**Смотрим на примере**

Я думаю, лучше рассматривать сразу на примере. Допустим, у нас есть два одинаковых блока. В html разметке я предлагаю обозначить их так:

<div class="block block1">Блок 1</div>

<div class="block block2">Блок 2</div>

Каждый элемент получил два стилевых класса. Зачем это нужно? Первым классом мы зададим общие правила внешнего вида, блоки же одинаковые, так что их можно описать совместно. В реальном примере, скорее всего, будет по-другому. Также каждый элемент получил собственный уникальный стилевой класс. Он тоже пригодится.

**.block{**

**font-size: 50px;**

**width: 200px;**

**height: 120px;**

**background: green;**

**margin-bottom: 30px;**

**transition: 1s**

**}**

Этими правилами мы прописали обоим блокам одинаковый внешний вид: высоту, ширину, размер шрифта, отступ и зеленый фон. Последнее свойство **transition** как раз и является относительно новым и входит в версию **CSS3**. Оно обозначает задержку, после которой выполняются определенные правила. По сути, именно это дает плавное появление.

В примере я прописал 1 секунду, по желанию можно поставить больше или меньше. Когда вы увидите свойство в действии, то сами сможете отрегулировать его значение.

**Как реализовать в css3 плавное появление?**

Дальше есть два варианта действий. Допустим, нам нужно пока скрыть второй блок, а при наведении на первый плавно выводить его. Наиболее часто для скрытия элемента применяют правило **display: none**, но оно не даст именно того, чего нужно – плавного появления. Элемент появиться резко.

Поэтому мы поступим по-другому. Смотрите этот код:

**.block2{**

**opacity: 0;**

**}**

**.block1:hover+.block2{**

**opacity: 1**

**}**

Мы использовали еще одно CSS3-свойство – **opacity**. Оно задает прозрачность элемента, которая прописывается от 0 (полностью прозрачный) до 1 (полностью непрозрачный) элемент.

Сначала полностью убираем второй блок от глаз, делая его прозрачным. Теперь нужно сделать так, чтобы при наведении мышки на первый блок плавно появлялся второй. Для этого используется такой вот интересный комбинированный селектор:

**.block1:hover+.block2{**

**opacity: 1**

**}**

То есть само изменение стилей происходит при наведении на первый блок, а изменяется непосредственно второй. Благодаря прописанному ранее **transition** он появится не резко, а плавно, что нам с вами и нужно. Изменяя значение transition можно сделать появление более резким или наоборот еще больше замедлить его.

Вот с такой вот скоростью появляется блок при выставленном значении **transition: 1s**. Кстати, значение можно задавать и в миллисекундах, если хотите.

Наведите курсор на блок

опа, я

### **1. Изменение цвета при наведении курсора**

Когда-то реализация такого эффекта была довольно кропотливой работой, с математическими вычислениями определенных значений RGB. Сейчас же достаточно записать стиль CSS, в котором необходимо добавить к селектору псевдокласс **:hover** и задать фоновый цвет, который плавно (за 0,3 секунды) заменит собой исходный цвет фона при наведении курсора на блок:

|  |  |
| --- | --- |
| 123 | .color:hover {  background:#53ea93;} |

### **2. Появление рамки**

Интересная и яркая трансформация — внутренняя рамка, плавно появляющаяся при наведении мыши. Хорошо подойдет для украшения различных кнопок. Чтобы добиться такого эффекта, используем псевдокласс **:hover** и свойство **box-shadow** с параметром **inset** (задает тень внутри элемента). Кроме этого, потребуется задать растяжение тени (в нашем случае это **23px**) и её цвет:

|  |  |
| --- | --- |
| 123 | .border:hover {  box-shadow: inset 0 0 0 23px #53ea93;} |

### **3. Свинг**

Данная CSS анимация — исключение, т. к. здесь свойство transition не используется. Вместо него мы задействовали:

* **@keyframes** — базовую директиву для создания покадровой CSS-анимации, которая позволяет делать т. н. раскадровку и описывать анимацию в виде списка ключевых моментов;
* **animation** и **animation-iteration-count** — свойства для задания параметров анимации (длительность и скорость) и количества циклов (повторов). В нашем случае повтор 1.

|  |  |
| --- | --- |
| 123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555657585960 | @-webkit-keyframes swing {  15% {    -webkit-transform: translateX(9px);    transform: translateX(9px);  }  30% {    -webkit-transform: translateX(-9px);    transform: translateX(-9px);  }  40% {    -webkit-transform: translateX(6px);    transform: translateX(6px);  }  50% {    -webkit-transform: translateX(-6px);    transform: translateX(-6px);  }  65% {    -webkit-transform: translateX(3px);    transform: translateX(3px);  }  100% {    -webkit-transform: translateX(0);    transform: translateX(0);  }} @keyframes swing {  15% {    -webkit-transform: translateX(9px);    transform: translateX(9px);  }  30% {    -webkit-transform: translateX(-9px);    transform: translateX(-9px);  }  40% {    -webkit-transform: translateX(6px);    transform: translateX(6px);  }  50% {    -webkit-transform: translateX(-6px);    transform: translateX(-6px);  }  65% {    -webkit-transform: translateX(3px);    transform: translateX(3px);  }  100% {    -webkit-transform: translateX(0);    transform: translateX(0);  }} .swing:hover {  -webkit-animation: swing 0.6s ease;  animation: swing 0.6s ease;   -webkit-animation-iteration-count: 1;   animation-iteration-count: 1;} |

### **4. Затухание**

Эффект плавного затухания — это, по сути, обычное изменение прозрачности элемента. Анимация создается в два этапа: сначала необходимо установить первоначальное состояние прозрачности **1** — то есть полная непрозрачность, после чего указать её значение при наведении мыши — **0.6**:

|  |  |
| --- | --- |
| 1234567 | .fade {  opacity: 1;} .fade:hover {  opacity: 0.6;} |

Для противоположного результата поменяйте значения местами:

### **5. Увеличение**

Чтобы при наведении курсора блок увеличивался, мы воспользуемся свойством **transform** и зададим ему значение **scale(1.2)**. При этом блок увеличится на 20 процентов с сохранением своих пропорций:

|  |  |
| --- | --- |
| 12345 | .grow:hover {  -webkit-transform: scale(1.2);  -ms-transform: scale(1.2);  transform: scale(1.2);} |

### **6. Уменьшение**

Уменьшить элемент так же просто, как и увеличить. Если в пятом пункте для увеличения масштаба нам необходимо было указать значение больше, чем 1, то для уменьшения блока мы просто укажем значение, которое будет меньше единицы, например, **scale(0.7)**. Теперь при наведении мыши блок будет пропорционально уменьшаться на 30 процентов от своего первоначального размера:

|  |  |
| --- | --- |
| 12345 | .shrink:hover {  -webkit-transform: scale(0.7);  -ms-transform: scale(0.7);  transform: scale(0.7);} |

### **7. Трансформация в круг**

Одна из часто используемых анимаций — прямоугольный элемент, который при наведении курсора преобразовывается в окружность. С помощью свойства CSS **border-radius**, использованного в паре с **:hover**и **transition**, это можно реализовать без проблем:

|  |  |
| --- | --- |
| 123 | .circle:hover {  border-radius: 70%;} |

### **8. Вращение**

Забавный вариант анимации — поворот элемента на определенное количество градусов. Для этого нам снова понадобится свойство **transform**, но уже с другим значением — **rotateZ(20deg)**. При таких параметрах блок будет повернут на 20 градусов по часовой стрелке относительно оси Z:

|  |  |
| --- | --- |
| 12345 | .rotate:hover {  -webkit-transform: rotateZ(20deg);  -ms-transform: rotateZ(20deg);  transform: rotateZ(20deg);} |

### **9. 3D тень**

Мнения дизайнеров расходятся в том, уместен ли данный эффект во флэт-дизайне. Тем не менее, эта CSS3 анимация является весьма оригинальной и также используется на веб-страницах. Добиваться трехмерного эффекта будем с помощью уже знакомых нам свойств **box-shadow** (создаст многослойную тень) и **transform** с параметром **translateX(-7px)** (обеспечит сдвиг блока по горизонтали влево на 7 пикселей):

|  |  |
| --- | --- |
| 123456789101112 | .threed:hover {  box-shadow:    1px 1px #53ea93,    2px 2px #53ea93,    3px 3px #53ea93,    4px 4px #53ea93,    5px 5px #53ea93,    6px 6px #53ea93,    7px 7px #53ea93;  -webkit-transform: translateX(-7px);  transform: translateX(-7px);} |