**Позиционирование элементов. Введение**

Отлично! Вы научились строить сетки страниц, познакомились с методологией БЭМ, освоили семантику и эффекты по наведению мыши. Можете верстать сайты, которые понятны и легко поддерживаются. Но иногда приходится сталкиваться с дизайном, в котором элементы расположены нестандартно: например, фиксируются относительно страницы или располагаются один поверх другого.

Осуществить такое на флексбоксах без использования хитростей не получится. На помощь приходит позиционирование элементов.

По умолчанию все элементы на странице «связаны» между собой и реагируют, когда соседний элемент изменяют. Но на многих сайтах, например, шапка остаётся на экране даже когда вы прокручиваете страницу вниз. Такое поведение шапки выбивается из общего поведения элементов. Это сделано благодаря позиционированию — управлению взаимным расположением элементов в потоке. Есть несколько видов позиционирования, и мы расскажем про каждый.

Вы научитесь располагать элементы в любом месте страницы, создавать фиксированные меню, центрировать всплывающие окна и делать эффекты по наведению мыши — даже на видео.

# Понятие потока (flow) и статическое позиционирование

Когда мы располагаем HTML-элементы на странице, браузер отрисовывает их по заложенным в него инструкциям: блочные элементы — каждый на новой строке, линейные рядом. Элементы не наезжают друг на друга. Если мы добавляем одному из них отступ, то сдвигаются следующие за ним. Таким образом, элементы находятся в связке. Они как будто видят друг друга и реагируют на происходящие с соседями перемены.

Такая связка элементов, их зависимость друг от друга называется flow, поток. По умолчанию все элементы находятся в потоке.

Взаимное расположение элементов в макете называется позиционированием. CSS-свойство position определяет, как браузер расположит элемент в потоке.

По умолчанию для всей страницы применяется статическое позиционирование, которое описывается правилом:

Скопировать кодCSS

position: static;

Это значит, что элемент:

1. находится в потоке; все другие элементы знают, где он расположен, и реагируют на его смещение;
2. перемещается визуально только за счёт отступов. Так как отступы — часть блочной модели элемента, то с точки зрения браузера мы не двигаем элемент, а меняем его размеры за счёт отступов.

**Относительное позиционирование**

Когда элемент остаётся в потоке, но смещается относительно самого себя, это называют относительное позиционирование. Оно задаётся правилом position: relative;.

Ключевые моменты этого способа позиционирования:

1. Поскольку элемент остаётся в потоке, отведённое под него пространство видят другие элементы.
2. Элемент не отступает от соседей, как при статическом позиционировании, а действительно смещается относительно первоначального положения соответствующими инструкциями:

top — от верхней границы блока;

right — от правой границы;

bottom — от нижней границы;

left — от левой границы.

Скопировать кодCSS

div {

position: relative;

top: 20px;

right: 40px;

bottom: 10px;

left: 10px;

}

# Фиксированное позиционирование

Третий вид позиционирования — фиксированный — задаётся правилом position: fixed;.

В корне отличается от изученных ранее тем, что фиксированное позиционирование вырывает элемент из потока. Это значит, что пространство, отведённое под элемент, займут следующие за ним элементы в потоке. А фиксированный элемент заслонит их, расположившись поверх соседей.

Положение этого элемента указывается относительно окна браузера:

Скопировать кодCSS

div {

position: fixed;

top: 10px;

left: 40px;

}

Такой код разместит элемент в левом верхнем углу окна, на 10 пикселей ниже верхней границы и на 40 правее левой. При скролле страницы этот блок остаётся на месте, поскольку фиксированное позиционирование закрепляет элемент относительно окна браузера, а не относительно документа. Похоже на корабль, бросивший якорь среди реки.

Среди свойств top, right, left, bottom существует приоритет: top и left важнее bottom и right .

Пример:

Скопировать кодCSS

div {

position: fixed;

top: 10px;

left: 40px;

bottom: 50px;

right: 30px;

}

Этот блок будет спозиционирован фиксированно в левом верхнем углу, несмотря на указанные ниже значения right и bottom . Это работает с любым типом позиционирования, кроме static.

Имейте в виду: без инструкций о месторасположении вырванный из потока блок может вообще потеряться за пределами экрана. Поэтому любому элементу, который вырван из потока, обязательно задавайте определённую позицию по двум направлениям — горизонтальному и вертикальному.

# Абсолютное позиционирование

Последний тип позиционирования, который мы изучим — абсолютное. Это один из самых распространённых типов позиционирования. Он задаётся правилом position: absolute;.

Начнём с того, что элемент, спозиционированный абсолютно, тоже вырван из потока, то есть его пространство в потоке заняли элементы, которые следовали за ним.

В отличие от фиксированного позиционирования, абсолютно спозиционированный элемент может вести себя по-разному в зависимости от контекста, в котором находится.

Первый случай — если его родительский элемент неспозиционирован, имеет заданное по умолчанию свойство position: static;. Тогда интересующий нас блок станет искать ближайшего сверху по иерархии прародителя, чтобы спозиционироваться относительно его границ.

И если не найдёт, расположится относительно границ body. То есть в неспозиционированном окружении код:

Скопировать кодCSS

div {

position: absolute;

top: 30px;

left: 40px;

}

расположит блок на 30 пикселей ниже верхней границы body и на 40 пикселей правее левой.

В отличие от фиксированного позиционирования, элемент не останется на месте при прокрутке, так как он расположен относительно тела документа, а не окна браузера.

Второй случай: родительский элемент спозиционирован, то есть имеет значение свойства position, отличное от static.

Тогда наш спозиционированный абсолютно блок будет размещён согласно инструкциям top, right, bottom, left относительно границ родителя.

Если .parent — родительский элемент для .child, такой CSS-код расположит блок .child в правом нижнем углу блока .parent:

Скопировать кодCSS

.parent {

position: relative;

}

.child {

position: absolute;

bottom: 0;

right: 0;

}

Попробуйте изменять разные значения в виджете. Обратите внимание, как ведет себя блок в спозиционированном и неспозиционированном родителе. Попробуйте указать противоположные значения, например left и right, чтобы запомнить, что в приоритете.

Закрепим навык на изображениях в меню, они обозначены классом menu\_\_picture.

Для начала сделаем их видимыми при наведении мыши на элемент menu\_\_item.

Здесь понадобится комплексный селектор:

Скопировать кодCSS

.menu\_\_item:hover .menu\_\_picture {

}

Этот селектор означает «элемент menu\_\_picture, вложенный в menu\_\_item, на который наведена мышь».

Измените для этого селектора свойство display на значение block.

# z-index

Когда при позиционировании мы вырываем элементы из потока, они могут перекрывать друг друга. Нужно определить правило, кто из них должен оказаться на поверхности, ближе к пользователю.

Для управления близостью элементов используется свойство z-index. Его название говорит о том, что задействовано третье измерение. Позиционирование идёт уже не по ширине (ось x) и не по высоте (ось y), а по глубине, т.е. по оси z. Значения z-index — целые числа. Чем больше число, тем ближе к нам окажется элемент. По умолчанию у всех элементов одинаковый z-index, равный 0.

Например:

Скопировать кодCSS

.background {

z-index: -1;

}

.first-block {

z-index: 0;

}

.second-block {

z-index: 1;

}

.third-block {

z-index: 99;

}

.fourth-block {

z-index: 100;

}

В этом коде блок background — самый дальний.

Далее блоки расположены по мере их визуальной близости к пользователю.

z-index не действует на элемент со свойством position: static — или если свойство position вообще не задано. Из этого правила есть исключение: если элемент находится во flex- или grid-контейнере (у родителя задано display: flex или display: grid), то z-index сработает даже когда позиция элемента — static.