**Псевдоклассы** – это инструмент, который позволяет элементам реагировать на действия пользователя и давать обратную связь. Например, псевдокласс ***:hover*** выбирает элемент, на который наведен курсор мыши. Селектор псевдокласса ***.article-link:hover*** выберет на странице все элементы с классом ***.article-link*** и применит к ним стили из правила, когда на них будет наведен курсор.

*button {*

*border: 2px solid red;*

*color: white;*

*background-color: black;*

*}*

*button:hover { // перекрашивает кнопку при наведении*

*border: 2px solid black;*

*background-color: red;*

*}*

Рассмотрим пример, когда при наведении на элемент одного класса будет меняться прозрачность элементов другого класса. Этого можно достичь с помощью свойства ***hover***.

*.error {*

*font-size: 120px;*

*}*

*.message {*

*opacity: 0; // сделать прозрачным*

*}*

*.error:hover ~ .message {*

*opacity: 1; // сделать видимым*

*}*

Псевдокласс ***:link*** стилизует все ссылки, по которым еще не переходили. Его редко используют, потому что селекторы ссылки ***a*** и ***a:link*** абсолютно идентичны, и стили применяются к одним и тем же элементам.

Псевдокласс ***:visited*** стилизует все ссылки, по которым уже переходили. В целях безопасности возможности стилизации посещенных ссылок ограничены следующими свойствами: ***color***, ***background-color***, ***border-color***, ***outline-color***, ***column-rule-color***, ***fill***, ***stroke***. Также браузер будет игнорировать прозрачные (***transparent***) и полупрозрачные цвета (***rgb* с альфа-каналом**).

Псевдокласс ***:active*** стилизует элементы в тот миг, когда на них уже нажали, но еще не отпустили клавишу мыши. Это промежуточное состояние называется **активным**.

Для исключения конфликтов рекомендовано писать псевдоклассы для элемента в следующем порядке:

*link -> visited -> hover -> active*

Псевдокласс ***:focus*** стилизует элементы, на которые навелись любым способом (например, с помощью *Tab* клавиатуры или после клика по ним). Как правило он изменяет обводку (***outline***) элемента.

*button:focus {*

*outline: 15px solid lightpink;*

*}*

Псевдокласс ***:focus-visible*** стилизует элементы, на которые навелись только с клавиатуры. Потребуется сбросить стили для ***:focus***.

*button:focus {*

*outline: none;*

*}*

*button:focus-visible {*

*border-color: rebeccapurple;*

*color: palevioletred;*

*}*

Псевдокласс ***:focus-within*** стилизует элементы, которые либо сами находятся в фокусе (на которые навелись любым способом), либо имеют дочерние элементы в фокусе.

*form:focus-within { // при фокусировки любого элемента формы, форма сменит цвет*

*background-color: #c4c4c4;*

*}*

Псевдоклассы ***:nth-child()*** и ***:nth-last-child()*** стилизуют каждый ***n***-й элемент с начала и конца соответственно. Внутри скобок указывается либо формула, либо номер, либо ключевые слова (например, ***even*** для четных и ***odd*** для нечетных). Подсчет элементов ведется внутри одного блока родителя. Визуализировать формулы можно [тут](https://css-tricks.com/examples/nth-child-tester/).

*li:nth-child(3n) { // перекрасим каждый 3й элемент списка*

*color: #fd24af;*

*}*

*li:nth-child(5) { // подчеркнем 5й элемент списка*

*text-decoration: underline;*

*}*

*.card:nth-child(even) { //* заменить правый отступ на левый у каждого четного элемента

*margin-right: 0;*

*margin-left: 80px;*

*}*

Псевдоклассы ***:first-child*** и ***:last-child*** являются производными ***:nth-child()*** и стилизуют первый и последний элементы соответственно внутри одного блока родителя.

Псевдокласс ***:nth-of-type()*** и его производные работают аналогично ***:nth-child()***, но считают только однотипные элементы.

*button:first-of-type { // перекрасить цвет первой кнопки внутри родителя*

*color: #fd24af;*

*}*

Псевдокласс ***:not()*** используется для исключения селектора из выборки. Внутри скобок могут быть и другие псевдоклассы, перечисленные через запятую.

*.text:not(h2) { // изменить цвет всех элементов, кроме заголовков h2*

*color: #fd24af;*

*}*

*p:not(:last-of-type) { // задать отступы всем абзацам, кроме последнего*

*margin-bottom: 25px;*

*}*

Псевдокласс ***:has()*** позволяет изменить родительский элемент указанного селектора. Сработает для любого уровня вложенности.

*.card:has(.label) { // если у card есть пометка label, то card меняет цвет рамки*

*border-color: #5f0dee;*

*}*

Пример выпадающего списка через псевдоклассы <https://codepen.io/spacehaz/pen/poYdWXz>

**Псевдоэлементы** – это инструмент, который позволяет выбрать только часть какого-то элемента. Например, псевдоэлемент ***::first-line*** выберет только первую строку текста, какой бы длины она ни была и как бы ни менялась. Селектор псевдоэлемента ***.article-text::first-line*** выберет на странице все элементы с классом ***.article-text*** и применит стили из правила только к их первой строке текста.

Псевдоэлементы ***::before*** и ***::after*** позволяют встраивать блоки перед и после элемента соответственно с помощью свойства ***content***. Не забываем сбрасывать ***box-sizing*** для них через общий селектор ***\****. Скринридеры не увидят эти изменения. Основная задача этих псевдоэлементов – избавиться от пустых или бессмысленных блоков. Не работает с тегами, вместо которых что-то рисуется.

*p::before { // добавим текст перед абзацем*

*content: "Заметка! ";*

*color: #fd24af;*

*}*

*p::after { // добавим текст после абзаца*

*content: "!";*

*color: #fd24af;*

*}*

*.header::before { // добавим затемняющий слой поверх картинки и перед заголовком*

*content: ‘ ’;*

*position: absolute;*

*z-index: -1;*

*inset: 0;*

*background-color: rgb(0 0 0 / 0.5);*

*}*

Свойство ***white-space*** позволяет редактировать текст внутри свойства ***content***. Например, переносить на новую строку.

*.page\_\_title:hover::before { // всплывающая подсказка при наведении на текст*

*content: 'На самом деле нет.\AВажнее &mdash; научиться ошибаться.';*

*white-space: pre;*

*}*

Псевдоэлемент ***::marker*** используется для создания самодельного маркера в списках. На текущий момент поддерживает свойства ***content***, ***white-space***, ***text-transform***, ***letter-spacing***, ***color*** и все свойства семейства ***font***. Раньше эту задачу решали при помощи ***::before***.

*li {*

*display: flex;*

*align-items: center;*

*}*

*li::before { // создать кастомный маркер и поместить перед элементом списка*

*content: “”;*

*display: inline-block;*

*width: 15px;*

*height: 15px;*

*margin-inline-end: 10px;*

*border-radius: 50%;*

*background-color: red;*

*}*

*li:nth-child(2)::before { // сменить цвет маркера*

*background-color: orange;*

*}*

*li::marker { // аналогично через marker*

*color: #23054a;*

*font-weight: bold;*

*font-size: 2rem;*

*}*

Псевдоэлемент ***::selection*** позволяет менять стили текста при выделении. На текущий момент поддерживает свойства ***color***, ***background-color***, ***text-decoration*** и ***text-shadow***.

*p::selection { // покрасить цвет и фон при выделении текста*

*color: #6fc1fd;*

*background-color: black;*

*}*

Псевдоэлемент ***::placeholder*** позволяет менять стили текста подсказок в полях ввода.

*input::placeholder {*

*color: rgb(0 0 0 / 60%);*

*}*

Шорткат ***transition*** позволяет управлять плавностью изменения состояния и объединяет несколько свойств в следующем порядке:

1. ***transition-property*** определяет свойство, к которому применится переход.
2. ***transition-duration*** задает длительность перехода.
3. ***transition-delay*** задает задержку перехода. Может принимать отрицательные значения.
4. ***transition-timing-function*** определяет, как меняется стиль элемента с течением времени. Может принимать следующие значения:

* ***ease*** – значение по умолчанию, ускоряется к середине анимации, замедляется в конце.
* ***ease-in*** – начинается медленно, заканчивается быстро.
* ***ease-out*** – начинается быстро, заканчивается медленно.
* ***ease-in-out*** – медленно начинается и заканчивается, а в середине ускоряется.
* ***linear*** – меняется с одинаковой скоростью.

Шорткат можно использовать для большинства свойств (*margin*, *height*, *font-size* и т.д.), кроме *display*, *visibility* и *z-index*. Время указывается в секундах или милисекундах.

*.link-nice::after {*

*content: '';*

*display: block;*

*width: 0%;*

*height: 3px;*

*background-color: black;*

*// изменение ширины подчеркивания с 100% до 0 без задержки*

*// изменение цвета текста на стандартное значение без задержки*

*transition: none;*

*}*

*.link-nice:hover::after {*

*width: 100%;*

*background-color: #fd24af;*

*// изменение ширины подчеркивания с 0 до 100% при наведении за 300мс*

*// изменение цвета текста при наведении за 0,5с*

*transition: width 300ms, background-color 0.5s;*

*}*

Шорткат ***transform*** позволяет управлять изменением состоянием объекта и объединяет несколько функций:

* ***scale()*** – отвечает за изменение масштаба элемента без влияния на соседей. Она объединяет функции *scaleX()* и *scaleY()*. В качестве аргумента может принимать одно (сразу для строчной оси X и блочной оси Y), два (первое – для оси X, второе – для оси Y) или три значения (первое – для оси X, второе – для оси Y, третье – для оси Z).

*.card:hover {*

*transform: scale(1.1); // задаем увеличение карточки на 10% при наведении*

*}*

* ***rotate()*** – отвечает за поворот элемента. В качестве аргумента может принимать градусы *deg*, повороты *turn*, радианы *rad* и грады *grad*. Если потребуется 3D-поворот, то можно указать, к какой конкретно оси относится заданное значение.

*.card:hover .card-image {*

*transform: rotate(5deg); // задаем поворот картинки карточки на 5 градусов при наведении*

*rotate: x 5deg; // альтернативная запись*

*}*

* ***translate()*** – отвечает за смещение элемента. В качестве аргумента может принимать одно (только для горизонтальной оси X), два (первое – для оси X, второе – для вертикальной оси Y) или три значения (первое – для оси X, второе – для оси Y, третье – для оси Z).

*.card-content {*

*transform: translate(0, 100%); // сдвигаем текст карточки вниз за экран*

*}*

*.card:hover .card-content {*

*transform: translate(0, 0); // убираем сдвиг текста карточки при наведении*

*transition: transform 0.5s 0.2s // добавляем плавности переходу*

*}*

* ***skew()*** – отвечает за наклон элемента. Она объединяет функции *skewX()* и *skewY()*. В качестве аргумента может принимать одно (смещает верхнюю часть относительно нижней) или два значения (первое смещает верхнюю часть относительно нижней, второе – смещает левую сторону относительно правой).
* ***matrix()*** – позволяет записать любую 2D-трансформацию в виде матрицы. Для удобства рекомендуется использовать [генератор матриц](https://angrytools.com/css-generator/transform/).

Функции трансформации можно указывать последовательно в одном шорткате, но результат может отличаться в зависимости от порядка их следования. Чтобы избежать сложения функций в шорткате и их влияния друг на друга, рекомендовано записывать их как отдельные свойства.

*transform: rotate(15deg) scale(1.1) translateY(20%);*

*// альтернативная запись*

*rotate: 15deg;*

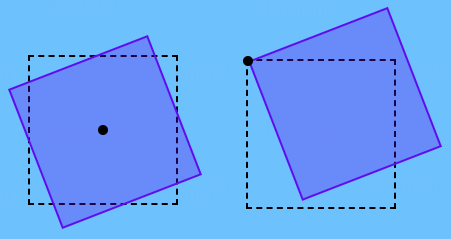
*scale: 1.1;*

*translateY: 20%;*

**Точка трансформации** – это координаты, вокруг которых происходит трансформация. По умолчанию, точка трансформации находится в центре элемента, а все координаты в вебе считаются от верхнего левого края.

Свойство ***transform-origin*** позволяет перемещать точку трансформации. В качестве аргумента может принимать до трех числовых значений или ключевых слов (*center*, *top*, *left*, *right*, *bottom*). Для оси Z нет ключевых слов.

Для примера, у правого куба задано свойство *transform-origin: 0 0;*



Свойства *opacity*, *transform* и отдельные функции трансформации, которые не меняют геометрию элемента и на находящихся рядом соседей и родителей, являются дешевыми, быстрыми в обработке и практически не нагружают браузер. В процессе анимации эти свойства затрагивают только слой с текущим элементом и не заставляют браузер перерисовывать всю страницу.

Если же этих свойств недостаточно, то для выделения элемента в отдельный слой используется свойство ***will-change***. В качестве значения оно принимает свойство, которое планируется изменять. По умолчанию используется значение *auto* и браузер сам пытается догадаться, нужно ли выносить элемент на отдельный слой.

*.link {*

*will-change: transform;*

*}*

*.link:hover {*

*transform: translateX(10px);*

*}*