



CSS

Cascading Style
Sheets
(Каскадные
таблицы стилей.)



ПЛАН

1

Единицы измерений, специфичность, !important.

2

Box-sizing, float.

3

Математические функции.

4

Flexbox.

ЕДЕНИЦЫ ИЗМЕРЕНИЙ

CSS предоставляет различные единицы измерений, которые используются для определения размеров, расстояний и других значений веб-элементов.

Длина- это число, за которым следует единица измерения, например 10px, 2em



В CSS существуют два типа единиц измерения: абсолютные и относительные. Эти единицы используются для задания размеров, отступов, шрифтов и других значений в CSS свойствах.

Абсолютные единицы представляют фиксированное значение, которое не зависит от контекста или размеров экрана.

- Относительные единицы измерения основаны на отношении к другим элементам или размерам окна просмотра.
- Относительные единицы позволяют создавать адаптивные и гибкие макеты, которые могут меняться в зависимости от контекста.

1. Между числом и единицей измерения не может быть пробела. Однако, если значение равно 0, единица измерения может быть опущена.
2. Для некоторых свойств CSS допускаются отрицательные длины.

1. **px** - это абсолютная единица измерения, которая всегда указывает на конкретное количество пикселей на экране. Таким образом, значение в пикселях всегда будет фиксированным, независимо от того, какое устройство или разрешение экрана используется.
2. **em** - это относительная единица измерения, которая зависит от размера шрифта родительского элемента. Когда вы указываете размер шрифта в единицах em, его значение будет умножаться на размер шрифта родительского элемента. Таким образом, если размер шрифта родительского элемента изменится, размер шрифта дочернего элемента, указанного в em, также изменится.
3. **rem** - это относительная единица измерения, которая также зависит от размера шрифта, но относится к корневому элементу страницы. Это значит, что значение rem всегда будет зависеть от размера шрифта, установленного для элемента html. Как и в случае с em, если размер шрифта изменится, значение rem также изменится.
4. **vw** и **vh** - это относительные единицы измерения, которые зависят от размера видимой области браузера. 100vw равен 100% ширины видимой области браузера, а 100vh - 100% высоты. Это позволяет создавать адаптивные макеты, которые будут отображаться одинаково на всех устройствах, независимо от разрешения экрана.

1. **px** - это абсолютная единица измерения, которая всегда указывает на конкретное количество пикселей на экране. Таким образом, значение в пикселях всегда будет фиксированным, независимо от того, какое устройство или разрешение экрана используется.
2. **em** - это относительная единица измерения, которая зависит от размера шрифта родительского элемента. Когда вы указываете размер шрифта в единицах em, его значение будет умножаться на размер шрифта родительского элемента. Таким образом, если размер шрифта родительского элемента изменится, размер шрифта дочернего элемента, указанного в em, также изменится.
3. **rem** - это относительная единица измерения, которая также зависит от размера шрифта, но относится к корневому элементу страницы. Это значит, что значение rem всегда будет зависеть от размера шрифта, установленного для элемента html. Как и в случае с em, если размер шрифта изменится, значение rem также изменится.
4. **vw** и **vh** - это относительные единицы измерения, которые зависят от размера видимой области браузера. 100vw равен 100% ширины видимой области браузера, а 100vh - 100% высоты. Это позволяет создавать адаптивные макеты, которые будут отображаться одинаково на всех устройствах, независимо от разрешения экрана.

СПЕЦИФИЧНОСТЬ

Специфичность (specificity) в CSS определяет, какой набор правил стилей будет применяться к элементу, когда у него есть несколько правил селекторов, которые конфликтуют между собой.

- Если есть два или более правил CSS, которые указывают на один и тот же элемент, селектор с наивысшим значением специфичности "выиграет", и его объявление стиля будет применено к этому HTML-элементу.
- Думайте о специфичности как о балле/ранге, который определяет, какое объявление стиля в конечном итоге применяется к элементу.

Каждый CSS-селектор имеет свое место в иерархии специфичности.

Существует четыре категории, которые определяют уровень специфичности селектора:

1. **Встроенные стили** - Пример: `<h1 style="color: pink;">`
2. **IDs** - Пример: `#navbar`
3. **Классы, псевдо-классы, селекторы атрибутов** - Пример: `.test, :hover, [href]`
4. **Элементы и псевдо-элементы** - Пример: `h1, ::before`

Как рассчитать специфичность?

Запомните, как вычислять специфичность!

Начните с 0, добавьте 100 для каждого значения идентификатора, добавьте 10 для каждого значения класса (или селектора псевдокласса или атрибута), добавьте 1 для каждого селектора элемента или псевдоэлемента.

Примечание: Встроенный стиль получает значение специфичности 1000, и ему всегда присваивается наивысший приоритет!< / p>

Примечание 2: Из этого правила есть одно исключение: если вы используете !important, оно даже переопределит встроенное стили!

В таблице ниже приведены некоторые примеры того, как рассчитать значения специфичности:

| Селектор | Значение специфичности | Расчет |
|--------------------------|------------------------|-----------------------------------------|
| p | 1 | 1 |
| p.test | 11 | 1 + 10 |
| p#demo | 101 | 1 + 100 |
| <p style="color: pink;"> | 1000 | 1000 |
| #demo | 100 | 100 |
| .test | 10 | 10 |
| p.test1.test2 | 21 | 1 + 10 + 10 |
| #navbar p#demo | 201 | 100 + 1 + 100 |
| * | 0 | 0 (универсальный селектор игнорируется) |

Селектор с наивысшим значением специфичности победит и вступит в силу!

!IMPORTANT

`!important` - это ключевое слово в CSS, которое можно добавить к объявлению стиля, чтобы принудительно переопределить другие стили, которые могут быть применены к элементу.

Когда стиль объявлен с использованием `!important`, он имеет наивысший приоритет и будет применяться независимо от специфичности других селекторов или порядка их объявления.

BOX-SIZING

Свойство `box-sizing` в CSS3 определяет, какая модель размеров должна применяться к элементу при расчете его полной ширины и высоты.

По умолчанию в CSS используется модель `content-box`, где ширина и высота элемента определяются только его содержимым (контентом), без учета границы (`border`), отступа (`padding`) и полосы прокрутки (если есть). Это означает, что если у элемента есть заданные границы и отступы, они добавляются к общей ширине и высоте элемента.



Без свойства box-sizing

По умолчанию ширина и высота элемента вычисляется так:

ширина + отступ + граница = фактическая ширина элемента

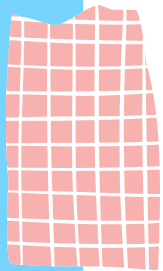
высота + отступ + граница = фактическая высота элемента

Это означает: когда вы устанавливаете ширину/высоту элемента, элемент будет больше, чем вы установили (потому что граница элемента и заполнение добавляются в указанной ширине/высоте элемента).

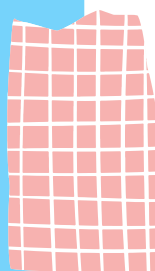
Однако, с использованием свойства `box-sizing`, можно изменить эту модель и управлять тем, каким образом элементы распределяют свое пространство.

Два основных значения для свойства `box-sizing`:

1. `content-box` (значение по умолчанию): Это стандартная модель размеров, где ширина и высота элемента рассчитываются на основе его контента, без учета границы и отступов.
2. `border-box`: При использовании этого значения, размер элемента будет включать в себя границу и отступы, входящие в указанные значения ширины и высоты элемента. Иными словами, размер элемента будет определяться с учетом границы и отступов, и доступное пространство для контента будет уменьшено соответственно.



FLOAT



Свойство **float** в CSS используется для управления позиционированием элемента, позволяя ему "плавать" вокруг других элементов на странице.

Когда элементу задано свойство **float**, он выравнивается либо слева (**float: left**), либо справа (**float: right**) от окружающих его элементов.

Обтекание текстом (Text Wrapping): Элементы, которым задано свойство float, создают обтекание текстом окружающих элементов. Текст будет выравниваться вокруг элемента, подобно тому, как текст обтекает изображения.

Сжатие блочной модели: Когда элементу задано свойство float, его блочная модель сжимается по ширине, чтобы соответствовать размеру содержимого и/или ширины родительского контейнера. Это позволяет элементам плавать рядом друг с другом в строку.

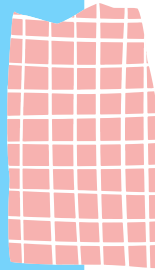
Отрыв элемента от потока: Элементы с float вырываются из обычного потока документа, что означает, что они не занимают место в родительском контейнере и остальные элементы не учитывают их при расположении. Окружающие элементы могут обтекать элементы с float, но могут также перекрывать их, если не хватает места.

Очистка плавающих элементов: Когда элементы с float находятся внутри родительского контейнера, родитель может не расширяться, чтобы охватить все плавающие элементы. Это может привести к различным проблемам с макетом. Для решения этой проблемы используются методы "очистки" (clearing), чтобы родительский контейнер расширялся и правильно размещал плавающие элементы.



Важно отметить, что свойство `float` имеет некоторые ограничения и может вызывать непредсказуемое поведение в некоторых ситуациях. В современном веб-дизайне часто предпочитают использовать другие техники позиционирования, такие как CSS Flexbox и CSS Grid, которые предоставляют более гибкую и предсказуемую модель размещения элементов.

МАТЕМАТИЧЕСКИЕ ФУНКЦИИ



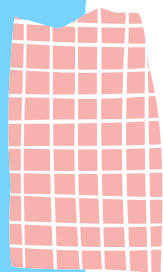
CSS включает несколько математических функций, которые могут быть использованы для выполнения простых математических операций и вычислений непосредственно внутри значений CSS свойств.

calc()

Функция `calc()` позволяет выполнить математические операции с единицами измерения в CSS. Она принимает выражение внутри себя и возвращает результат вычислений. Выражение может включать операторы `+`, `-`, `*`, `/` и скобки для управления порядком операций.

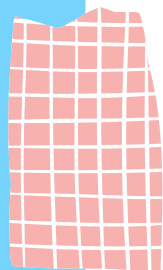
`calc()` позволяет создавать адаптивные макеты, где элементы автоматически меняют свои размеры в зависимости от размеров экрана или родительских контейнеров. Например, можно использовать `calc()` для определения ширины элемента в процентах от ширины родительского контейнера, с учетом фиксированного отступа или границы.

```
#div1 {  
  position: absolute;  
  left: 50px;  
  width: calc(100% - 100px);  
  border: 1px solid black;  
  background-color: yellow;  
  padding: 5px;  
}
```



max()

принимает одно или несколько значений и возвращает наибольшее значение из них.



- Это позволяет выбрать максимальное значение из набора значений, которое затем используется в CSS свойстве.
- max() может быть использована для числовых значений, таких как ширина, высота, отступы и другие.

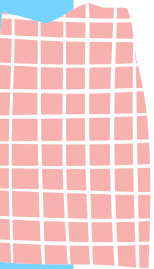
В этом примере ширина элемента будет равна 50% от ширины родительского контейнера или 300 пикселей, в зависимости от того, какое значение больше.

```
#div2 {  
  background-color: yellow;  
  height: 100px;  
  width: max(50%, 300px);  
}
```



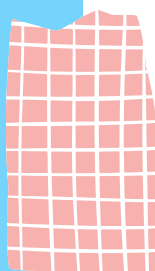
min()

принимает одно или несколько значений и возвращает наименьшее значение из них.

- 
- Это позволяет выбрать минимальное значение из набора значений, которое затем используется в CSS свойстве.
 - min() может быть использована для числовых значений, таких как ширина, высота, отступы и другие.

В этом примере ширина элемента будет равна 50% от ширины родительского контейнера или 200 пикселей, в зависимости от того, какое значение меньше.

```
#div2 {  
  background-color: yellow;  
  height: 100px;  
  width: max(50%, 300px);  
}
```



Функции `max()` и `min()` особенно полезны, когда требуется выбрать максимальное или минимальное значение из набора значений, чтобы адаптировать стиль в зависимости от условий. Они позволяют более гибко управлять значениями свойств CSS и делают стили более динамическими.



**THANK
YOU!**

Have a
great day
ahead.