будут определены только различающиеся свойства этих кнопок, например, цвет фона. Для этих правил можно использовать такие названия классов: .button-send, .button-clear, .button-navigation.

Каждая кнопка в HTML-коде будет иметь два класса: общий и дополнительный.

<a class="button button-send">Отправить</a>