Лабораторная работа №3.

Тема: Стилизация веб-страницы с использованием CSS.

Цель работы: изучить принципы применения CSS для стилизации различных элементов веб-страницы. Освоить использование блочной модели, селекторов и каскадности стилей.

Основные термины и понятия:

CSS (Cascading Style Sheets) – это язык, используемый для описания вида HTML-документов. С его помощью можно задавать цвета, шрифты, другие визуальные параметры и эффекты, а также определять расположение HTML-элементов на странице.

CSS позволяет отделить *структуру* документа (HTML) от его *представления* (стилей). Это упрощает поддержку и разработку веб-страниц, а также улучшает их производительность.

Кроме того, CSS позволяет создавать адаптивные веб-страницы, которые автоматически подстраиваются под различные устройства и экраны.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на него никакого влияния.

Объявление стиля состоит из двух частей: **селектора** и **объявления**. В HTML имена элементов нечувствительны к регистру, поэтому «h1» работает так же, как и «H1». Объявление состоит из двух частей: имя свойства (например, color) и значение свойства (grey). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (в фигурных скобках) перечисляются форматирующие команды – свойства и их значения.

Существует несколько способов **подключения** CSS к HTML-документу.

– **Внешняя таблица** стилей представляет собой текстовый файл с расширением .css, в котором находится набор CSS-стилей. Внутри файла могут содержатся только стили, без HTML-разметки. Внешняя таблица стилей подключается к веб-странице с помощью тега <link>, расположенного внутри раздела <head></head>.

К веб-странице можно подключить несколько таблиц стилей, добавляя последовательно несколько элементов <link>. В необязательном атрибуте media можно указать условия подключения таблицы стилей. rel="stylesheet" указывает тип ссылки (ссылка на таблицу стилей).

Этот метод наиболее предпочтителен, так как позволяет централизованно управлять стилями. Внешние стили делают код более организованным и легко поддерживаемым, так как все стили находятся в одном месте.

```
<head>
   <link rel="stylesheet" href="css/style.css">
   <link rel="stylesheet" href="css/assets.css" media="screen and (max-width: 600px)">
</head>
```

Атрибут type не является обязательным по стандарту HTML5, поэтому его можно не указывать. Если атрибут отсутствует, по умолчанию используется значение type="text/css".

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Пример внешнего стиля</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <p>Пример текста с внешними стилями</p>
</body>
</html>
```

Файл styles.css:

```
p {
    color: blue;
    font-size: 20px;
}
```

– **Внутренние стили** встраиваются в раздел <head></head> HTML-документа и определяются внутри элемента <style></style>. Внутренние стили имеют приоритет над внешними, но уступают **встроенным стилям**. Этот метод удобен для страниц с уникальными стилями. Однако, если у вас много страниц с одинаковыми стилями, использование внутренних стилей может привести к дублированию кода и усложнению его поддержки.

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>Пример внутреннего стиля</title>
    <style>
        p {
            color: blue;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <p>Пример текста с внутренними стилями</p>
</body>
</html>
```

– **Встроенные стили** указываются непосредственно внутри HTML-элемента с помощью атрибута style. Этот метод подходит для небольших изменений, но не рекомендуется для крупных проектов. Встроенные стили могут сделать код менее

читаемым и сложным для поддержки, особенно если на странице много элементов с индивидуальными стилями.

```
<p style="color: blue; font-size: 20px;">Пример текста с встроенными стилями</p>
```

Такие стили действуют только на тот элемент, для которого они заданы.

Правило @import позволяет загружать внешние таблицы стилей. Чтобы директива @import работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами:

```
<style>
@import url(mobile.css);
p {
font-size: 0.9em;
color: grey;
}
</style>
```

Правило @import также используется для подключения веб-шрифтов.

Селекторы представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторами могут быть элементы, их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

Виды селекторов:

– Универсальный селектор. Соответствует любому HTML-элементу. Например, * {margin: 0;} обнулит внешние отступы для всех элементов сайта. Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом: *:after {CSS-стили}, *:checked {CSS-стили}.

– Селектор элемента. Позволяют форматировать все элементы данного типа на всех страницах сайта. Например, h1 {font-family: Lobster, cursive;} задаст общий стиль форматирования всех заголовков h1.

– Селектор класса. Позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта. Например, для создания заголовка с классом headline необходимо добавить атрибут class со значением headline в открывающий тег <h1> и задать стиль для указанного класса. Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.

.html

```
<h1 class="headline">Инструкция пользования персональным компьютером</h1>
```

.css

```
.headline {
    text-transform: uppercase;
    color: lightblue;
}
```

Если элемент имеет несколько классов, они записываются через пробел.

```
<h1 class="headline post-title">Инструкция пользования персональным компьютером</h1>
```

– **Селектор идентификатора.** Позволяет форматировать *один* конкретный элемент. Значение id должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ. Значение не должно содержать пробелов.

Нет никаких других ограничений на то, какую форму может принимать id, в частности, идентификаторы могут состоять только из цифр, начинаться с цифры, начинаться с подчеркивания, состоять только из знаков препинания и т. д.

Уникальный идентификатор элемента может использоваться для различных целей, в частности, как способ ссылки на конкретные части документа с использованием идентификаторов фрагментов, как способ нацеливания на элемент при создании сценариев и как способ стилизации конкретного элемента из CSS.

.html

```
<div id="sidebar"></div>
```

.css

```
#sidebar {
width: 300px;
float: left;
}
```

– **Селектор потомка** применяет стили к элементам, расположенным внутри элемента-контейнера. Например, ul li {text-transform: uppercase;} – выберет все элементы li, являющиеся потомками всех элементов ul.

Если нужно применить стили к потомкам определенного элемента, этому элементу можно задать класс:

- p.first a {color: green;} – применится ко всем ссылкам, потомкам абзаца с классом first;
- p .first a {color: green;} – применится к ссылкам, расположенным внутри любого элемента класса .first, который является потомком элемента <p>;
- .first a {color: green;} – применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом .first.

– **Дочерний селектор**. Дочерний элемент является прямым потомком (находится на первом уровне вложенности относительно родительского элемента). У элемента может быть несколько дочерних элементов, а родительский элемент только один. Дочерний селектор позволяет применить стили к дочернему элементу.

Например, p > strong – выберет все элементы strong, являющиеся дочерними по отношению к элементу p.

– **Сестринский селектор.** Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня:

- h1 + p – выберет все первые абзацы, идущие непосредственно за любым элементом <h1>, не затрагивая остальные абзацы;
- h1 ~ p – выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него.

– **Селекторы атрибутов** выбирают элементы на основе имени атрибута или значения атрибута:

- [attr] – все элементы, с именем атрибута attr, например, [alt] – все элементы, для которых задан атрибут alt;
- селектор[attr] – элементы, на которые действует селектор, и которые содержат атрибут с именем attr, например, img[alt] – только картинки, для которых задан атрибут alt;

• селектор[attr="value"] – аналогично предыдущему, но атрибут attr должен содержать точное значение value, например, img[title="flower"] – все картинки, название которых flower;
• селектор[attr~="value"] – элементы, на которые действует селектор, и которые содержат атрибут с именем attr, значение которого содержит слово value (value может быть отделено пробелами), например, p[class~="feature"] – абзацы, у которых есть класс с именем feature;
• селектор[attr|="value"] – элементы, на которые действует селектор, и которые содержат атрибут с именем attr, значение которого value или начинается на value- (после value следует дефис), напимер, p[class|="feature"] – абзацы, имя класса которых feature или начинается на feature (как правило, имена классов пишут в kebab-case);
• селектор[attr^="value"] – элементы, на которые действует селектор, и которые содержат атрибут с именем attr, значение которого начинается на value, например, a[href^="http://"] – все ссылки, начинающиеся на http://;
• селектор[attr$="value"] – аналогично предыдущему, но значение атрибута attr должно заканчиваться на value, например, img[src$=".png"] – все картинки с расширением .png;
• селектор[attr*="value"] – аналогично двум предыдущим, но значение атрибута attr должно содержать подстроку value в любом месте, например, a[href*="book"] – все ссылки, название которых содержит book.

– **Селектор псевдокласса.** Псевдоклассы — это классы, фактически не прикрепленные к HTML-элементам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.

Псевдоклассы характеризуют элементы со следующими свойствами:
• :link – не посещенная ссылка;
• :visited – посещенная ссылка;
• :hover – любой элемент, по которому проводят курсором мыши;
• :focus – интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;
• :active – элемент, который был активизирован пользователем, например, когда пользователь держит нажатую кнопку мыши на элементе <button>;
• :valid – поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;
• :invalid – поля формы, содержимое которых не соответствует указанному типу данных;
• :enabled – все активные поля форм;
• :disabled – заблокированные поля форм, т.е., находящиеся в неактивном состоянии;
• :in-range – поля формы, значения которых находятся в заданном диапазоне;
• :out-of-range – поля формы, значения которых не входят в установленный диапазон;
• :lang() – элементы с текстом на указанном языке;
• :not(селектор) – элементы, которые не содержат указанный селектор – класс, идентификатор, название или тип поля формы – :not([type="submit"]);
• :target – элемент, на который указывает текущий URL (когда в URL после знака # находится id какого-либо элемента на странице);
• :checked – выделенные (выбранные пользователем) элементы формы типа radio (<input type="radio">), checkbox (<input type="checkbox">) или option (<option> внутри <select>).

– **Селектор структурных псевдоклассов.** Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

• :nth-child(odd) – нечётные дочерние элементы;
• :nth-child(even) – чётные дочерние элементы;
• :nth-child(3n) – каждый третий элемент среди дочерних;
• :nth-child(3n+2) – каждый третий элемент, начиная со второго дочернего;
• :nth-child(n+2) – выбирает все элементы, начиная со второго;
• :nth-child(3) – третий дочерний элемент;
• :nth-last-child() – в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с :nth-child(), но начиная с последнего, в обратную сторону;
• :first-child – первый дочерний элемент;
• :last-child – последний дочерний элемент;
• :only-child – элемент, являющийся единственным дочерним элементом;
• :empty – элементы, у которых нет дочерних элементов;
• :root – элемент, являющийся корневым в документе – элемент html.

– **Селектор структурных псевдоклассов типа**. Указывают на конкретный тип элемента:
• :nth-of-type() – выбирает элементы по аналогии с :nth-child(), при этом берёт во внимание только тип элемента;
• :first-of-type – выбирает первый элемент данного типа;
• :last-of-type – выбирает последний элемент данного типа;
• :nth-last-of-type() – выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца;
• :only-of-type – выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

– **Селектор псевдоэлемента.** Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства content:
• :first-letter – выбирает первую букву каждого абзаца, применяется только к блочным элементам;
• :first-line – выбирает первую строку текста элемента, применяется только к блочным элементам;
• :before – вставляет генерируемое содержимое перед элементом;
• :after – добавляет генерируемое содержимое после элемента.

Для более точного отбора элементов для форматирования можно использовать **комбинации селекторов**:
• a[href][title] – выберет все ссылки, для которых заданы атрибуты href и title;
• img[alt*="css"]:nth-of-type(even) – выберет все четные картинки, альтернативный текст которых содержит слово css.

Группировка селекторов

Один и тот же стиль можно одновременно применить к нескольким элементам. Для этого необходимо в левой части объявления перечислить через запятую нужные селекторы:

```
h1,
h2,
p,
span {
color: tomato;
background: white;
}
```

Наследование и каскад – два фундаментальных понятия в CSS, которые тесно связаны между собой.

Наследование заключается в том, что элементы наследуют свойства от своего родителя (элемента, их содержащего).

Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.

Наследование является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам. Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как color, font, letter-spacing, line-height, list-style, text-align, text-indent, text-transform, visibility, white-space и word-spacing. Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.

Свойства, относящиеся к форматированию блоков, не наследуются. Это background, border, display, float и clear, height и width, margin, min-max-height и -width, outline, overflow, padding, position, text-decoration, vertical-align и z-index.

Принудительное наследование: с помощью ключевого слова inherit можно принудить элемент наследовать любое значение свойства родительского элемента. Это работает даже для тех свойств, которые не наследуются по умолчанию.

Как задаются и работают CSS-стили

Стили могут наследоваться от родительского элемента (наследуемые свойства или с помощью значения inherit).

Стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше.

К одному элементу могут применяться стили из разных источников. Проверить, какие стили применяются, можно в режиме разработчика браузера. Для этого над элементом нужно щёлкнуть правой кнопкой мыши и выбрать пункт «Посмотреть код» (или что-то аналогичное). В правом столбце будут перечислены все свойства, которые заданы для этого элемента или наследуются от родительского элемента, а также файлы стилей, в которых они указаны, и порядковый номер строки кода.

При определении стиля можно использовать любую комбинацию селекторов – селектор элемента, псевдокласса элемента, класса или идентификатора элемента.

Каскадирование – это механизм, который управляет конечным результатом в ситуации, когда к одному элементу применяются разные CSS-правила.

Существует три критерия, которые определяют порядок применения свойств – правило !important, специфичность и порядок, в котором подключены таблицы стилей.

Правило !important

Вес правила можно задать с помощью ключевого слова !important, которое добавляется сразу после значения свойства, например, span {font-weight: bold!important;}. Правило необходимо размещать в конец объявления перед закрывающей скобкой, без пробела. Такое объявление будет иметь приоритет над всеми остальными правилами. Это правило позволяет отменить значение свойства и установить новое для элемента из группы элементов в случае, когда нет прямого доступа к файлу со стилями.

Для каждого правила браузер вычисляет **специфичность селектора**, и если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило,

имеющее наибольшую специфичность. Значение специфичности состоит из четырех частей: 0, 0, 0, 0. Специфичность селектора определяется следующим образом:

- для id добавляется 0, 1, 0, 0;
- для class добавляется 0, 0, 1, 0;
- для каждого элемента и псевдоэлемента добавляется 0, 0, 0, 1;
- для встроенного стиля, добавленного непосредственно к элементу – 1, 0, 0, 0;
- универсальный селектор не имеет специфичности.

```
h1 {color: lightblue;} /*специфичность 0, 0, 0, 1*/
em {color: silver;} /*специфичность 0, 0, 0, 1*/
h1 em {color: gold;} /*специфичность: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2*/
div#main p.about {color: blue;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 + 0, 0, 0, 1 +
0, 0, 1, 0 = 0, 1, 1, 2*/
.sidebar {color: grey;} /*специфичность 0, 0, 1, 0*/
#sidebar {color: orange;} /*специфичность 0, 1, 0, 0*/
li#sidebar  {color: aqua;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 = 0, 1, 0, 1*/
```

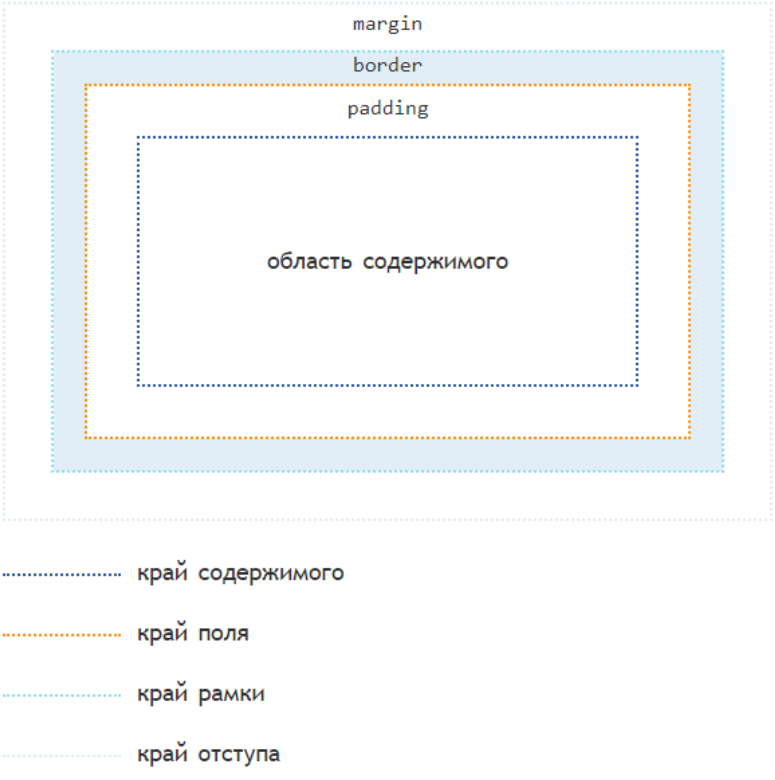
В результате к элементу применятся те правила, специфичность которых больше. Например, если на элемент действуют две специфичности со значениями 0, 0, 0, 2 и 0, 1, 0, 1, то выиграет второе правило.

Порядок подключённых таблиц

Вы можете создать несколько внешних таблиц стилей и подключить их к одной веб-странице. Если в разных таблицах будут встречаться разные значения свойств одного элемента, то в результате к элементу применится правило, находящееся в таблице стилей, идущей в списке ниже.

Блочная модель

Каждый блок имеет *область содержимого*, в которой находится текст, дочерние элементы, изображение и т.п., и необязательные окружающие ее padding, border и margin. Размер каждой области определяется соответствующими свойствами и может быть нулевым, или, в случае margin, отрицательным.



Поля, рамка и отступы могут быть разбиты на верхний, правый, нижний и левый сегменты, каждый из которых независимо управляется своим соответствующим свойством.

Фон области содержимого, полей и рамки блока определяется свойствами фона. Область рамки может быть дополнительно окрашена с помощью свойства border. Отступы элемента всегда прозрачны, что позволяет показывать фон родительского элемента.

Так как поля и отступы элемента не являются обязательными, по умолчанию их значение равно нулю. Тем не менее, некоторые браузеры добавляют этим свойствам положительные значения по умолчанию на основе своих таблиц стилей. Очистить стили браузеров для всех элементов можно при помощи универсального селектора:

```
* {
    margin: 0;
    padding: 0;
 }
```

Выделяют две основные категории HTML-элементов, которые соответствуют типам их содержимого и поведению в структуре веб-страницы – **блочные** и **строчные элементы**. С помощью блочных элементов можно создавать структуру веб-страницы, строчные элементы используются для форматирования текстовых фрагментов (за исключением элементов <area> и <img>).

Блочные элементы – элементы высшего уровня, которые форматируются визуально как блоки, располагаясь на странице в окне браузера вертикально. Значения свойства display, такие как block, list-item и table делают элементы блочными. Блочные элементы генерируют основной блок, который содержит только блок элемента. Элементы со значением display: list-item генерируют дополнительные блоки для маркеров, которые позиционируются относительно основного блока.

<address>, <article>, <aside>,
<blockquote>,
<dd>, , <dl>, <dt>, <details>,
<fieldset>, <figcaption>, <figure>, <footer>, <form>,
<h1>-<h6>, <header>, <hr>,
, <legend>,
<nav>, <noscript>,
, <output>, <optgroup>, <option>,
<p>, <pre>,
<section>, <summary>,
<table>,

Блочные элементы могут размещаться непосредственно внутри элемента <body>. Они создают разрыв строки перед элементом и после него, образуя прямоугольную область, по ширине занимающую всю ширину веб-страницы или блока-родителя.

Блочные элементы могут содержать как строчные, так и блочные элементы, но не оба типа элементов сразу. При необходимости, строки текста, принадлежащие блочному контейнеру, могут быть обёрнуты анонимными контейнерами, которые будут вести себя внутри блока как элементы со значением display: block;, а строчные элементы обёрнуты элементом <p>. Блочные элементы могут содержаться только в пределах блочных элементов.

Элемент <p> относится к блочным элементам, но он не должен содержать внутри себя другой элемент <p>, а также любой другой блочный элемент.

Встроенные (строчные) элементы генерируют внутристрочные контейнеры. Они не формируют новые блоки контента. Значения свойства display, такие как inline и inline-table делают элементы строчными.

<a>, <area>,
<b>, <bdo>, <bdi>,
<cite>, <code>,
<dfn>, <del>,
,
<i>, <iframe>, <img>, <ins>,
<kbd>,
<label>,
<map>, <mark>,
<s>, <samp>, <small>, , , <sub>, <sup>,
<time>,
<q>,
<ruby>,
<u>,
<var>

Строчные элементы могут содержать только данные и другие строчные элементы. Исключение составляет элемент <a>, который согласно спецификации HTML5 может оборачивать целые абзацы, списки, таблицы, заголовки и целые разделы при условии, что они не содержат другие интерактивные элементы – другие ссылки и кнопки.

CSS предоставляет множество свойств для стилизации элементов. Рассмотрим основные из них:

Стили для текста

– color: задает цвет текста.
– font-size: задает размер шрифта.
– font-family: задает семейство шрифтов.
– text-align: задает выравнивание текста.

```
p {
    color: blue;
    font-size: 20px;
    font-family: Arial, sans-serif;
    text-align: center;
}
```

CSS-свойства цвета и фона

– color: цвет переднего плана элемента.
– background-color: задает цвет фона.
– background-image: используется для вставки фонового изображения.
– background-repeat: задает повторяющийся фон.
– background-attachment: определяет, фиксируется ли фоновый рисунок, или прокручивается вместе с содержимым страницы.

– background-position: по умолчанию фоновый рисунок позиционируется в левом верхнем углу экрана. Свойство background-position позволяет изменять это значение, и фоновый рисунок может располагаться в любом месте экрана.

– background: свойство background входит в состав всех свойств, перечисленных в этой части. С помощью background вы можете сжимать несколько свойств и записывать стили в сокращённом виде, что облегчает чтение таблиц.

```
body {
    background-color: lightgray;
    background-image: url('background.jpg');
    background-repeat: no-repeat;
}
```

CSS-свойства шрифта

– font-family: указывает приоритетный список шрифтов, используемых для отображения данного элемента или web-страницы.

– font-style: стиль начертания для шрифта (normal, italic, oblique, …).

– font-weight: задаёт насыщенность шрифта (normal, bold)

– font-size: размер шрифта (используются различные единицы измерения, например, пикселы и проценты)

– font: сокращенная запись свойств шрифта

CSS-свойства при отображении текста

– text-indent: позволяет выделить параграф с помощью установки отступа для его первой строки.

– text-align: выравнивание текста

– text-decoration: позволяет добавлять различные декоративные эффекты. Например, можно подчеркнуть текст, провести линию по или над текстом и т. д.

– letter-spacing: задает интервал между буквами

– text-transform: управляет регистром символов (capitalize, uppercase или lowercase)

Стили для блочных элементов

– width и height: задают ширину и высоту элементов.

– margin: задает внешние отступы.

– padding: задает внутренние отступы.

– border: задает границу элемента.

```
div {
    width: 200px;
    height: 100px;
    margin: 20px;
    padding: 10px;
    border: 2px solid black;
}
```

Всё изученное можно применять и для **ссылок/links** (например, изменять шрифт, цвет, подчёркивание и т. д.). Новым будет то, что в CSS эти свойства можно определять по-разному, в зависимости от того, посетили уже ссылку, активна ли она, находится ли указатель мыши над ссылкой. Это позволяет добавить интересные эффекты. Для этого используются так называемые **псевдоклассы**, которые позволяет учитывать различные условия или события при определении свойств HTML-тега.

– псевдокласс :**link** используется для ссылок на страницы, которые пользователь ещё не посещал;

– псевдокласс :**visited** используется для ссылок на страницы, которые пользователь посетил;

– псевдокласс :**active** используется для активных ссылок;

– псевдокласс :**hover** используется для ссылок, над которыми находится указатель мыши. Например, чтобы ссылки становились оранжевыми и имели курсивное начертание при прохождении указателя над ними:

```
a:hover {
color: orange;
font-style: italic;
}
```

Спецификация CSS даёт неограниченные возможности для **оформления таблиц**. По умолчанию таблица и ячейки таблицы не имеют видимых границ и фона, при этом ячейки внутри таблицы не прилегают вплотную друг к другу.

Ширина ячеек таблицы определяется шириной их содержимого, поэтому ширина столбцов таблицы может быть разной. Высота всех ячеек ряда одинаковая и определяется высотой самой высокой ячейки.

CSS-рамка элемента представляет собой одну или несколько линий, окружающих содержимое элемента и его поля padding. Рамка задаётся с помощью свойства border. Стиль рамки задается с помощью трех свойств: **стиль**, **цвет** и **ширина**.

Таблица и ячейки внутри неё по умолчанию отображаются в браузере без видимых границ. **Границы таблицы** задаются свойством border:

```
table {
   border-collapse: collapse;
   /*убираем пустые промежутки между ячейками*/
   border: 1px solid grey;
   /*устанавливаем для таблицы внешнюю границу серого цвета толщиной 1px*/
}
```

По умолчанию фон таблицы и ячеек прозрачный. Если страница или блок, содержащие таблицу, имеют фон, то он будет просвечиваться сквозь таблицу. Если фон задан и для таблицы, и для ячеек, то в местах наложения фона таблицы и ячеек будет виден фон только ячеек. В качестве фона для таблицы в целом и её ячеек могут выступать: заливка сплошным цветом, градиентная заливка, фоновое изображение.

CSS-градиент представляет собой переходы от одного цвета к другому.

Градиенты создаются с помощью функций linear-gradient() – линейный градиент и radial-gradient() – радиальный.

Градиентный фон можно устанавливать в свойствах background, background-image, border-image и list-style-image.

Линейный градиент создается с помощью двух и более цветов, для которых задано направление, или линия градиента. Если направление не указано, используется значение по умолчанию – сверху-вниз. Цвета градиента по умолчанию распределяются равномерно в направлении, перпендикулярном линии градиента.

```
background: linear-gradient(угол / сторона или угол наклона с помощью ключевого слова
(пары ключевых слов), первый цвет, второй цвет и т.д.);
```

Радиальный градиент отличается от линейного тем, что цвета выходят из одной точки (центра градиента) и равномерно распределяются наружу, рисуя форму круга или эллипса.

CSS переменные (пользовательские CSS-свойства) – это сущности, определяемые разработчиком, хранящие конкретные значения, которые можно повторно использовать в документе. Они устанавливаются с использованием custom property нотации (например, --main-color: black;) и доступны через функцию var() (например, color: var(--main-color);).

Сложные веб-сайты имеют очень большое количество CSS, часто с множеством повторяющихся значений. Например, один и тот же цвет может использоваться в сотнях разных мест, что требует глобального поиска и замены, если этот цвет необходимо изменить. CSS переменные позволяют сохранять значение в одном месте, а затем многократно использовать его в любом другом месте. Дополнительным преимуществом являются семантические идентификаторы. Для примера: запись --main-text-color более понятна, чем #00ff00, особенно если этот же цвет используется и в другом контексте.

CSS переменные подчиняются каскаду и наследуют значения от своих родителей.

Задание:

Разработайте CSS-стили для созданной HTML-страницы:

– Настройте оформление текста (шрифты, размеры, стили заголовков и ссылок).

– Реализуйте оформление таблиц (рамки, отступы, заливка).

– Создайте градиентный фон для одного из разделов страницы.

– Настройте внешний вид списков, используя нестандартные маркеры.

– Реализуйте эффекты наведения для ссылок и кнопок.

Дополнительно: примените CSS-переменные для создания единой цветовой схемы.

Ход выполнения лабораторной работы должен быть отражен в *отчете*. Отчет должен содержать титульный лист, цель работы, задание, листинг исходного кода, описание проделанной работы, скриншоты результата, вывод, ответы на контрольные вопросы.

Контрольные вопросы:

1. Что такое CSS? Для чего используется?
2. Какие существуют способы подключения CSS к HTML-документу?
3. Как объявить стиль?
4. Что такое селекторы?
5. Перечислите виды селекторов?
6. Можно ли комбинировать селекторы?
7. Как работает группировка селекторов?
8. Связь наследования и каскада.
9. Что такое наследование? Какие свойства наследуются и не наследуются?
10. Как работает механизм принудительного наследования?
11. Что такое каскадирование?
12. Перечислите критерии, которые определяют порядок применения свойств.
13. Определение блочной модели.
14. Какие элементы являются блочными?
15. Какие элементы являются строчными?

16. Как определить CSS переменную?
17. Перечислите CSS-свойства фона и цвета.
18. Перечислите CSS-свойства шрифта.
19. Какие псевдоклассы используются для ссылок?
20. Какие виды CSS-градиентов вы знаете?