ПЕРЕМЕННЫЕ И ФУНКЦИИ CSS.

CSS переменные. Функция calc(). Min() и max(). Функция clamp().

CSS переменные

При создании сайта мы используем определенный набор настроек, которые превращают страницы нашего сайта в цельный проект.

Это могут быть цвета, как базовые: черный и белый, так и какие-то особенные: акцентный, кастомный белый, варианты градиентов и др. При этом каждый из цветов может играть самые разные роли: цвет текста, фона или границ, и встречаться в самых разных местах кода.

Или размеры. Единый размер заголовков, отступы внутри кнопок или толщина границы. Все они перекликаются между собой по определенным правилам.

Получается, что смена предпочтений или подбор более подходящего варианта становится процедурой достаточно утомительной и затратной.

CSS переменные позволяют сохранять значение в одном месте, а затем многократно использовать его в любом другом месте. Дополнительным преимуществом являются семантические идентификаторы.

Например: запись **--main-text-color** более понятна, чем **#00ff00**, особенно если этот же цвет используется и в другом контексте.

При этом CSS переменные подчиняются каскаду и наследуют значения от своих родителей.

Итак, **CSS переменные** (пользовательские CSS-свойства) — это сущности, определяемые автором CSS, хранящие конкретные значения, которые можно повторно использовать в документе.

Синтаксис

Объявление переменной:

```
element {
  --имя_переменной: значение;
}
```

Использование переменной:

```
element {
  background-color: var(--имя_переменной);
}
```

Кстати, функцию **var()** нельзя использовать в именах свойств, селекторах или где-либо ещё, кроме как в качестве значений для свойств.

Функции **var()** в качестве **первого аргумента** обязательно нужно передать имя **кастомного** свойства, значение которого нужно получить. Необязательный **второй аргумент** функции используется в качестве **резервного** значения. Если кастомное свойство, указанное в первом аргументе, оказалось недоступным, функция вернёт второе значение.

```
/* red если --my-var не определена */
.one { color: var(--my-var, red);}

/* pink если --my-var и --my-background не определены */
.two {
   color: var(--my-var, var(--my-background, pink));
  }
```

```
/* "--my-background, pink" будет воспринят как значение
в случае, если --my-var не определена */
.three {
   color: var(--my-var, --my-background, pink);
  }
```

Для наших проектов гораздо удобнее будет объявлять переменную в псевдоклассе **:root** в начале файла стилей.

CSS псевдокласс **:root** находит корневой элемент дерева документа. Применимо к HTML, **:root** находит элемент <html> и идентичен селектору по тегу **html**, но его специфичность выше.

```
/* Объявление */
:root {
  --big-size-text: 2em;
}

/* Использование */
element {
  font-size: var(--big-size-text);
}
```

Наследование и возвращаемые значения

Пользовательские свойства могут наследоваться. Это означает, что если не установлено никакого значения для пользовательского свойства на данном элементе, то используется свойство его родителя.

Например, для верстки

```
    <div class="one">
        <div class="two">
            <div class="three"></div>
            <div class="four"></div>
        </div>
    </div>
```

со следующим CSS:

```
.two {
  --test: 10px;
}
.three {
  --test: 2em;
}
```

В результате var(--test) будет задавать значения:

- для элемента class="two": 10px
- для элемента class="three": 2em
- для элемента class="four": 10px (унаследовано от родителя)
- для элемента class="one": недопустимое значение, что является значением по умолчанию для любого пользовательского свойства.

Функция calc()

Функция **calc()** позволяет выполнять некоторые вычисления и использовать их в качестве значений свойств.

Функция **calc()** принимает в качестве параметра одно выражение, а в качестве **значения** используется его **результат**. Выражение может быть любым простым выражением, сочетающим следующие операторы с использованием стандартных правил приоритета операторов:

- **+** Добавление.
- **-** Вычитание.
- ***** Умножение. Хотя бы один из аргументов должен быть числом (целым или с дробной частью).
- **/** Деление. Правая часть должна быть числом (целым или с дробной частью).

Операнды в выражении могут быть любыми значениями длины, как абсолютными, так и относительными. При желании вы можете использовать разные единицы измерения

для каждого значения в выражении. А еще использовать круглые скобки для установления порядка вычислений.

При составлении выражений необходимо следовать следующим правилам:

- Операторы **+** и **-** должны быть **окружены пробелами**. Например, выражение

  calc(50% -8px)

  будет анализироваться как «процент, за которым следует отрицательная длина».

  calc(8px + -50%)

  Это выражение аналогично, будет рассматриваться как «длина, за которой следует оператор сложения и отрицательный процент».

- Операторы * и **/** не требуют пробелов, но для обеспечения единообразия внешнего вида выражений, рекомендуют их добавлять.
- Математические выражения, включающие проценты ширины и высоты в столбцах или строках таблицы (а также их группах и ячейках) как в таблицах с автоматическим, так и с фиксированным макетом, могут обрабатываться так, как если бы они были заданы с помощью значения auto.
- Допускается вложение **calc()** функций, в этом случае внутренние функции рассматриваются как простые круглые скобки.
- Для длин вы не можете использовать 0 в качестве значения; вместо этого вы должны использовать версию с единицей измерения:

  ВЕРНО! margin-top: calc(0px + 20px);

  НЕВЕРНО! margin-top: calc(0 + 20px);

- Функция calc() не может напрямую заменять числовое значение для процентных типов;

ВЕРНО! margin-top: calc(100 / 4)%;

НЕВЕРНО! margin-top: calc(100% / 4);

Если в качестве результата выражения получилось дробное число, а в значении свойства ожидается целое, то значение будет округлено до ближайшего целого числа (в большую сторону).

В случае, когда **calc()** используется для управления размером текста, убедитесь, что одно из значений содержит относительную единицу длины, например:

```
h1 {
   font-size: calc(1.5rem + 3vw);
}
```

Это гарантирует, что размер текста будет масштабироваться при увеличении страницы.

Функция min()

Функция позволяет установить **наименьшее** значение из списка в качестве значения свойства.

Например,

```
div {
   width: min(50vw, 200px);
}
```

В приведенном примере ширина блока будет **не более 200 пикселей**, но будет меньше, если ширина области просмотра меньше 400 пикселей (в этом случае 1vw будет 4 пикселя, поэтому 50vw будет 200 пикселей).

При таком подходе абсолютная единица измерения используется, чтобы указать фиксированное максимальное

значение свойства, а относительная единица, чтобы уменьшать значение для соответствия меньшим окнам просмотра.

Вы можете указать более двух значений, если это необходимо. Разрешается использовать результаты сложных вычислений в качестве значений и устанавливать порядок действий с помощью круглых скобок. Также допустимо использовать разные единицы измерения для каждого значения в выражении.

Функция max()

В противовес предыдущей, эта функция позволяет установить **наибольшее** значение из списка в качестве значения свойства.

```
div{
    width: max(20vw, 400px);
}
```

В примере ширина будет не менее 400 пикселей, но будет больше, если ширина области просмотра превышает 2000 пикселей (в этом случае 1vw будет 20 пикселей, а 20vw будет 400 пикселей).

Абсолютная единица измерения используется для указания фиксированного минимального значения свойства и относительная единица, позволяющая увеличивать значение для соответствия большим окнам просмотра.

В целом, работа с функцией аналогична **min()**.

Функция clamp()

Функция фиксирует среднее в диапазоне значений между определенной минимальной и максимальной границей. Функция принимает три параметра: минимальное значение, предпочтительное значение и максимально допустимое значение.

min

Минимальное значение — нижняя граница диапазона допустимых значений. Если предпочтительное значение меньше этого значения, будет использоваться минимальное значение.

val

Предпочтительным значением является выражение, значение которого будет использоваться, пока результат находится между минимальным и максимальным значениями.

max

Наибольшее значение выражения, которое будет присвоено значению свойства, если предпочтительное значение превышает эту верхнюю границу.

При этом, любое из значений может быть математической функцией, вложенной **min()** или **max()**, вычисляемым выражением.

Интересно что для математических выражений сложение, вычитание, умножение и деление можно использовать напрямую, без функции **calc()**. При необходимости вы также можете использовать круглые скобки для установления порядка вычислений.

Например, настраиваем размер текста заголовка с помощью font-size.

```
h1 {
    font-size: clamp(1.8rem, 2.5vw, 2.8rem);
  }
```

Значение, относящееся к области просмотра, 2.5vw но оно должно находиться в диапазоне между 1.8rem и 2.8rem, поэтому:

- если расчетное значение 2.5vw меньше 1.8rem, то будет использоваться 1.8rem.
- если расчетное значение 2.5vw больше 2.8rem, то будет использоваться 2.8rem.

Это позволяет избежать слишком маленького текста заголовка в очень узком окне или слишком большого в очень широком.

ЗАДАНИЕ № 34 ПЕРЕМЕННЫЕ CSS.

В папке sample представлен макет сайта.

При его оформлении использовался определенный набор цветов: голубой, кастомный белый, темно-серый.

Также набор размеров текста: от 1em до 4em.

Единые настройки скругления кнопок и карточек.

Всё это позволяет сделать наш проект цельным.

Вашей задачей станет проанализировать стили и перенести часть настроек в переменные. Храните их в разделе :root.

Возможно следует изменить некоторые настройки, чтобы сократить количество используемых вариантов значений.

Поэкспериментируйте с цветами и размерами элементов, редактируя переменные. Оцените удобство их использования.