**Что такое :nth-child?**

:nth-child — это псевдокласс, который позволяет выбирать элементы по их порядковому номеру. Например, можно выбрать пятый абзац и применить к нему нужные стили, или выбрать все четные строки в таблице и сделать её полосатой.

В этой демонстрации мы детально разберем механизм работы этого селектора. Начнем с исходной html-разметки. Работать будем с обычным списком.

**: nth-child (4)**

Самый простой вариант использования данного селектора: задать конкретное число. Наш селектор:

li:nth-child(4)

Он обозначает: «Выбрать четвёртый по счету элемент списка». Как видите, подсветился нужный элемент.

### Несколько :nth-child(номер)

Чтобы выбрать небольшое количество элементов, можно несколько раз использовать :nth-child с нужными номерами.

Однако, если количество таких элементов велико, то такой подход не сработает. Представьте, сколько CSS-кода нужно, чтобы выделить все чётные элементы в примере.

Не забывайте экспериментировать. Поменяйте номера в CSS-коде, чтобы подсветить другие элементы.

### :nth-child(2n)

В общем виде значение этого псевдокласса задаётся с помощью выражения: an+b, где a и b — целые числа, а n — счетчик, принимающий целые значения от 0 и больше: 0,1,2,3…

Если после вычисления выражения браузер находит элемент с полученным номером, то он применяет к нему стили. Рассмотрим пример вычисления номеров для :nth-child(2n).

1. n=0; 2\*0 = 0; нет элементов

2. n=1; 2\*1 = 2; 2-й элемент

3. n=2; 2\*2 = 4; 4-й элемент

### :nth-child(even)

Выражение :nth-child(2n) соответствует всем чётным элементам. Для чётных элементов существует также специальное ключевое слово even.

### :nth-child(2n+1)

Посчитаем номера для :nth-child(2n+1).

1. n=0; 2\*0 + 1 = 1; 1-й элемент

2. n=1; 2\*1 + 1 = 3; 3-й элемент

3. n=2; 2\*2 + 1 = 5; 5-й элемент

...

Т.е. все нечётные элементы.

### :nth-child(odd)

Для нечётных элементов тоже существует ключевое слово: odd.

### :nth-child(odd) вместе с :nth-child(even)

Используя разные стили для чётных и нечётных элементов, можно создавать интересные визуальные эффекты. Именно таким образом, с помощью чистого CSS, делают полосатые таблицы.

### :nth-child(3n-1)

Рассчитаем выражение посложнее :nth-child(3n-1).

1. n=0; 3\*0 - 1 = -1; нет элементов

2. n=1; 3\*1 - 1 = 2; 2-й элемент

3. n=2; 3\*2 - 1 = 5; 5-й элемент

...

Поэкспериментируйте с выражением в стилях.

### :nth-child(n+8)

Если использовать большое положительное число b в формуле an+b, то можно выделять все элементы, за исключением начальных. И чем больше b, тем больше начальных пропускается. Расчет для :nth-child(n+8):

1. n=0; 0+8 = 8; 8-й элемент

2. n=1; 1+8 = 9; 9-й элемент

3. n=2; 2+8 = 10; 10-й элемент

...

### :nth-child(-n+14)

Можно использовать отрицательный n. Расчёт для :nth-child(-n+14):

1. n=0; 0+14 = 14; 14-й элемент

2. n=1; -1+14 = 13; 13-й элемент

...

15. n=14; -14+14 = 0; нет совпадений

Т.е. :nth-child(n+8) означает не выделять 7 элементов вначале, а все остальное выделить. :nth-child(-n+14) обозначает выделить 14 элементов в начале, а все остальное не выделять.

### Комбинирование выражений

На предыдущих шагах мы с помощью разных выражений выделяли разные множества элементов. Существует возможность задавать множество с помощью комбинирования выражений.

Например: :nth-child(n+8):nth-child(-n+14). Итоговое множество получается как пересечение двух исходных:

:nth-child(n+8) выделит: 8-21

:nth-child(-n+14) выделит: 1-14

на пересечении 1-14 и 8-21: 8-14

### :nth-child(n+4):nth-child(-n+18)

Комбинируя выражения, мы можем задавать произвольные диапазоны элементов. Можно немного расширить выделение из предыдущего шага.

Сдвинем левую границу: :nth-child(n+8) → :nth-child(n+4)

Сдвинем правую границу: :nth-child(-n+14) → :nth-child(-n+18)

Попробуйте поизменять размер выделения

### :nth-child(n+4):nth-child(-n+18):nth-child(odd)

Можно комбинировать более двух выражений. Выражение в заголовке обозначает: «взять элементы с 4 по 18 включительно и среди них выделить только нечётные».

### Два сложных множества

Мы создали множество «элементы с 4 по 18 включительно, нечётные».

Создадим второе множество «элементы с 4 по 18 включительно, чётные».

Вот такой интересный псевдокласс :nth-child, с помощью которого можно выделять практически любые множества элементов, используя их порядковый номер.

### Псевдокласс :nth-of-type

:nth-of-type работает почти так же, как и :nth-child. Разница заключается в том, что он учитывает тип элемента.

Для того, чтобы поэкспериментировать с этим псевдоклассом, изменим код примера. Вначале в HTML коде создадим список из двух элементов: span и strong. И обернём этот список в контейнер.

### :nth-of-type(3)

Попробуем простейшие выражения с числом.

Выделить третий span:

span:nth-of-type(3)

Выделить пятый strong:

strong:nth-of-type(5)

### :nth-of-type(odd)

Выражения с чётными/нечётными элементами.

Выделить нечётные span:

span:nth-of-type(odd)

Выделить чётные strong:

strong:nth-of-type(even)

### Комбинированные :nth-of-type

Выделить нечётные span из первых четырёх:

span:nth-of-type(odd):nth-of-type(-n+4)

Выделить чётные strong из идущих после четвёртого:

strong:nth-of-type(even):nth-of-type(n+5)

### Отличие :nth-of-type от :nth-child

Значения для двух псевдоклассов задаются абсолютно одинаково. Отличие заключается в том, что нумерация элементов при использовании :nth-of-type идёт только между потомками одного родителя заданного типа. Нумерация элементов при использовании :nth-child идёт между всеми потомками одного родителя.

Продемонстрируем это отличие еще раз. span:nth-of-type(odd) подсвечивает только нечётные лежачие прямоугольники.

### Отличие :nth-of-type от :nth-child

Если для этого же кода использовать span:nth-child(odd), то подсвечиваются все лежачие прямоугольники.

Это связано с тем, что после каждого элемента span, расположен strong, который влияет на счетчик псевдокласса :nth-child.

### Практический пример использования :nth-child

Иногда дизайнеры рисуют каталог товаров, в котором товары расположены в несколько колонок, причем края колонок выровнены строго по краям родительского блока.

Сверстать такой блок не проблема. Можно задать разные классы элементам левой и правой колонок и пусть программист мучается при генерации правильного html кода страницы.

Но намного лучше сверстать такой список товаров, не используя разные классы. На этом шаге задан исходный HTML.

### Практический пример: оформляем товары

Зададим базовый CSS код нашего каталога. Пунктирные линии справа и слева — это границы блока, по которым должны выравниваться края колонок.

Нужно создать двухколончатую сетку, в которой ширина каждой колонки составляет 45% от ширины родительского блока. Правая колонка прижата к правому краю, а левая к левому. Приступим.

### Практический пример: сетка

Чтобы получить нужную сетку, достаточно задать следующие проценты:

45% - ширина левой колонки

10% - отступ справа левой колонки

45% - ширина правой колонки

Итого: 45% + 10% + 45% = 100%

Мы использовали box-sizing:border-box;, чтобы на ширину колонок не влияли границы. Товары расположились в одну колонку, зато она правильной ширины.

### Практический пример: две колонки

Чтобы появилась вторая колонка, нужно избавиться от отступа справа у каждого второго элемента списка.

Мы не используем разные классы для разных элементов, поэтому на помощь приходит псевдокласс li:nth-child(even). С помощью него удаляются лишние отступы и идеальная сетка готова.

### Практический пример: три колонки

При таком подходе можно изменять количество колонок только с помощью CSS. То есть, HTML код страницы, а значит и логику генерации кода на сервере менять не надо.

Сначала изменим проценты:

30% - ширина колонок

5% - отступы справа у 1 и 2 колонки

Итого: 30% + 5% + 30% + 5% + 30% = 100%

### Практический пример: три колонки готовы

Затем нужно удалить отступ справа у каждого третьего элемента. Для этого используем li:nth-child(3n). Готово!

Отличное решение, особенно для адаптивной верстки.

ul {

  width: 400px;

  height: 400px;

  margin: 50px auto;

  padding: 0;

  list-style: none;

  border-right: 1px dashed #cccccc;

  border-left: 1px dashed #cccccc;

}

li {

  float: left;

  width: 30%;

  margin-right: 5%;

  box-sizing: border-box;

  height: 50px;

  margin-bottom: 20px;

  border-radius: 10px;

  background: #eeeeee;

  border: 1px solid #dddddd;

  text-align: center;

  line-height: 50px;

  font-size: 20px;

  font-family: "Tahoma", sans-serif;

  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

li:nth-child(3n) {

  margin-right: 0;

  background-color: #5EB95E;

  color: white;

}

  <ul>

    <li>Лук</li>

    <li>Морковь</li>

    <li>Сыр</li>

    <li>Колбаса</li>

    <li>Хлеб</li>

    <li>Картофель</li>

    <li>Огурцы</li>

  </ul>