**Простая анимация на чистом CSS**

**Цель практического задания** – познакомить с возможностями CSS.

**Введение**

Мы привыкли использовать CSS по его прямому назначению — для построения сеток и стилизации интерфейсов. И это, бесспорно, основная задача. Но в этом задании мы рассмотрим, как ещё можно использовать CSS — для создания и анимирования элементов. Чтобы понимать происходящее, достаточно владеть HTML и CSS на базовом уровне.

Будем делать такое дерево:

## Создание дерева

### 1. Разметка

Начнём всё же с HTML, без него никак. Нам понадобится создать простую разметку для описания будущего дерева.

Рисуют обычно на холсте, и нам он тоже понадобится, поэтому для начала создадим блок .canvas. Внутри этого блока будет находиться непосредственно дерево .tree. У дерева будет ствол .trunk и несколько веток — элементы с классом .branch. По сути ветви — это части ствола, и мы отразим это в разметке, сделав их дочерними элементами блока .trunk. То же самое с листьями — элементы .leaf будут вложены в родительский элемент своей ветки.

Вот так выглядит фрагмент разметки дерева с одной веткой, остальные — по аналогии:

<div class="canvas">

<div class="tree">

<div class="trunk">

<div class="branch">

<div class="leaf"></div>

<div class="leaf"></div>

<div class="leaf"></div>

</div>

</div>

</div>

</div>

Конечно, пока это просто несколько пустых блоков, но каркас мы уже сформировали. Двигаемся дальше.

### 2. Подготовка

Переходим к CSS. Для начала нужно спозиционировать наше будущее дерево на холсте. Сделаем .canvas флекс-контейнером и выровняем вложенный элемент .tree по центру. Также определим размеры холста и зададим ему фоновый цвет.

.canvas {

display: flex;

justify-content: center;

align-items: center;

width: 100%;

min-height: 600px;

background-color: #d1cee0;

}

Возможно, вы пока не знакомы с [флексбоксами](https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox" \t "_blank), но это не страшно. Сейчас мы используем их только чтобы отцентровать элемент. Если будет интересно, [интерактивный курс про флексбоксы](https://htmlacademy.ru/courses/96) поможет разобраться в теме.

### 3. Рисуем ствол и ветки

Всё, теперь мы точно добрались до рисования.

И ствол, и ветки будут одинакового цвета и формы, поэтому сразу объединим все CSS-свойства, с помощью которых добьёмся нужного эффекта. Цвет зададим с помощью [градиента](https://developer.mozilla.org/en-US/docs/Web/CSS/linear-gradient), чтобы элементы казались менее плоскими, а ещё добавим небольшое скругление на концах веток и ствола. Так будет выглядеть код:

.trunk,

.branch {

border-radius: 25px;

background: linear-gradient(to right, #7f3333, #4d2020);

}

Чтобы элементы, наконец, отобразились, нужно задать им размеры. Ствол сделаем шириной 10px, а ветки в два раза тоньше — по 5px. Плюс зададим стволу высоту и выровняем его по центру. На следующем шаге мы будем распределять ветки по своим местам на стволе дерева, а для этого нужно задать стволу относительное позиционирование, а веткам — абсолютное. Это позволит задавать положение каждой конкретной ветки относительно ствола свойствами top, right, bottom, left и имитировать рост веток.

.trunk {

position: relative;

width: 10px;

height: 500px;

margin: 0 auto;

}

.branch {

position: absolute;

width: 5px;

}

### 4. Ставим ветки на место

Пришло время нашей заготовке превратиться в то, что действительно будет похоже на дерево.

1. Для начала часть веток должна быть слева, а другая — справа. Мы будем выбирать нужные ветки с помощью селектора nth-child(even) — для чётных элементов и nth-child(odd) — для нечётный, а затем вращать их с помощью [свойства transform](https://developer.mozilla.org/ru/docs/Web/CSS/transform) и функции rotate на 60 градусов влево и вправо.
2. .branch:nth-child(even) {
3. transform: rotate(60deg);
4. }
5. .branch:nth-child(odd) {
6. transform: rotate(-60deg);

}

Здесь есть одна тонкость. По умолчанию элемент вращается относительно своего центра, а это не то поведение, которое нам нужно от веток. Они должны вращаться относительно нижней точки элемента — места прикрепления к стволу. И есть хорошая новость — мы можем переопределить поведение по умолчанию, используя свойство transform-origin с подходящим значением, тогда ветки будут вращаться относительно своей нижней точки, а не вокруг центра. Добавим элементу .branch нужное свойство в дополнение к уже существующим:

.branch {

position: absolute;

width: 5px;

transform-origin: bottom center;

}

Чтобы лучше понять, как работает свойство transform-origin, посмотрите [эту демку](https://codepen.io/shshaw/pen/pOVKjb).

1. Теперь нам нужно учесть, что ветки расположены несколькими ярусами, и их длина становится тем меньше, чем ближе к верхушке дерева они растут. Укажем для каждой ветки расстояние от верхушки и её длину. Ниже фрагмент кода для первых трёх веток, а дальше зададим значения для остальных веток по аналогии:
2. .branch:nth-child(1) {
3. top: 180px;
5. height: 180px;
6. }
7. .branch:nth-child(2) {
8. top: 160px;
9. height: 150px;
10. }
11. .branch:nth-child(3) {
12. top: 120px;
14. height: 150px;

}

### 5. Рисуем листья

Как видите, мы соблюдаем логику, заложенную природой — ветки выросли из ствола, а листья будут расти из веток. Листья — дочерние элементы, поэтому снова позиционируем их относительно родительского элемента .branch.

.leaf {

position: absolute;

width: 15px;

height: 15px;

border-radius: 75% 0 75% 0;

background: linear-gradient(to right, #77ed9e, #53ad71);

}

Ещё добавили листикам размер и цвет с помощью градиента, а также закруглили края. Осталось расположить каждый ряд на своём уровне, задав свойство top с соответствующими значениями. Вот код для первых двух рядов:

.leaf:nth-child(1) {

top: 5px;

}

.leaf:nth-child(2) {

top: 20px;

}

На этом с созданием дерева мы закончили, осталось только разместить листочки на своих местах и, наконец, добавить анимацию.

## Анимация листьев

### 1. Ключевые кадры

Давайте для начала выясним, как устроена анимация, и какие CSS-свойства нужны, чтобы её создать.

Для объявления анимации и задания ключевых кадров используется правило @keyframes, после которого указывается название анимации. С помощью ключевых кадров можно задать нужное поведение для элементов на любом этапе. Кадры можно задавать в процентах: например, 0% — это начало анимации, 100% — её конец. Это не единственный способ — можно воспользоваться ключевыми словами from и to, но проценты позволяют задать любое промежуточное состояние. Код нашей анимации:

@keyframes leaf-odd-grow {

0% {

transform: scale(0);

}

100% {

transform: scale(1);

}

}

Что здесь происходит? Свойство transform мы уже применяли ранее, но теперь используем функцию scale, которая позволяет изменять масштаб элемента. В начале анимации масштаб нулевой (параметр 0), а затем он должен увеличиться до обычного масштаба (параметр 1). И это именно то, что нужно, чтобы имитировать плавный рост наших листьев.

### 2. Анимирование элементов: теория

Хорошо, мы создали анимацию, а теперь нам нужно её применить к конкретным элементам. Для этого понадобятся несколько CSS-свойств:

* animation-name — название анимации. Мы уже задавали название при создании анимации с помощью @keyframes, именно его и нужно указать.
* animation-duration — длительность анимации. Измеряется в секундах или миллисекундах.
* animation-delay — задержка анимации. Свойство позволяет установить время между тем моментом, когда анимация была присвоена элементу, и непосредственно началом анимации.
* animation-fill-mode — состояние элемента до и после анимации. С помощью этого свойства можно контролировать, как будет себя вести элемент до начала анимации и после её завершения. У свойства есть [несколько значений](https://developer.mozilla.org/ru/docs/Web/CSS/animation-fill-mode).

Это только некоторые свойства, которые понадобятся нам сейчас, но есть и другие. Познакомиться с остальными можно, изучив [MDN](https://developer.mozilla.org/ru/docs/Web/CSS/animation) или [курс по анимации](https://htmlacademy.ru/courses/80).

### 3. Анимирование элементов: практика

Теперь мы знаем что делать — нужно задать созданную ранее анимацию leaf-odd-grow листочкам, и они начнут расти. Для этого укажем название анимации и её длительность.

.leaf:nth-child(odd) {

left: 100%;

transform-origin: 0% 100%;

animation-name: leaf-odd-grow;

animation-duration: 4s;

animation-fill-mode: both;

}

Но для чего мы задали значение both свойству animation-fill-mode? По умолчанию после окончания анимации элементы возвращаются в исходное состояние, а в данном случае нам это не нужно. Мы хотим, чтобы исходное состояние анимации было как в первом ключевом кадре (0%), а финальное состояние — как в последнем (100%).

Кроме самой анимации в этом фрагменте кода мы задаём положение листков относительно родительского элемента и точку применения трансформации с помощью уже знакомого свойства transform-origin. По умолчанию листки росли бы в центральной точке и увеличивались равномерно во все стороны. Но тогда в начале анимации они бы повисли в воздухе рядом с веткой, что не очень реалистично, поэтому мы переопределили это поведение и заставили их расти от начала ветки.

Возможно, вы обратили внимание, что мы анимировали только нечётные элементы. Для анимации остальных нам понадобится добавить поворот на 90 градусов, чтобы листки росли с обеих сторон ветки. Получается, чётные будут направлены в одну сторону, а нечётные — в другую.

@keyframes leaf-even-grow {

0% {

transform: rotate(-90deg) scale(0);

}

100% {

transform: rotate(-90deg) scale(1);

}

}

.leaf:nth-child(even) {

left: -150%;

transform-origin: 50% 100%;

animation-name: leaf-even-grow;

animation-duration: 4s;

animation-fill-mode: both;

}

Мы создали ещё одну анимацию, теперь чётные элементы будут увеличиваться в размере также, как и нечётные, но плюс к этому они с самого начала анимации будут повёрнуты под нужным углом.

Остался последних штрих — добавим задержку анимации для каждого ряда листьев, чтобы они появлялись не одновременно, а по очереди. Вот эти три листка появятся на концах веток и будут последними, так как у них самая большая задержка. Для всех остальных задержка будет уменьшаться с шагом 0.5s:

.leaf:nth-child(1) {

top: 5px;

animation-delay: 3.5s;

}

.leaf:nth-child(2) {

top: 20px;

animation-delay: 3s;

}

.leaf:nth-child(3) {

top: 50px;

animation-delay: 2.5s;

}

Ура-ура, дерево готово!