CSS

Что такое CSS

CSS (Cascading Style Sheets) - каскадные таблицы стилей.

Стиль - набор параметров, задающий внешнее представление объекта. Например, пусть мы хотим, чтобы все заголовки первого уровня (теги <h1>) на одной странице имели красный цвет, размер - 24 и были написаны курсивом, а на другой странице были бы синего цвета, размера - 12. Наш заголовок - это объект, а цвет, размер и начертание - это параметры. Просто параметры нашего объекта для разных страниц разные, т.е. они отличаются стилем.
Каждый элемент на странице может иметь свой стиль. Набор стилей всех элементов называют таблицей стилей.
Если для одного элемента задано несколько стилей (как в примере с заголовками), то применяется каскадирование, которое определяет приоритет того или иного стиля.

CSS, как и любой язык, имеет свой синтаксис. В нем нет ни элементов, ни параметров, ни тегов. В нем есть правила: CSS состоит из селектора и блока объявления стилей,

Сам блок объявления стилей состоит из свойств и их значений, разделенных точкой с запятой:

 пример селектора

p.title { font-family: Arial, Helvetica, sans-serif; font-size: 18px; color: maroon; font-weight: bold }

 пример селектора

p#title { font-family: Arial, Helvetica, sans-serif; font-size: 18px; color: maroon; font-weight: bold }

В примере используется селектор с именем (id) title для которого задано определение стиля. Определение состоит из свойства и его значения разделенных двоеточием. Определения разделяются точкой с запятой.

```
селектор {

        свойство: значение;

        свойство: значение;

        свойство: значение;

}
```

Селекторы

Классы (.myclass)

ID (#myid)

Элементы (div)

Стандарт CSS определяет приоритеты, в порядке которых применяются правила стилей, если для какого-то элемента подходят свойства нескольких правил одновременно (или, в редких случаях, в одном правиле есть одноименные свойства). Это называется «каскадом», в котором для свойств рассчитываются приоритеты или «веса», что делает результаты предсказуемыми.

Приоритеты рассчитываются от большего к меньшему:

1. свойство задано при помощи !important;

2. стиль прописан инлайново в теге в html через атрибут style="";

3. количество идентификаторов (#id) в селекторе;

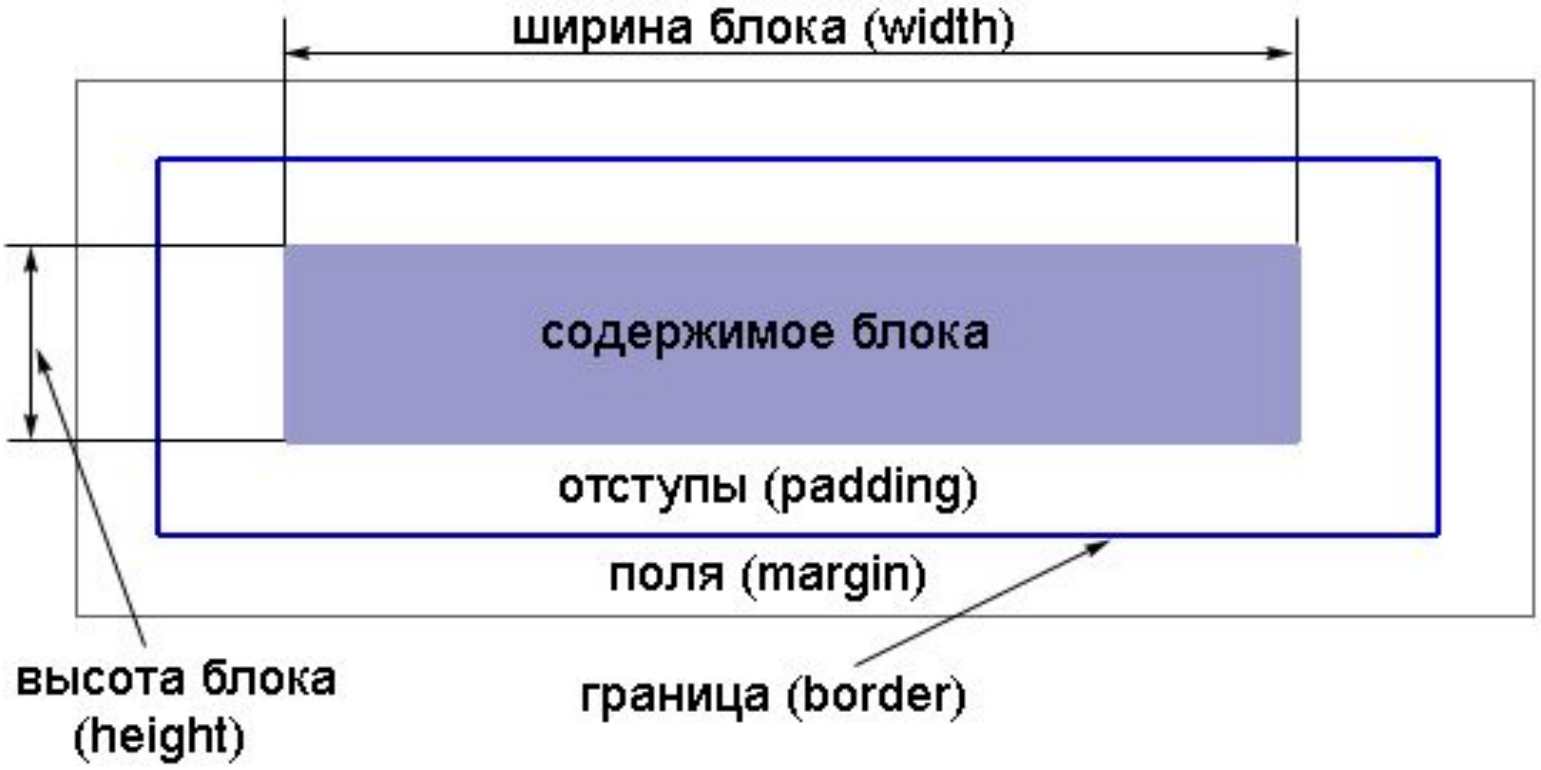
4. количество классов (.class) и псевдоклассов (:pseudoclass) в селекторе;

5. количество имён тегов в селекторе;

Кроме того, имеет значение **относительный порядок** расположения свойств – **свойство, указанное позже, имеет приоритет**.

Также следует помнить о **наследовании** — это перенос правил форматирования для элементов, находящихся внутри других.

Например, если для тега <p> задан красный цвет текста, а для курсива , который находится внутри абзаца, нет, то в этом случае вложенный элемент наследует свойства своего родителя и курсивный текст также будет красным.



У блока есть содержимое, например, для элемента p - это текст. Вокруг содержимого есть отступы (padding), они служат для того, чтобы текст не примыкал вплотную к границе блока. Фон отступов такой же, как и у содержимого.

Затем идет граница блока (border), которая может быть как видимой, так и невидимой. Также блок имеет поля (margin), которые задают дополнительное свободное пространство вокруг блока. Фон полей прозрачный, т.е. сквозь него просвечивает фон родительского элемента.

Размер блока, т.е. его ширина (width) и высота (height), определяются содержимым. И это надо запомнить: поля и отступы не учитываются в размере блока.

Элементы могут быть блочными и строчными По умолчанию для каждого элемента его вид определен, так элементы div и p являются блочными, а и <a> - строчными. Но иногда это необходимо изменить, для этого используется свойство display. Это свойство множество значений.

Border , Padding , Margin

```
<body>
    <p>Текст в параграфе.</p>
    <p>Текст в параграфе.</p>
</body>
```

Универсальное свойство border позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному свойству. Для установки границы только на определенных сторонах элемента, воспользуйтесь свойствами border-top, border-bottom, border-left,border-right.

Значение border-width определяет толщину границы. Для управления ее видом предоставляется несколько значений border-style.

Для того, чтобы увидеть отступы, поля и границы, зададим границу

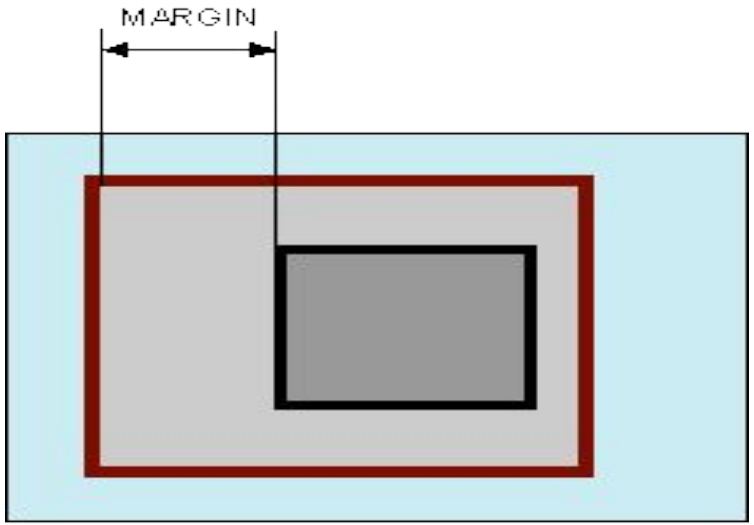
p{ border:1px solid red;}

Текст в параграфе.

Текст в параграфе.

Отступы от границы задаются свойством padding, оно позволяет задать величину внутреннего отступа сразу для всех сторон элемента или определить ее только для указанных сторон.

Добавляем к нашему свойству p{ padding:10px;}



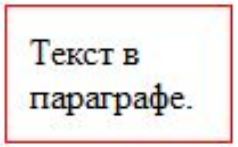
Далее задаем внешние отступы – Margin . Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента

Если у элемента нет родителя, отступом будет расстояние от края элемента до края окна браузера с учетом того, что у самого окна по умолчанию тоже установлены отступы. Чтобы от них избавиться, следует устанавливать значение margin для селектора <body> равное нулю.

Свойство margin позволяет задать величину (px или %) отступа сразу для всех сторон элемента или определить ее только для указанных сторон.

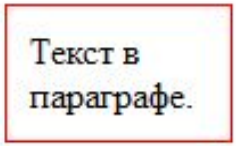
Auto Указывает, что размер отступов будет автоматически рассчитан браузером.

Inherit Наследует значение родителя.



И снова добавляем свойства к нашему параграфу p{ margin:50px;}

А теперь добавляем ширину и высоту нашим блокам



p{ width:100px; height:50px;}

box-sizing

За долгое время, люди осознали, что обременять себя изнурительными вычислениями не очень весело, поэтому было создано новое свойство CSS под названием box-sizing. Когда вы устанавливаете box-sizing: border-box; для элемента, поля и границы самого элемента больше не увеличат его ширину.

```
.simple {
        width: 500px;
        margin: 20px auto;
        -webkit-box-sizing: border-box;
        -moz-box-sizing: border-box;
        box-sizing: border-box; }
.fancy {
        width: 500px;
        margin: 20px auto;
        padding: 50px;
        border: solid blue 10px;
        -webkit-box-sizing: border-box;
        -moz-box-sizing: border-box;
        box-sizing: border-box; }
```

```
{ -webkit-box-sizing: border-box; -moz-box-sizing: border-box; box-sizing: border-box; }
```

Это гарантирует, что размеры всех элементов могут быть определены более интуитивно. Поскольку box-sizing достаточно новое свойство, вы должны использовать -webkit- и -moz- префиксы, как я в этих примерах. Эта техника позволяет реализовывать экспериментальные возможности в конкретных браузерах. Кроме того, имейте в виду, что один из них это IE8 и старше.

Многослойность

Представьте себе множество листков бумаги, на каждом из которых что-то написано или нарисовано. Мы видим только содержание верхнего листа, но если мы его снимем, то увидим содержание следующего листа и т.д.
Также и в CSS, мы можем создать несколько слоев, на каждом разместить необходимые элементы и пронумеровать слои с помощью свойства z-index. Чем больше номер, тем выше слой находится в стопке слоев. Например, если сделать 6 слоев, то пользователь сначала увидит именно слой z-index:6.

Чтобы было визуально понятно давайте изменим нашему блоку контент цвет фона

#menu {Position: relative; z-index:1;}

#content{ Left;0;background: green; z-index:2}

body{background: blue;}

Теперь, давайте сделаем текст на странице белым цветом:

body{background: blue;color: white;}

Теперь сделаем цвета заголовков красным (для h1) и желтым (для h2):

h1{color:red;}

h2{color:yellow;}

Стиль z-index может работать только в паре с position: relative\absolute\fixed

Группировка селекторов

Если стили для нескольких селекторов совпадают (например, заголовки первых трех уровней - зеленого цвета), то их можно сгруппировать. Для этого селекторы, нужно перечислить через запятую. Пример:

```
h1, h2, h3{  color:green;}
```

Если кроме цвета, мы хотим задать нашим заголовкам размер. Тогда мы можем просто дописать стили ниже:

```
h1, h2, h3{  color:green;}
h1{  font-size:18px;}
h2{  font-size:16px;}
h3{  font-size:14px;}
```

Color

в компьютерном дизайне чаще всего используют цветовую модель RGB. В ее основе лежат три цвета: red - красный, green - зеленый, blue - синий. С помощью смешивания этих трех цветов можно получить почти весь видимый спектр.
Для указания RGB-цвета в HTML используется шестнадцатеричная система счисления. Начинается код цвета со спецсимвола # (что и указывает на шестнадцатеричный код), а далее следует шесть символов. Первые два отвечают за количество красного оттенка, третий и четвертый за насыщенность зеленого, а пятый и шестой - синего. Так, #FF0000 - красный цвет, #00FF00 - зеленый, #0000FF - синий, а #FFFFFF - белый.

Значениями свойства color могут быть именные цвета (red, blue...), шестнадцатеричные коды цветов (#FF0000) и десятичные коды цвета в модели RGB (rgb(255, 0, 0)).
Итак, задать цвет текста для элемента можно тремя способами:

```
body{  color:green;}

h1{  color:#FF0000;}

h2{  color:rgb(255,0,0);}
```

Фон - background

На самом деле это группа свойств, так или иначе связанная с фоном. При помощи CSS, фон можно задать любому элементу. Пример:

```
<body>    Здесь содержимое документа    </body>
```

Рассмотрим группу свойств background:

background-color - задает цвет фона. По умолчанию не наследуется, но его можно сделать наследуемым, если в качестве значения указать значение inherit.

```
body{ background-color:#243CED; color: yellow; }
```

background-image - задает фоновое изображение. Значением свойства является URL графического файла. Формат задания следующий: сначала идет обозначение функции url, а затем в круглых скобках указывается путь к файлу. Путь к файлу указывается относительно таблицы стилей.

```
body{
        background-image:url(picture.gif);
        background-color:#243CED;
        color:yellow;
}
```

background-repeat - задает возможность повторения фонового изображения. В качестве фонового изображения может выступать как цельное изображение (например, шапка сайта), так и маленькое изображение, которое должно замостить собой все пространство элемента. Данное свойство как раз и указывает, повторять ли изображение и, если да, то как именно повторять. Возможны 4 варианта:

repeat - повторять изображение по горизонтали и вертикали.

repeat-x - повторять изображение только по горизонтали.

repeat-y - повторять изображение только по вертикали.

no-repeat - не повторять изображение.

По умолчанию используется значение repeat

```
body{
        background-image:url(picture.gif);
        background-repeat:no-repeat;
        background-color:#243CED;
        color:yellow;
}
```

background-position - задает расположение элемента относительно окна браузера. Значения можно задавать в процентах, в единицах длины и при помощи ключевых слов.

```
body{
        background-image:url(picture.gif);
        background-repeat:no-repeat;
        background-color:#243CED;
        background-position:10% 30%;
        color:yellow;
}
```

background-size свойство позволяет масштабировать фоновое изображение по вертикали и горизонтали (background-image). Оно описывает, как изображение будет растягиваться и обрезаться, чтобы полностью закрыть собой фоновую область. С помощью этого свойства изображение также можно уменьшать по ширине и по высоте. Не наследуется.

```
div {background-size: 300px 150px;}

div {background-size: 50% 30%;}

div {background-size: cover;}

div {background-size: contain;}
```

Значения:	
auto	Значение по умолчанию. Высота и ширина изображения равны его оригинальным размерам.
px / em / vh / vw	Размер задается парой значений, первое значение устанавливает ширину изображения, второе — высоту. Для того, чтобы фон масштабировался вместе с текстом, размеры изображения нужно задавать в em.
%	Задает размер фонового изображения в процентах от ширины или высоты элемента, которое заполняется фоном.
cover	Масштабирует изображение с сохранением пропорций так, чтобы его ширина или высота равнялась ширине или высоте блока.
contain	Масштабирует изображение с сохранением пропорций таким образом, чтобы оно целиком поместилось внутри блока.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Сокращенная запись свойства background в CSS для многих свойств существует сокращенная запись. В этом случае значения всех свойств перечисляются через пробел в произвольном порядке.

```
body{  background:url(picture.gif) no-repeat #33CCFF center top;}
```

Псевдоэлементы

CSS применяется к элементам HTML. Но есть несколько элементов, которых не существует в HTML, но они присутствуют на странице (первая буква слова и первая строка абзаца). Такие элементы и называют псевдоэлементами. Им можно задавать стиль, как и элементам HTML.
after - добавление контента ПОСЛЕ указанного элемента
before - добавление контента ДО указанного элемента
firstletter - стили для первой буквы в контенте элемента
firstline - стилевое оформление первой строки текста в элементе
selection - применение стилей при выделении текста в элементе
Одной из самых распространённых задач является добавление фразы до или после элемента , при помощи before и after

<body>

какой то текст

Как красиво, не правда ли? Пиши себе и пиши, первые буквы сами получатся большими !

А в этом самом абзаце первая строчка будет в 16 px; написана будет жирным шрифтом и красного цвета. А также можно выделять разными цветами первые строки и задавать отличительные стили при выделении текста

</body>

В css пишем :

.quote {display block; width:50%; margin: 50px auto 50px auto;}

p { display block; overflow:hidden; width:50%; margin: 50px auto 50px auto;}

div { display block; overflow:hidden; width:50%; margin: 50px auto 50px auto;}

.quote:before {content:”цитата”;}

.quote:after {content:”цитата после текста”;}

p:first-letter{ color:red; font-size:20pt; color:green }

p:first-line{ color:blue;}

div:first-line {font-size:16px;font-weight:bold; color:red}

first-line применяется только к блочным элементам !

Псевдоклассы

Псевдоклассы - это элементы, которые указывают определённый набор стилей, в случае различных событий и изменений состояния элемента. Например, как изменятся стили в случае наведения, клика и тд на элемент.
hover - при наведении курсора на элемент
focus - при клике на элемент, например поле для ввода данных
active - при активации элемента пользователем, то есть в момент клика
link - стиль для не посещённых ссылок
visited - стилевое оформление к посещенным ссылкам
firstchild - стиль для первого дочернего элемента селектора
lastchild - изменения в последнем элементе родителя

В CSS существуют псевдоклассы, которые работают со ссылками. У ссылок есть четыре состояния: простая, активная, посещенная и та, на которую наведен курсор. Состояние ссылок зависит от действия пользователя, и браузер, в зависимости от этих действий может применять разные стили. Для описания этих стилей и существуют псевдоклассы.
a:active - задает стиль активной ссылки.

a:visited - задает стиль посещенной ссылки.

a:hover - задает стиль ссылки, на которую наведен курсор.

По умолчанию ссылки синего цвета и подчеркнуты. Пусть на нашей html-странице есть ссылка:

```
<body>
    <div class=”exmpl”>
    <input type="text" value="Черный текст">
        <a href="#">ссылка 1</a>
        <a href="#">ссылка 2</a>
        <a href="#">ссылка 3</a>
        <a href="#">ссылка 4</a>
    </div>
 </body>
```

Давайте сделаем ссылку зеленой и уберем подчеркивание:

```
a{  color:green;  text-decoration:none;}
```

Свойство text-decoration отвечает как раз за подчеркивание, а его значение none указывает, что подчеркивать не надо.

Теперь, давайте зададим стиль для ссылки, на которую наведен курсор. Пусть она становится красного цвета:

```
a:hover{  color:red;}

a:active{  color: OldLace;}

a:visited{  color: NavyBlue;}

a:nth-child(2n) {background: #f0f0f0;}
```

Псевдокласс :nth-child используется для добавления стиля к элементам на основе нумерации в дереве элементов.

first-child применяет стилевое оформление к первому дочернему элементу своего родителя.

last-child задает стилевое оформление последнего элемента своего родителя.

```
a: first-child {  color:blue; }

a: last -child {  color:yellow; }

Input {color:black}

Input:focus {color:red}
```

Селекторы и их комбинации

Мы уже с вами знаем как в цсс записывать свойства для тега , ид , класса

```
<body>
        <h1 id=”first”>заголовок </h1>

        <h2 class=”second”>заголовок </h1>
        <h2 class=”second last”>заголовок </h1>
        <div class=”exmpl”>
        <input type="text" value="Черный текст">
                <a href="#">ссылка 1</a>
                <a href="#">ссылка 2</a>
                <a href="#">ссылка 3</a>
                <a href="#">ссылка 4</a>
        </div>
<a href="#">ссылка 5</a>
  </body>
```

Семантика

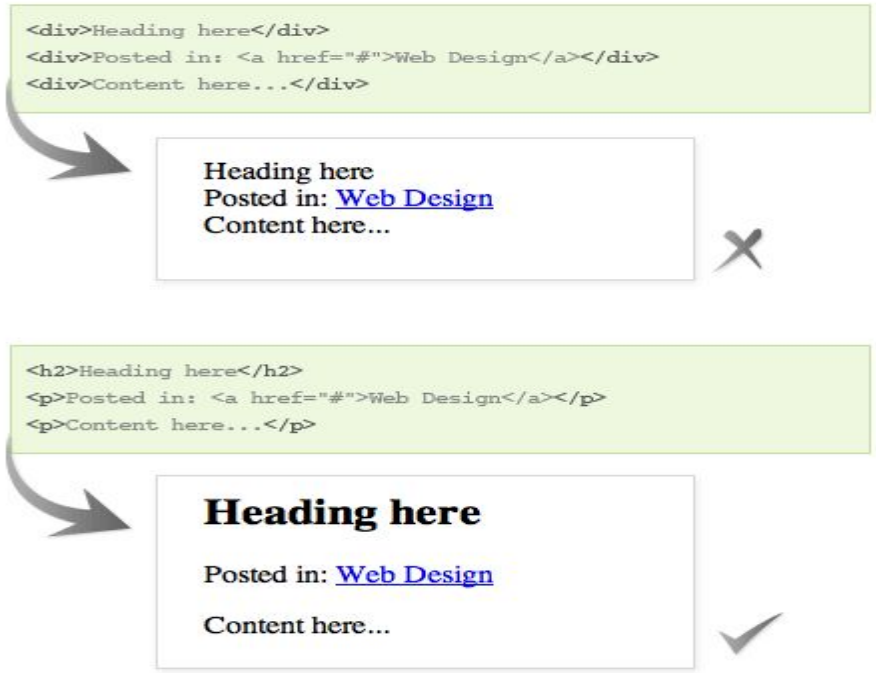
Семантика в HTML верстке - это использование HTML тегов по их назначению, правильное их наименование и их группировка. Так, например, тег h1 описывает заголовок документа. Заголовков есть шесть видов h1-h6, которые отличаются размерами и значимостью (в стандартных стилях они отличаются размерами, а отличие на вашем сайте должны сделать вы сами). Думаю, с заголовком все понятно. Есть также ссылки <a>, списки <dl>, <ul> и <ol>, таблицы <table> и много других. Если есть задача сделать таблицу, то каждый, не задумываясь, будет использовать тег <table>. Но бывают и спорные моменты.

Если в контенте есть ссылка на другую страницу, то ее мы просто делаем тегом <a>. Это понятно. А вот, например, есть кнопка со стилями cursor:pointer, которая имеет состояния :hover и :active . Ее тоже делать ссылкой? Нет! Ведь при клике на нее мы не переходим на другую страницу. Потому для такого элемента нужно использовать какой-то нейтральный тег - или . А если кнопка , которая визуально похожа на ссылку , находится в форме то однозначно надо использовать

<input type=" ">

Верстая форму, вам наверняка нужно будет делать подписи к полям формы - "Ваше имя" или "Введите email". Эти подписи нужно делать при помощи тега <label>. Именно он предназначен для данного элемента. К тому же, помимо семантики, вы получите и правильный функционал. Например, подписи в элементам <input type="checkbox"> и <input type="radio">, при клике на них, будут делать активными(выделенными) соответствующие элементы управления.

Возможности CSS позволяют из почти любого элемента сделать визуально другой вид, менять элементы местами, скрывать их и т.д. Для проверки правильности выбора тега, попробуйте отключить CSS. Если заголовок остался похожим на заголовок, список на список, а ссылка на ссылку, то вы идете в правильном направлении.



Также семантика связана и с валидностью. Тег <a> не может содержать другого тега <a>, в строчный элемент не может содержать блочный. Если ваша страница не прошла проверку на валидность https://validator.w3.org , то скорее всего у нее проблемы и с семантикой.

Разница между class(класс) и id(идентификатор) заключается в том, что идентификатор должен быть уникальным, то есть единственным на странице. Вообще, идентификаторами нужно пользоваться осторожно. Их чаще используют для обращения к ним из скриптов или при работе с формами.

Потому желательно не задавать элементам идентификаторы без особой необходимости. Скорее хорошим тоном будет еще задать идентификатор основным блокам - header, footer, content, breadcrumbs(хлебные крошки), search.

И так надо запомнить, что названия должны быть логичными и интуитивно понятными. Это также поможет ориентироваться в вашем коде другим людям и легче запоминать классы элементов.

```html
<div class="date">
  <div class="day">17</div>
  <div class="mth">Nov</div>
</div>
```

=

```html
<p class="date">
  17 <span>Nov</span>
</p>
```

```css
div.date {
  width: 50px;
  height: 50px;
  padding-top: 10px;
  background: #ccc;
  text-align: center;
}
div.date .day {
  font-style: italic;
}
div.date .mth {
  text-transform: uppercase;
}
```

```css
p.date {
  width: 50px;
  height: 50px;
  padding-top: 10px;
  background: #ccc;
  text-align: center;
  font-style: italic;
}
p.date span {
  text-transform: uppercase;
  font-style: normal;
  display: block;
}
```

HTML страница в общем случае должна быть поделена на блоки - header, footer, menu, left, right. Эти блоки делятся не только для красоты, но и по функциональности. В шапке сайта обычно лого компании, девиз, контакты - вся самая важная информация сайта. В контентной части - статья. Это самая важная информация данной страницы. А в боковой части - дополнительная информация. Это может быть список категорий, материалы по теме, другая информация. Страница сайта не может быть построена из простого набора блоков, они должны быть в правильном порядке и иметь четкую и правильную структуру.

Также важно разделять контент страницы и ее оформление. Для этого достаточно просто вынести все CSS стили в отдельный CSS файл, а скрипты - в JavaScript файл. Картинки должны быть на странице только в роли картинок - в теге <img> с атрибутом alt. А все изображения, относящиеся к оформлению, стоит сделать в виде блоков с фоновым изображением.

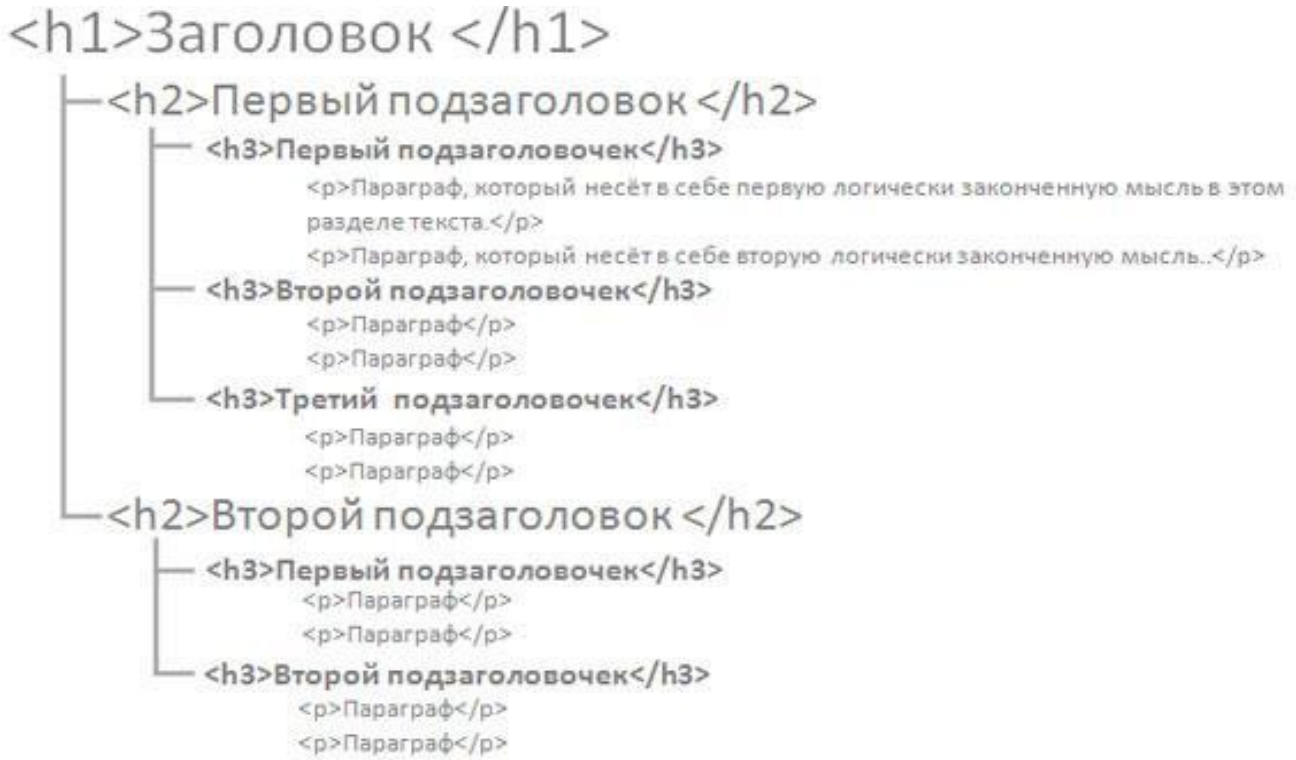
Не стоит прописывать стилизацию элемента в теге style, если нет особой необходимости. Ведь теперь есть псевдоклассы, при помощи которых вы можете выбрать элементы по их порядковому номеру, четные и нечетные, первые и последние.

Ну а для раскрутки сайта польза семантики, думаю, ясна. Поисковый робот первым делом ищет на странице заголовок по тегу <h1>. Ему будет сложно это сделать ,если ваш заголовок не находится в теге <h1>! :-) Также важно правильное оформление меню, хлебных крошек. Очередной раз напомню, что не обойтись без атрибутов title для ссылок и alt для картинок. Это нужно и для семантики, и для валидности, и для юзабилити.

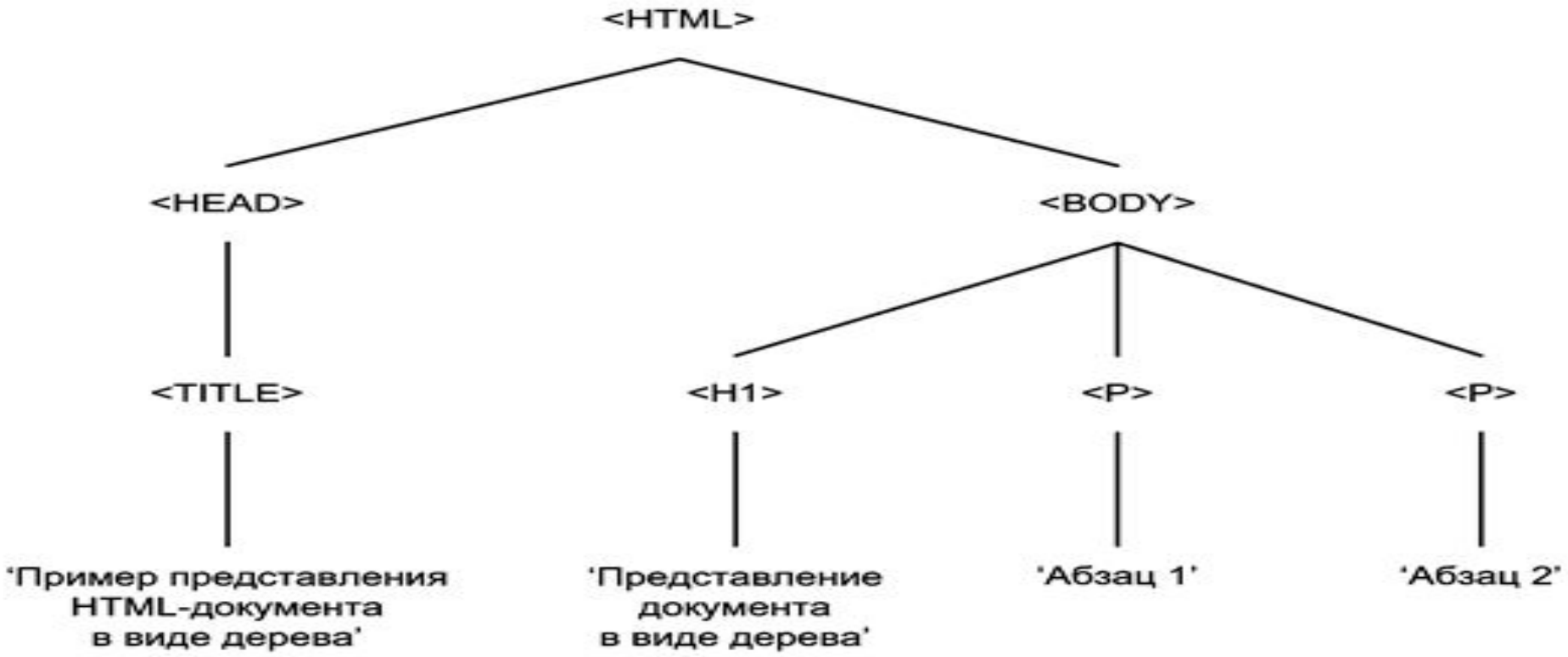
Заголовки должны выделяться тегами H1, H2, H3, H4, но никак не B и STRONG. Параграф - с помощью P тега, но никак не DIV. При создании меню лучше всего использовать UL список, внутри которого будут лежать LI элементы меню. Этим мы показываем, что ссылки равносильные. Если имеются вложенные списки, соответственно создаем внутри первичного LI элемента еще один UL список.

В HTML, тег IMG должен использоваться только для больших картинок и иконок. Не использовать атрибуты STYLE внутри HTML тега. Все стили выносить в отдельный CSS файл.

То же самое по поводу JavaScript.



```
<html>
<head>
      <tytle>Пример представления HTML-документа в виде дерева</tytle>
<head>
<body>
       <h1>Представление документа в виде дерева</h1>
       <p1>Абзац 1</p>
       <p1>Абзац 1</p>
</body>
</html>
```



Кроссбраузерность

Кроссбраузерность страницы – это верстка которая отображается одинаково во всех основных браузеров : Opera, Firefox, Internet Explorer , Safari, Google Chrome

Если в не основном для вас браузере какой то блок отображается не так как нужно , проверяйте семантику блока , далее стили которыми описали элемент в CSS , скорее всего что то написано не верно или не написано не полностью! Стили «по умолчанию» в разных браузерах РАЗНЫЕ !

Вендорные префиксы — это приставки к названию CSS свойства, которые добавляют производители браузеров для нестандартизированных свойств.

Согласно спецификации CSS 2.1 CSS идентификаторы, которые начинаются с "-" или "_" зарезервированы для CSS расширений браузеров. Наличие этих знаков в начале свойства гарантирует то, что в будущем расширения браузеров никогда не пересекутся со стандартными CSS свойствами. Т.е. ни один браузер не начнет «случайно» понимать свойство, которое для него не предназначено.

-o-	Opera
-moz-	Firefox
-ms-	IE
-ms-filter:	IE8
-webkit-	Safari, Google Chrome

Например, CSS свойство opacity, отвечающее за прозрачность элемента, кроссбраузерно используется так:

-ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";/* IE 8*/

-moz-opacity:0.5;/* Mozilla 1.6 */

-webkit-opacity:0.5/* Safari 2.0+ , Chrome, */

-o-opacity:0.5/* Opera, */

opacity:0.5

Вендорные приставки упрощают кроссбраузерную верстку но к ним следует обращаться в последнюю очередь!