2.1. Основы CSS

CSS (Cascading Style Sheets) — язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML).

Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

Отделяя стиль представления документов от содержимого документов, CSS упрощает создание веб-страниц и обслуживание сайтов.

CSS поддерживает таблицы стилей для конкретных носителей, поэтому авторы могут адаптировать представление своих документов к визуальным браузерам, слуховым устройствам, принтерам, брайлевским устройствам, карманным устройствам и т.д.

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 (код в фигурных скобках) перечисляются форматирующие команды — свойства и их значения.

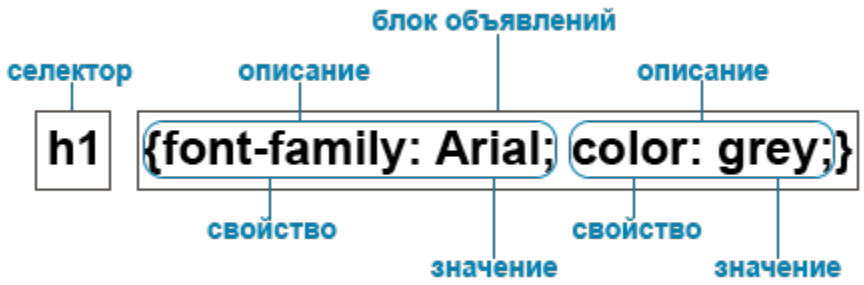


РИС. 1. СТРУКТУРА ОБЪЯВЛЕНИЯ

Хотя приведенный пример пытается влиять только на пару свойств, необходимых для рендеринга HTML-документа, он сам по себе квалифицируется как таблица стилей. В сочетании с другими таблицами стилей (одна фундаментальная особенность CSS заключается в том, что таблицы стилей объединяются), правило будет определять окончательное представление документа.

1. Виды таблиц стилей

1.1. Внешняя таблица стилей

Внешняя таблица стилей представляет собой текстовый файл с расширением `.css`, в котором находится набор CSS-стилей элементов. Файл создаётся в редакторе кода, так же как и HTML-страница. Внутри файла могут содержатся только стили, без HTML-разметки. Внешняя таблица стилей подключается к веб-странице с помощью элемента `<link>`, расположенного внутри раздела `<head></head>`. Такие стили работают для всех страниц сайта.

К каждой веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько элементов `<link>`, указав в атрибуте `media` назначение данной таблицы стилей. `rel="stylesheet"` указывает тип ссылки (ссылка на таблицу стилей).

HTML

```
<head>
<link rel="stylesheet" href="css/style.css">
<link rel="stylesheet" href="css/assets.css" media="all">
</head>
```

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

1.2. Внутренние стили

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

(заданным через атрибут `style`).

```HTML
<head>
<style>
h1,
h2 {
color: red;
font-family: "Times New Roman", Georgia, Serif;
line-height: 1.3em;
}
</style>
</head>
<body>
...
</body>
```

1.3. Встроенные стили

Когда мы пишем **встроенные стили**, мы пишем CSS-код в HTML-файл, непосредственно внутри элемента с помощью атрибута `style`:

```HTML
<p style="font-weight: bold; color: red;">Обратите внимание на этот текст.</p>
```

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

1.4. Правило @import

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

```HTML
<style>
@import url(mobile.css);
```

```
p {
font-size: 0.9em;
color: grey;
}
</style>
```

Правило `@import` также используется для подключения веб-шрифтов:

```CSS
@import url(https://fonts.googleapis.com/css?family=Open+Sans&subset=latin,cyrillic);
```

2. Виды селекторов

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.

2.1. 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}`.

2.2. Селектор элемента

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта. Например, `h1 {font-family: Lobster, cursive;}` задаст общий стиль форматирования всех заголовков `h1`.

2.3. Селектор класса

Селекторы 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 атрибутов класса, их значения объединяются с пробелами.

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

2.4. Селектор идентификатора

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;
}
```

2.5. Селектор потомка

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера. Например, `ul li {text-transform: uppercase;}` — выберет все элементы `li`, являющиеся потомками всех элементов `ul`.

Если нужно отформатировать потомки определенного элемента, этому элементу нужно задать стилевой класс:

- `p.first a {color: green;}` — данный стиль применится ко всем ссылкам, потомкам абзаца с классом `first`;
- `p .first a {color: green;}` — если добавить пробел, то будут стилизованы ссылки, расположенные внутри любого элемента класса `.first`, который является потомком элемента `<p>`;
- `.first a {color: green;}` — данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом `.first`.

2.6. Дочерний селектор

Дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один. Дочерний селектор позволяет применить стили только если дочерний элемент идёт сразу за родительским элементом и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен.

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

2.7. 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.

2.8. Селектор атрибута

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

- `[атрибут]` — все элементы, содержащие указанный атрибут, `[alt]` — все элементы, для которых задан атрибут `alt`;
- `селектор[атрибут]` — элементы данного типа, содержащие указанный атрибут, `img[alt]` — только картинки, для которых задан атрибут `alt`;
- `селектор[атрибут="значение"]` — элементы данного типа, содержащие указанный атрибут с конкретным значением, `img[title="flower"]` — все картинки, название которых содержит слово `flower`;
- `селектор[атрибут~="значение"]` — элементы частично содержащие данное значение, например, если для элемента задано несколько классов через пробел, `p[class~="feature"]` — абзацы, имя класса которых содержит `feature`;
- `селектор[атрибут|="значение"]` — элементы, список значений атрибута которых начинается с указанного слова, `p[class|="feature"]` — абзацы, имя класса которых `feature` или начинается на `feature`;
- `селектор[атрибут^="значение"]` — элементы, значение атрибута которых начинается с указанного значения, `a[href^="http://"]` — все ссылки, начинающиеся на `http://`;
- `селектор[атрибут$="значение"]` — элементы, значение атрибута которых заканчивается указанным значением, `img[src$=".png"]` — все картинки в формате `png`;
- `селектор[атрибут*="значение"]` — элементы, значение атрибута которых содержит в любом месте указанное слово, `a[href*="book"]` — все ссылки, название которых содержит `book`.

2.9. 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;
- `: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 с символом `#`, на который ссылаются в документе;
- `:checked` — выделенные (выбранные пользователем) элементы формы.

2.10. 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)` — выбирает каждый третий элемент, начиная со второго дочернего элемента `(+2)`;
- `:nth-child(n+2)` — выбирает все элементы, начиная со второго;
- `:nth-child(3)` — выбирает 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` — позволяет оформить только самый первый дочерний элемент;
- `:last-child` — позволяет форматировать последний дочерний элемент;
- `:only-child` — выбирает элемент, являющийся единственным дочерним элементом;
- `:empty` — выбирает элементы, у которых нет дочерних элементов;
- `:root` — выбирает элемент, являющийся корневым в документе — элемент `html`.

2.11. Селектор структурных псевдоклассов типа

Указывают на конкретный тип дочернего элемента:

- `:nth-of-type()` — выбирает элементы по аналогии с `:nth-child()`, при этом берёт во внимание только тип элемента;
- `:first-of-type` — выбирает первый дочерний 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.

2.12. 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.

3. Комбинация селекторов

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`.

4. 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:

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

5. Наследование и каскад

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.

5.1. Наследование

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

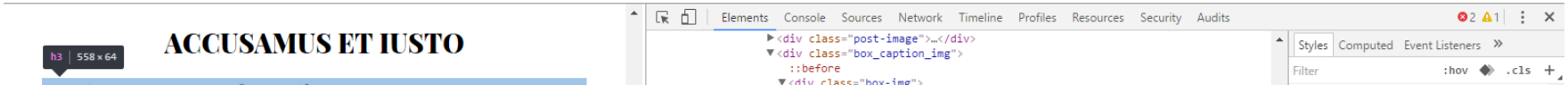
С помощью ключевого слова `inherit` можно принудить элемент наследовать любое значение свойства родительского элемента. Это работает даже для тех свойств, которые не наследуются по умолчанию.

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.



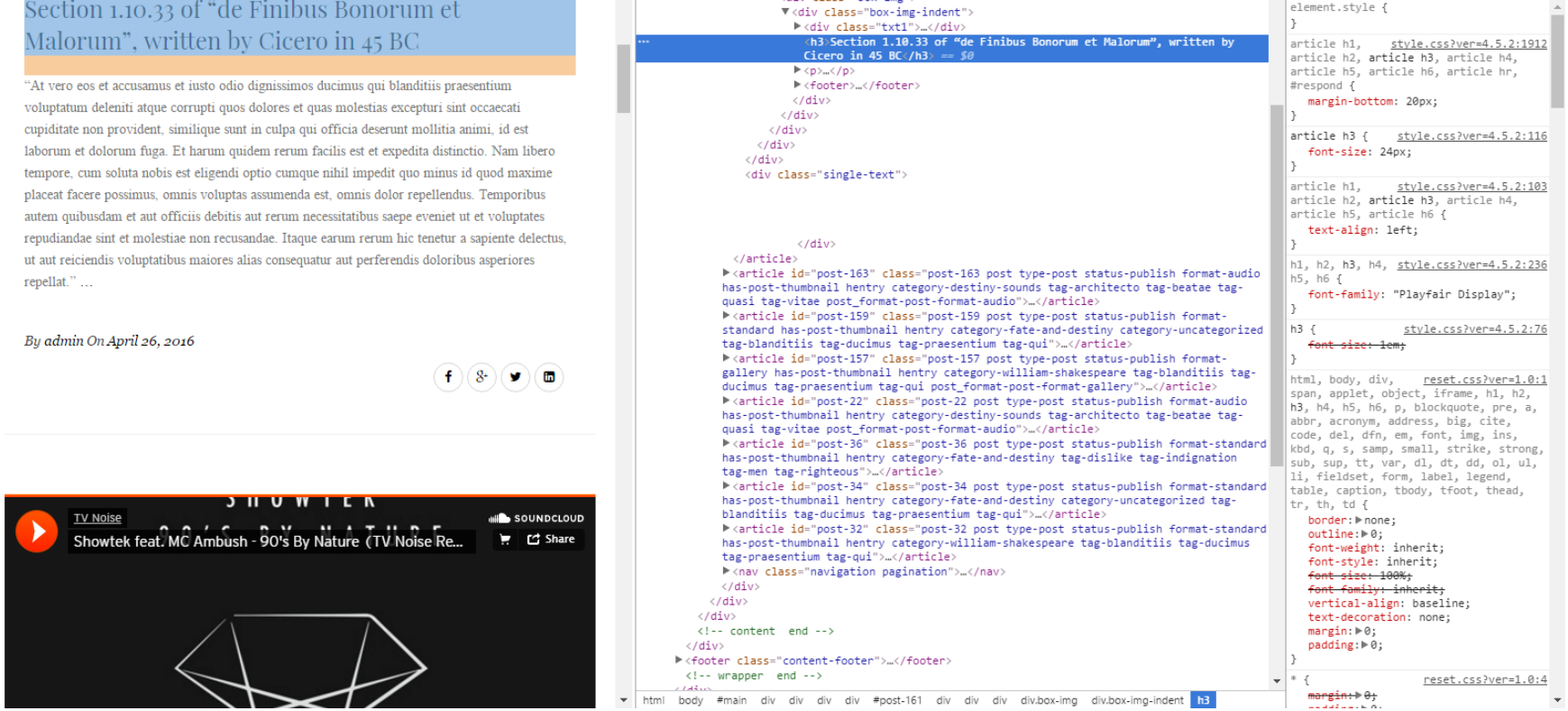


РИС. 2. РЕЖИМ РАЗРАБОТЧИКА В БРАУЗЕРЕ GOOGLE CHROME

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

HTML

```
<div id="wrap" class="box clear"></div>
```

CSS

```
div {border: 1px solid #eee;}
#wrap {width: 500px;}
.box {float: left;}
.clear {clear: both;}
```

5.2. Каскад

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.

Специфичность

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.

CSS

```
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
.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.