Позиционирование элементов

Комментарии в коде

Чтобы помочь разобраться в коде, его снабжают комментариями. Браузер не воспринимает эти тексты как команды, если они заключены в особые конструкции.

```
<!-- комментарий, этот текст не будет исполняться как команда браузеру -->
```

В CSS комментарии выглядят иначе:

```
/* комментарий, этот текст не будет исполняться как команда браузеру */
```

Комментарий может располагаться на одной строке или на нескольких:

```
<!-- комментарий,
этот текст
не будет исполняться
как команда браузеру -->
```

```
/* комментарий,
этот текст
не будет исполняться
как команда браузеру */
```

Конструкция /* */ для многострочного комментария используется и в языке JavaScript. Однострочные комментарии в JS-коде предваряются двумя слешами:

```
// Объявляем функцию-обработчик события под кодом, меняющим цвет.
```

Понятие потока (flow) и статическое позиционирование

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 поток (англ. flow). 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:

```
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 счет отступов. Так как отступы — часть блочной модели элемента, то с точки зрения браузера мы не двигаем элемент, а меняем его размеры за счет 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 остается 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;

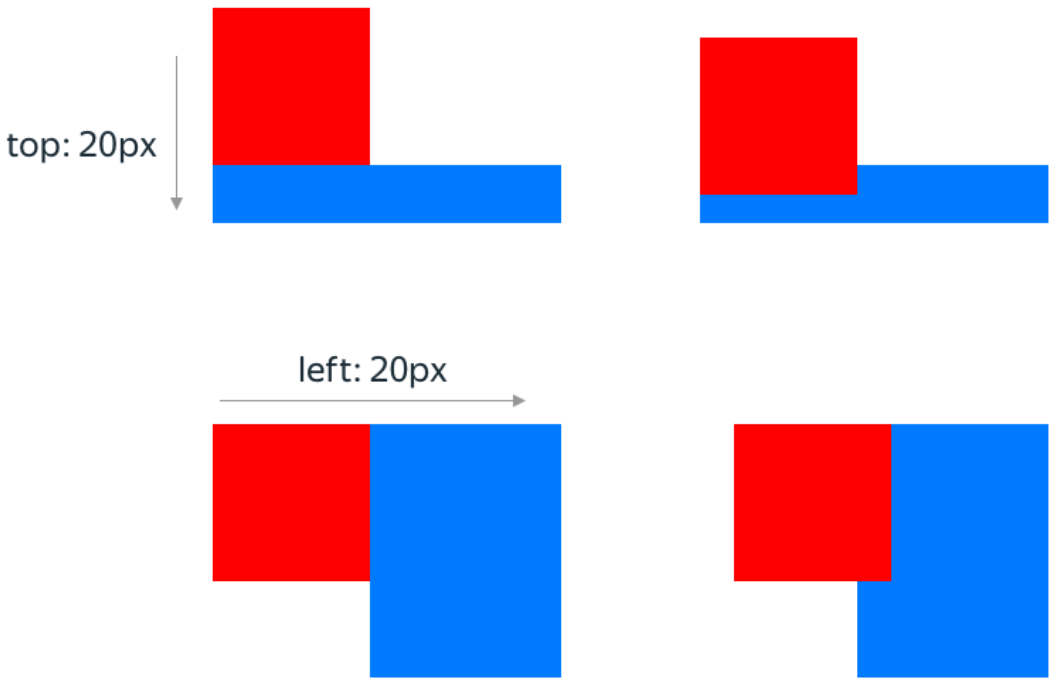
   right — от правой границы;

   bottom — от нижней границы;

   left — от левой границы.

```
div {
  position: relative;
  top: 20px;
  right: 40px;
  bottom: 10px;
  left: 10px;
}
```

Сдвигается только сам элемент, а выделенная под него область в потоке остаётся на месте:



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:

```
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.

Браузер в первую очередь "смотрит" в верхний левый угол, поэтому среди свойств top, right, left, bottom существует **приоритет**.

```
div {
  position: fixed;
  top: 10px;
  left: 40px;
  bottom: 50px;
  right: 30px;
}
```

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.

Абсолютное позиционирование

Это один из самых распространённых типов позиционирования. Он задаётся правилом **position: absolute;** 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:

```
div {
  position: absolute;
  top: 30px;
  left: 40px;
}
```

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

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**:

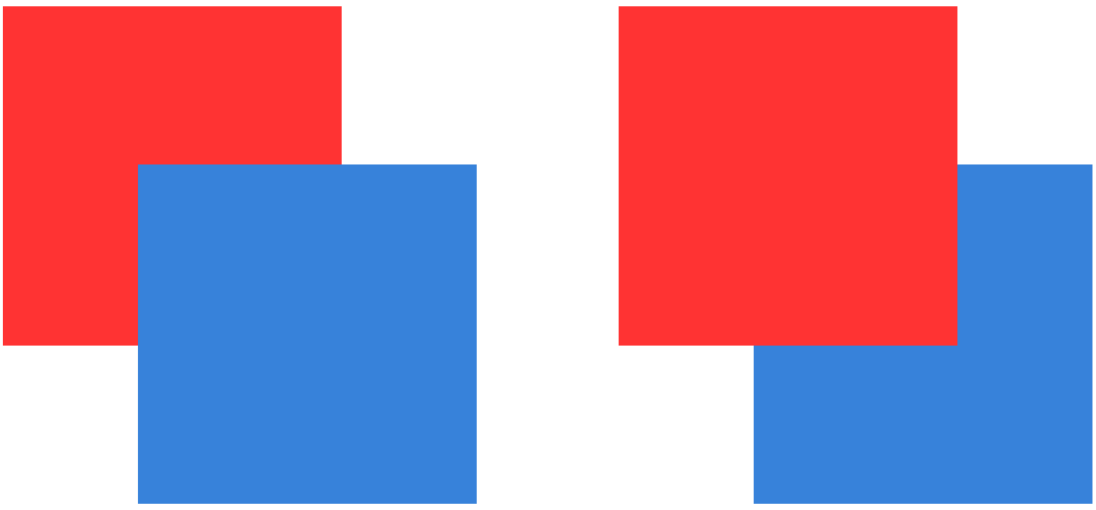
```
.parent {
  position: relative;
}

.child {
  position: absolute;
```

```
  bottom: 0;
  right: 0;
}
```

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.

Например:



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:

```
.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.