Урок 2.4. Позиционирование в CSS. Комбинирование селекторов

Комбинирование селекторов

Вложенность

Чтобы переопределить свойства элемента, который находится только внутри определённого контейнера, существует комбинирование селекторов.

Например, чтобы переопределить свойства `.element` только внутри контейнера `.container-green` (то есть если `.element` будет находиться в других контейнерах, то к нему эти свойства не применятся), необходимо воспользоваться вложенностью через пробел:

```
/* базовые стили элемента, которые применяются всегда */
.element {
    width: 100px;
    height: 100px;
    background: red;
}

/* применится только для тех элементов, которые находятся внутри контейнера
   .green-container */
.container-green .element {
    background: green;
}
```

При этом не имеет значения уровень вложенности, где бы ни находился `.element`, если один из его родителей `.container-green`, то фон станет зелёным.

```
<div class="container">
    <!-- элементы будут иметь красный фон -->
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
</div>
```

```
<div class="container-green">
    <!-- элементы будут иметь зелёный фон -->
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="second-container">
        <!-- здесь тоже будет зелёный фон, несмотря на то, что
            .container-green – не непосредственный родитель -->
        <div class="element"></div>
    </div>
</div>
```

Непосредственная вложенность

В отличие от предыдущего способа, здесь выбираться будут только те элементы, которые непосредственно вложены в родителя:

```
/* базовые стили элемента, которые применяются всегда */
.element {
    width: 100px;
    height: 100px;
    background: red;
}

/* применится только для тех элементов, которые находятся НЕПОСРЕДСТВЕННО
   внутри контейнера .green-container */
.container-green > .element {
    background: green;
}
```

```
<div class="container">
    <!-- элементы будут иметь красный фон -->
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
</div>
<div class="container-green">
    <!-- элементы будут иметь зелёный фон -->
    <div class="element"></div>
    <div class="element"></div>
    <div class="element"></div>
    <div class="second-container">
        <!-- этот элемент будет иметь базовый красный фон,
             потому что не является непосредственным потомком
             .container-green -->
        <div class="element"></div>
```

```
    </div>
</div>
```

Селектор следующего элемента

Данный метод позволяет выбрать следующего за текущим соседа, находящегося на том же уровне.

```
/* у элемента .ellipse, который следует сразу за элементом
   .circle, будет красный фон */
.circle + .ellipse {
    background: red;
}
```

```
<div class="container">
    <div class="circle"></div>
    <!-- у данного элемента будет красный фон -->
    <div class="ellipse"></div>

    <div class="square"></div>
    <!-- у данного элемента не будет красного фона, потому что
         он не следует сразу за элементом .circle -->
    <div class="ellipse"></div>
</div>
```

Селектор любого следующего элемента

В отличие от предыдущего метода, здесь не имеет значение, является ли элемент непосредственным соседом.

```
/* у элемента .ellipse, который следует после элемента
   .circle, будет красный фон. При этом не важно, является ли .ellipse
   непосредственным соседом. Главное условие, чтобы .ellipse располагался
   после .circle на одном и том же уровне. При этом между ними могут быть
   другие элементы.
 */
.circle ~ .ellipse {
    background: red;
}
```

```
<div class="container">
    <div class="circle"></div>
    <!-- у данного элемента будет красный фон -->
    <div class="ellipse"></div>

    <div class="square"></div>
    <!-- у данного элемента тоже будет красный фон, потому что
         перед ним на том же уровне был элемент .circle (несмотря на
         наличие .square и .ellipse между ними
    -->
    <div class="ellipse"></div>
</div>
```

Стоит отметить, что и в этом методе, и в предыдущем выбирать можно только следующий элемент, который в вёрстке находится ниже на том же уровне вложенности.

Позиционирование

Базовый поток документа

- Вывод элементов на страницу осуществляется в том порядке, в котором они следуют в HTML коде
- Блочные элементы располагаются сверху вниз
- Инлайн элементы располагаются слева направо, и переносятся на следующую строку, когда места по ширине не остаётся

Position

- Свойство `position` используется, чтобы изменить базовое поведение элементов в потоке
- `position` имеет 5 значений:
  - `static` - статичное позиционирование
  - `relative` - относительное
  - `absolute` - абсолютное
  - `fixed` - фиксированное

    - `sticky` - "липкое"

`position: static`

- Значение по умолчанию
- Элемент находится в потоке
- Элемент не считается позиционированным

`position: relative`

- Элемент ведёт себя как элемент в потоке, но:
- Свойства `left`, `top`, `bottom`, `right` могут сдвигать его со своего места
- Место, занимаемое элементом в потоке - сохраняется

`position: absolute`

- Элемент не сохраняет своё место в потоке и вынимается из него
- По умолчанию занимает то место, которое занимал бы в потоке, но пересекает другие элементы в потоке
- Может быть передвинут свойствами `left`, `top`, `bottom`, `right``left`, `top`, `bottom`, `right`
- Позиционирование выполняется относительно ближайшего позиционированного предка

`position: fixed`

- Похоже на абсолютное, но в отличие от него, всегда выполняется относительно краёв браузера
- При скролле элемент не скроллится и остаётся на изначальной позиции (следствие первого пункта)

`position: sticky`

- Залипающее позиционирование
- Элемент ведёт себя как `relative`, пока скролл не дошёл до значения, указанного в свойстве `top`, потом переключается на `fixed`
- Свойство `top` - обязательное к указанию

- Также работает и со свойством `left`, однако вследствие непопулярности горизонтального скролла, используется редко

`z-index`

- Позволяет задать порядок элементов по оси Z (дальше от нас, ближе к нам)
- По умолчанию ближе находятся элементы, расположенные в вёрстке ниже
- Не работает на `position: static`
- Значения в диапазоне `0..9999`. Разрешены любые, и отрицательныее, но при отрицательных на элемент нельзя будет нажать курсором, он будет недоступен для взаимодействия
- По умолчаню при вложенности элементов ближе к нам располагаются те элементы, чей уровень вложенности больше