Вёрстка многослойных элементов интерфейса
Выполнен на 51%

- Введение
- Углублённая теория −
  - Позиционирование
  - Как работает position: sticky
  - z-index
  - Многослойность с помощью CSS Grid Layout
  - Про vh, vw и другие единицы измерения
  - Словарь терминов
- Методика вёрстки многослойных элементов +
- Кейс 1, лёгкий уровень +
- Кейс 2, лёгкий уровень +
- Кейс 3, лёгкий уровень
- Кейс 4, лёгкий уровень
- Кейс 5, средний уровень +
- Кейс 6, средний уровень
- Кейс 7, средний уровень
- Кейс 8, сложный уровень +
- Кейс 9, сложный уровень
- Кейс 10, сложный уровень

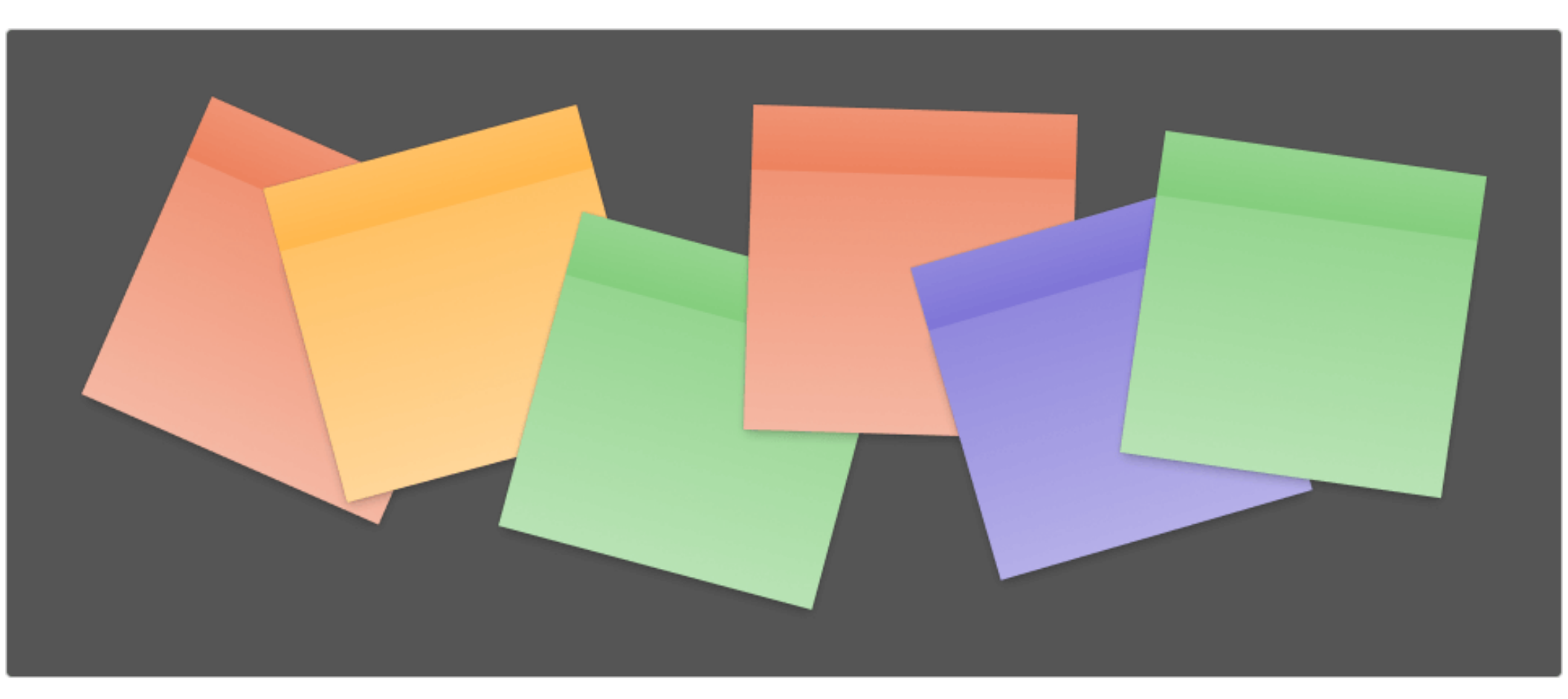
z-index

Если в одном месте страницы оказываются несколько абсолютно спозиционированных элементов, они могут перекрывать друг друга. Какие элементы будут перекрыты, а какие нет, определяется индексом позиционированного уровня (`z-index`).

Сайт с многослойными элементами

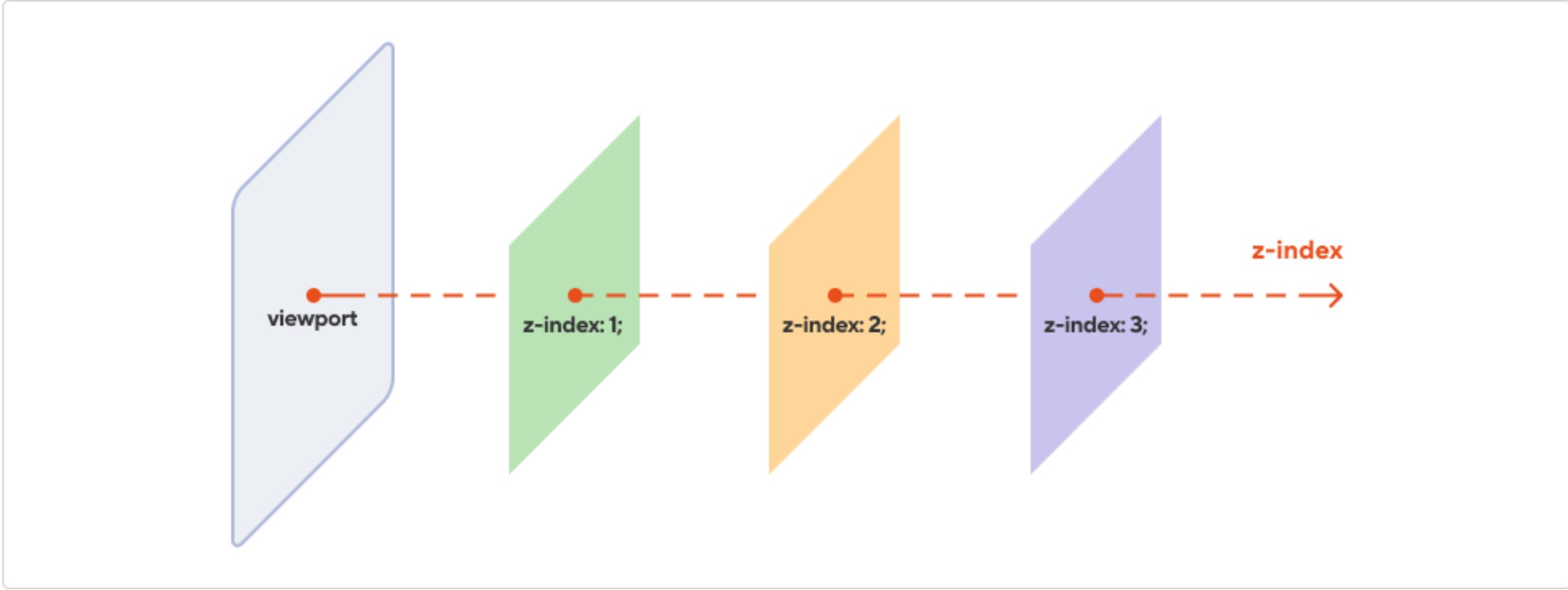
Если вы знакомы с понятием «стопок слоёв» в графических редакторах (*Photoshop*, *Figma* или *Sketch*), то представляете, как работает индекс позиционированного уровня: он определяет порядок, в котором элементы накладываются друг поверх друга на странице.

Представьте веб-страницу как лист бумаги, а спозиционированный элемент — как стикер, приклеиваемый поверх. Каждый раз, когда вы добавляете спозиционированный элемент, вы «приклеиваете» на неё стикер. Если вы добавляете относительно спозиционированный элемент или элемент с «липким» позиционированием, ничего страшного не произойдёт: они займут положенное им место в потоке. Если это абсолютно спозиционированный элемент, невидимый другим элементам, вы рискуете перекрыть какой-либо контент на странице.



Спозиционированные элементы на странице

Обычно порядок наложения элементов соответствует их порядку в HTML-разметке страницы. Но порядком наложения можно управлять, используя CSS-свойство `z-index`. В значении свойства указывается число. Чем больше значение, тем выше в «стопке слоёв» появится элемент.



Ось z-index

Попробовать CSS-свойство `z-index` в действии можно в наших тренажёрах.

У CSS-свойства `z-index` есть ограничение, он работает только с позиционированными элементами (`position: absolute`, `position: relative`, `position: fixed` и `position: sticky`).

− Замечание

`z-index` ещё используется для позиционирования элементов в флексбоксах и CSS-гридах.

Положение элементов по умолчанию

Для начала давайте разберёмся с тем, как браузер по умолчанию позиционирует элементы, когда не указаны значения `z-index`:

1. Первым всегда идёт корневой элемент страницы `<html>`.
2. Затем браузер размещает неспозиционированные элементы в том порядке, в котором они находятся в разметке страницы.
3. После добавляются спозиционированные элементы в том порядке, в котором они находятся в разметке страницы.

Неспозиционированный элемент — это элемент со значением `position: static` по умолчанию. Спозиционированный элемент — это элемент с любым другим значением свойства `position`: `absolute` (абсолютное), `relative` (относительное), `sticky` (липкое) или `fixed` (фиксированное).

Z-index и контекст наложения

Подробнее разберём, как управлять положением элементов в «стопке» с помощью `z-index`.

Тут всё просто, спозиционированный элемент с более высоким значением `z-index` будет отображаться перед спозиционированным элементом с более низким значением `z-index`. При этом не важно, как они расположены в разметке страницы, на одном или на разных уровнях дерева элементов.

Рассмотрим пример. У нас есть три блока (розовый, голубой и зелёный), розовый блок содержит дочерний элемент — блок оранжевого цвета.

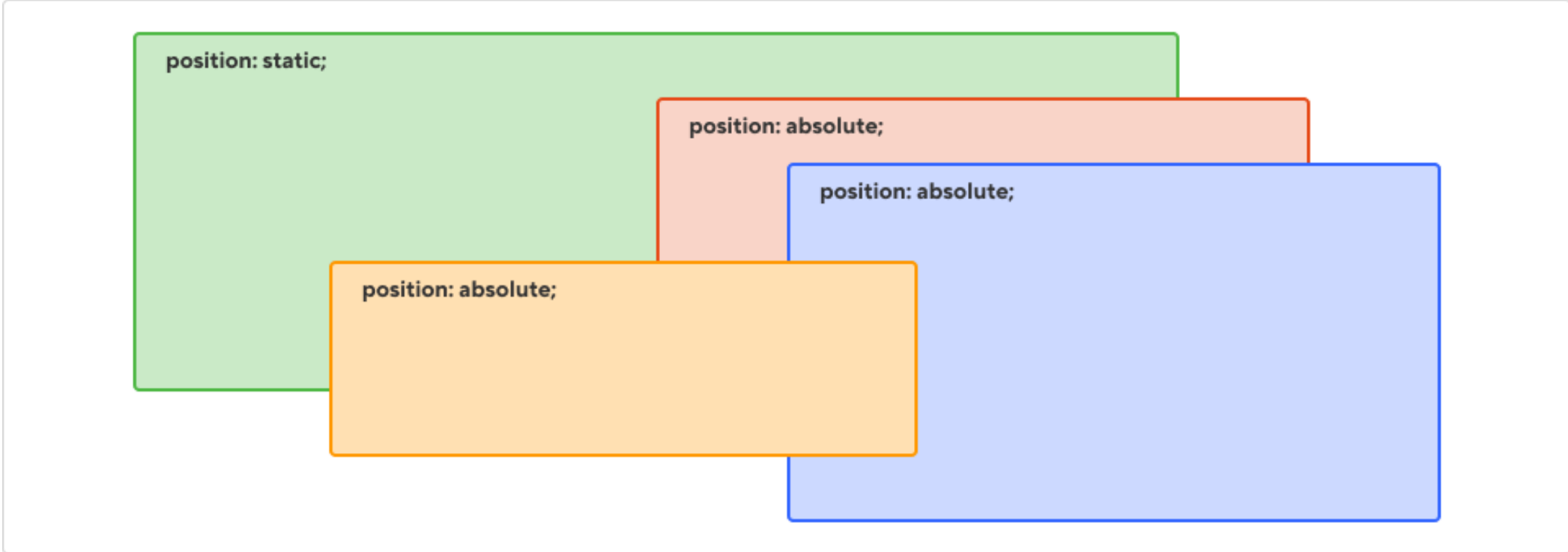
HTML:

```
<div class="pink">
  <div class="orange"></div>
</div>
<div class="blue"></div>
<div class="green"></div>
```

CSS:

```
.blue,
.pink,
.orange {
  position: absolute;
}

.blue {
  z-index: 2;
}

.orange {
  z-index: 3;
}

.green {
  z-index: 100; /* не действует, так как зелёный блок неспозиционированный */
}
```

Блоки `blue`, `pink` и `orange` являются абсолютно спозиционированными. Установим для блока `blue` `z-index: 2`, для блока `orange` `z-index: 3`, а для блока `green` `z-index: 100`. В результате, не смотря на то, что блоки на разных уровнях дерева элементов, на переднем фоне окажется блок `orange` — это спозиционированный элемент с наибольшим значением CSS-свойства `z-index`. На блок `green` свойство `z-index` не действует.



Перед синим блоком отображается оранжевый блок, потому что у него более высокий `z-index`

Если нужно, чтобы элемент отображался в самом верхнем слое, мы делаем элемент спозиционированным и устанавливаем самое большое значение CSS-свойства `z-index`.

Но эта схема не работает с *контекстом наложения (stacking context)*. Контекст наложения формируется, когда к элементу применяется одно из перечисленных CSS-свойств:

- z-index (любое значение, отличное от `auto`);
- filter (фильтр);
- opacity (непрозрачность);
- transform (трансформация);
- элемент является дочерним элементом `flex` контейнера (flexbox), со значением `z-index`, отличным от `auto`;
- элемент является дочерним элементом `grid` контейнера (grid), со значением `z-index`, отличным от `auto`.

− Замечание

Описание контекста наложения и полный перечень свойств, которые его создают можно найти в спецификации.

В этом случае установка большого значения CSS-свойства `z-index` подействует только в текущем контексте наложения.

Если провести аналогию с графическим редактором *Photoshop*, спозиционированный элемент со свойством `z-index` — это слой, а контекст наложения — это группа слоёв. Вы можете как угодно изменять порядок наложения слоёв в пределах группы. Также вы можете менять порядок наложения самих групп. Однако вы не можете наложить определённый слой из нижней группы поверх слоя верхней группы — разве что переместить наверх всю нижнюю группу, либо извлечь нужный слой из этой группы.

Разберём на примере. Усложним предыдущую задачу.

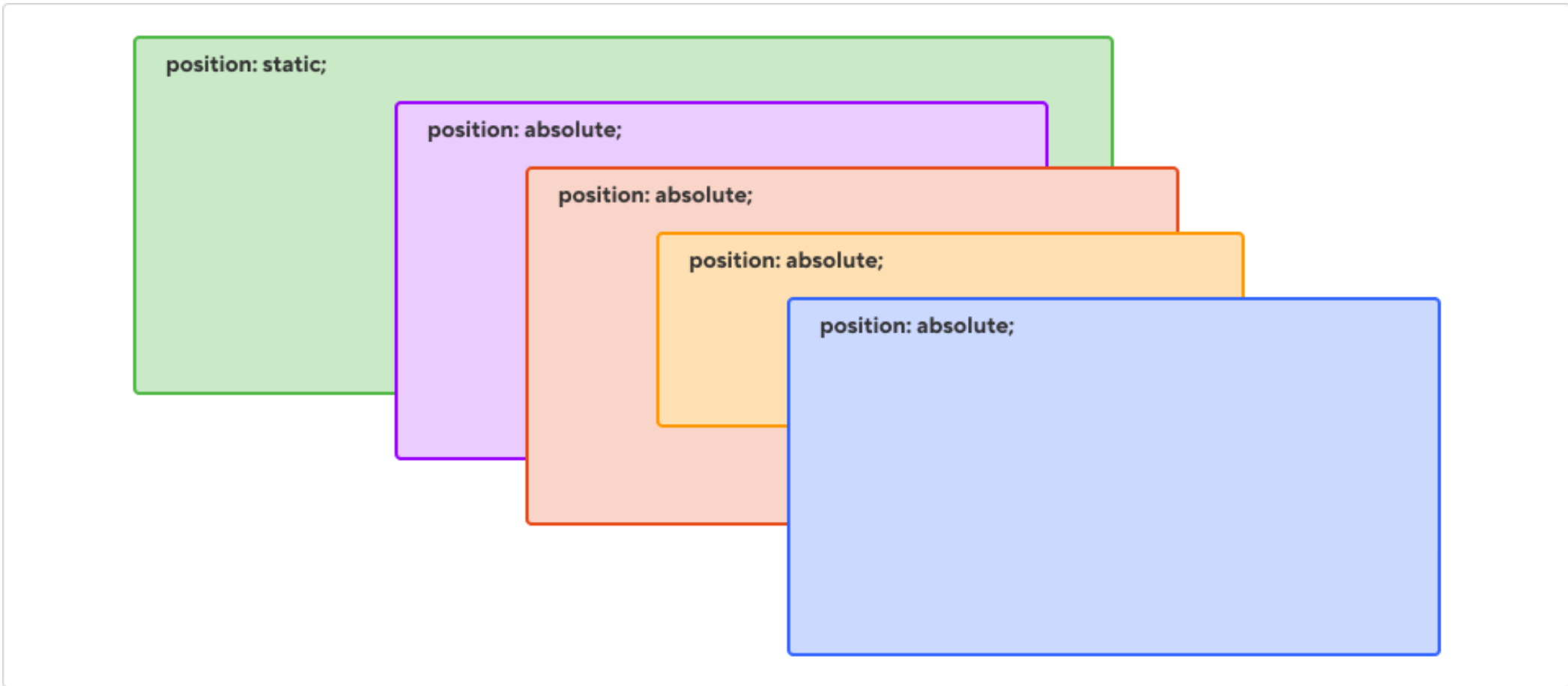
Добавим ещё один позиционированный блок `purple` в макет, который мы хотим разместить за розовым. Обновим наш код, чтобы он выглядел следующим образом:

HTML:

```
<div class="pink">
  <div class="orange"></div>
</div>
<div class="blue"></div>
<div class="purple"></div>
<div class="green"></div>
```

CSS:

```
.blue,
.pink,
.orange,
.purple {
  position: absolute;
}

.purple {
  z-index: 0;
}

.pink {
  z-index: 1; /* формируется контекст наложения, который определяет порядок всех его дочерних элементов
*/
}

.blue {
  z-index: 2;
}

.orange {
  z-index: 3;
}

.green {
  z-index: 100; /* не действует, так как зелёный блок неспозиционированный */
}
```



Новое положение блоков

Розовый блок отображается перед фиолетовым, как и ожидалось, но что же случилось с оранжевым блоком? Он оказался за синим, хотя у него более высокий `z-index`. Это связано с тем, что мы определили значение `z-index` для блока `pink`. Тем самым создали *контекст наложения*. Теперь мы не сможем расположить оранжевый блок перед синим, потому что они находятся в разных контекстах наложения. У блока `orange` `z-index` для окружающих не может быть больше, чем установлен у его родителя. В нашем примере для оранжевого блока по отношению к синему блоку действует `z-index: 1;` (значение его родителя), в то время как у блока `blue` `z-index: 2`.

Влияние свойства `z-index` распространяется как на сам элемент, так и на его дочерние элементы, то есть поднимая родительский элемент выше по *оси Z*, вы поднимаете и его дочерние элементы. Если мы хотим поместить оранжевый блок перед синим, нам нужно увеличить индекс позиционированного уровня для его родителя, то есть для блока `pink`.

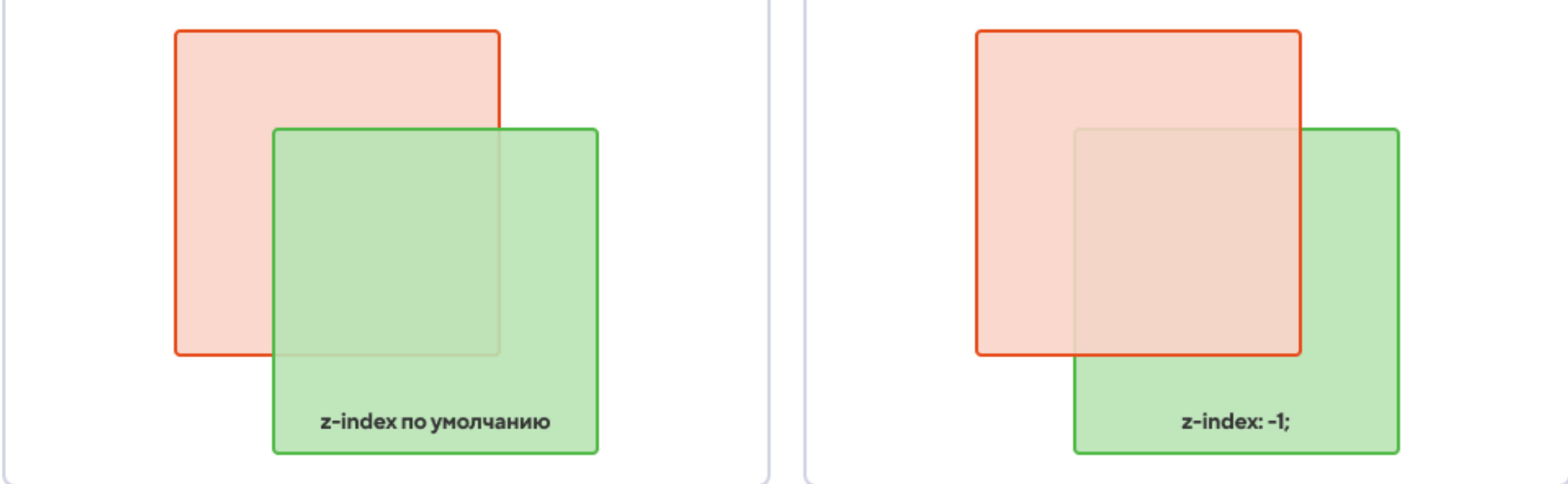
Отрицательные значения z-index

Свойству `z-index` присваивают не только положительные значения, но и отрицательные, например `z-index: -1;`. За счёт отрицательного значения `z-index` элемент будет находится позади его непосредственного родителя или любого из его предков.

Пример:

```
<div class="red-block">
  <div class="green-block"></div>
</div>
```

```
.red-block {
  position: relative;
  width: 200px;
  height: 200px;
  background-color: rgba(255, 0, 0, 0.7);
}

.green-block {
  position: absolute;
  top: 40px;
  left: 40px;
  width: 200px;
  height: 200px;
  background-color: rgba(0, 255, 0, 0.7);
  z-index: -1;
}
```



положение элемента с `z-index` по умолчанию и с `z-index: -1;`

‹ Как работает position: sticky
Многослойность с помощью CSS Grid Layout ›

html academy
Sk Участник

Практикум
Тренажёры
Подписка
Для команд и компаний
Учебник по PHP

Профессии
Фронтенд-разработчик
JavaScript-разработчик
Фулстек-разработчик

Услуги
Работа наставником
Для учителей
Стать автором

Курсы
HTML и CSS. Профессиональная вёрстка сайтов
HTML и CSS. Адаптивная вёрстка и автоматизация
JavaScript. Профессиональная разработка веб-интерфейсов
JavaScript. Архитектура клиентских приложений
React. Разработка сложных клиентских приложений
Node.js. Профессиональная разработка REST API
Node.js и Nest.js. Микросервисная архитектура
TypeScript. Теория типов
Алгоритмы и структуры данных
Паттерны проектирования
Webpack
Vue.js 3. Разработка клиентских приложений
Git и GitHub
Анимация для фронтендеров

Блог
С чего начать
Шпаргалки для разработчиков
Отчеты о курсах

Информация
Об Академии
О центре карьеры

Остальное
Написать нам
Мероприятия
Форум

Соглашение
Конфиденциальность
Сведения об образовательной организации
Лицензия № 4696