Тема:PowerPoint

Что такое HTML

HTML *(от англ. HyperText Markup Language — «язык разметки гипертекста»;) — стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме.*

HTML представляет собой набор тегов, “описывающих” структуру документа. Вот ключевые:

Основные: html, head, title, body.

Структурные: div, span.

Текстовые: p, ul, ol, li, h1-h6, br, em, strong, b, i.

Таблицы: table, tr, td, th.

Ссылки: a.

Мультимедиа: img, object.

Фреймы: frameset, frame, iframe.

Формы: form, input, textarea, label, select, option.

Факультативные: hr.

Специальные: script, link, meta.

Теги в HTML-документе заключены в скобки <>. Кроме того, стоит запомнить, что теги бывают двух видов:

Парные, включающие в в себя открывающий и закрывающий тег (к примеру, какой-то текст).

Одиночные, состоящие только лишь из открывающего тега (например,
).

Благодаря тегам веб-браузер “идентифицирует” структуру текста. Речь идет о том, какая часть считается заголовком, какая — новым абзацем и пр.

Как выглядит HTML-документ. HTML-документ — это текстовый файл, имеющий расширение .html. Создание и редактирование HTML-документов выполняется как в обычном блокноте, так и в различных специализированных редакторах, например, Dreamweaver, Visual Studio и PHPStorm.

Чтобы разобраться в этом вопросе детальнее, откройте блокнот и добавьте в него следующие строки:

```
<html>
<head>
<title>
Моя страница</title>
</head>
<body>
Hello!!!
</body>
</html>
```

HTML

Язык HTML до 5-й версии определялся как приложение SGML (стандартного обобщённого языка разметки по стандарту ISO 8879). Спецификации HTML5 формулируются в терминах DOM (объектной модели документа).

Строгим вариантом HTML является XHTML, он наследует синтаксис XML и является приложением языка XML в области разметки гипертекста.

HTML-страницы, как правило, открываются браузерами обмениваясь с сервером информацией по протоколу HTTP или HTTPS, в виде простого текста или с использованием шифрования.

Общее представление

Язык гипертекстовой разметки HTML был разработан британским учёным Тимом Бернерсом-Ли приблизительно в 1986—1991 годах в стенах ЦЕРНа в Женеве в Швейцарии[3]. HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и семантических элементов — дескрипторов. Дескрипторы также часто называют «тегами». С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена
поддержка гипертекста. Мультимедийные возможности были добавлены позже.
Первым общедоступным описанием HTML был документ «Теги HTML», впервые упомянутый в Интернете Тимом Бернерсом-Ли в конце 1991 года,[4][5]. В нём описываются 18 элементов, составляющих первоначальный, относительно простой дизайн HTML. За исключением тега гиперссылки, на них сильно повлиял SGMLguid, внутренний формат документации, основанный на стандартном обобщенном языке разметки (SGML), в CERN. Одиннадцать из этих элементов всё ещё существуют в HTML 4[6]. Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <table> предназначен для создания в документах таблиц, но иногда используется и для оформления размещения элементов на странице. С течением времени основная идея платформонезависимости языка HTML была принесена в жертву современным потребностям в мультимедийном и графическом оформлении.

CSS

CSS *(англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.*

Использование CSS

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля

Способы подключения CSS к документу

Правила CSS могут располагаться как в самом веб-документе, внешний вид которого они описывают, так и во внешних файлах, имеющих расширение. CSS. Формат CSS — это текстовый файл, в котором содержится перечень правил CSS и комментариев к ним.

- когда описание стилей находится в отдельном файле, оно может быть подключено к документу посредством элемента <link>, включённого в элемент <head>.
- когда файл стилей размещается отдельно от родительского документа, он может быть подключён к документу инструкцией @import в элементе <style>.
- когда стили описаны внутри документа, они могут быть включены в элемент <style>, который, включается в элемент <head>
- когда стили описаны в теле документа, они могут располагаться в атрибутах отдельного элемента
- В первых двух случаях к документу применены *внешние стили*, а во вторых — *внутренние стили*.

Правила построения CSS

В первых трёх случаях подключения стилей CSS к документу (см. выше) каждое правило CSS из файла имеет две основные части — *селектор* и *блок объявлений*. *Селектор*, расположенный в левой части правила до знака «{» определяет, на какие части документа (возможно, специально обозначенные) распространяется правило. *Блок объявлений* располагается в правой части правила. Он помещается в фигурные скобки, и, в свою очередь, состоит из одного или более *объявлений*, разделённых знаком «;». Каждое *объявление* представляет собой сочетание *свойства CSS* и *значения*, разделённых знаком «:». Селекторы могут группироваться в одной строке через запятую. В таком случае свойство применяется к каждому из них.

селектор, **селектор**

{ свойство: значение;

свойство: значение;

свойство: значение;

}

Виды селекторов

Универсальный селектор

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

Селектор тегов

```
p {
font-family: arial, helvetica, sans-serif;
}
```

Селектор идентификаторов

```
#paragraph1 {
    margin: 0;
}
```

Селектор атрибутов

```
a[href="http://www.somesite.com"] {
 font-weight: bold;
}
```

История создания и развития CSS

CSS — одна из широкого спектра технологий, одобренных консорциумом W3C и получивших общее название «стандарты Web»[2]. В 1990-х годах стала ясна необходимость стандартизировать Web, создать какие-то единые правила, по которым программисты и веб-дизайнеры проектировали бы сайты. Так появились языки HTML 4.01 и XHTML, и стандарт CSS.

В начале 1990-х различные браузеры имели свои стили для отображения веб-страниц. HTML развивался очень быстро и был способен удовлетворить все существовавшие на тот момент потребности по оформлению информации, поэтому CSS не получил тогда широкого признания.

Термин «каскадные таблицы стилей» был предложен Хоконом Ли в 1994 году. Совместно с Бертом Босом он стал развивать CSS.

В отличие от многих существовавших на тот момент языков стиля, CSS использует наследование от родителя к потомку, поэтому разработчик может определить разные стили, основываясь на уже определённых ранее стилях.

В середине 1990-х Консорциум Всемирной паутины (W3C) стал проявлять интерес к CSS, и в декабре 1996 года была издана рекомендация CSS1.

Уровень 1 (CSS1)

Рекомендация W3C, принята 17 декабря 1996 года, откорректирована 11 января 1999 года. Среди возможностей, предоставляемых этой рекомендацией:

Параметры шрифтов. Возможности по заданию гарнитуры и размера шрифта, а также его стиля — обычного, курсивного или полужирного.

Цвета. Спецификация позволяет определять цвета текста, фона, рамок и других элементов страницы.

Атрибуты текста. Возможность задавать межсимвольный интервал, расстояние между словами и высоту строки (то есть межстрочные отступы)

Выравнивание для текста, изображений, таблиц и других элементов.

Свойства блоков, такие как высота, ширина, внутренние (pudding) и внешние (margin) отступы и рамки. Также в спецификацию входили ограниченные средства по позиционированию элементов, такие как <float> и <clear>.

Уровень 2 (CSS2)

Рекомендация W3C, принята 12 мая 1998 года[4]. Основана на CSS1 с сохранением обратной совместимости за несколькими исключениями. Добавление к функциональности:

Блочная вёрстка. Появились относительное, абсолютное и фиксированное позиционирование. Позволяет управлять размещением элементов по странице без табличной вёрстки.

Типы носителей. Позволяет устанавливать разные стили для разных носителей (например монитор, принтер, КПК).

Звуковые таблицы стилей. Определяет голос, громкость и т. д. для звуковых носителей (например для слепых посетителей сайта).

Страничные носители. Позволяет, например, установить разные стили для элементов на чётных и нечётных страницах при печати.

Расширенный механизм селекторов.

Указатели.

Генерируемое содержимое. Позволяет добавлять содержимое, которого нет в исходном документе, до или после нужного элемента.

Уровень 2, ревизия 1 (CSS2.1)

CSS2.1 основана на CSS2. Кроме исправления ошибок, в новой ревизии изменены некоторые части спецификации, а некоторые и вовсе удалены. Удалённые части могут быть в будущем добавлены в CSS3.

Структура HTML

Как театр начинается с вешалки, так и любой HTML-документ начинается с базовой структуры. Она включает в себя теги, которые есть в любом HTML-файле. Эти теги и служебная информация нужны браузеру для корректного отображения информации.

Взглянем на базовую структуру любого HTML-документа:

```
<!DOCTYPE html>
 <html lang="ru">
<head>
        <meta charset="UTF-8">
        <title>Моя первая страница</title>
 </head>
 <body>
</body>
</html>
```

Этот набор кажется не очень большим, но браузеру он сообщает множество полезной информации. В этом уроке разберёмся с каждой строчкой этой структуры.

DOCTYPE

Первая конструкция в любом HTML-документе — элемент DOCTYPE. Он не относится к тегам и никаким образом не может отображаться на странице. Его задача — указать браузеру, какой стандарт HTML используется в этом документе. Сейчас это везде стандарт HTML5. Записывается он следующим образом:

```
<!DOCTYPE html>
```

С приходом стандарта HTML5 элемент DOCTYPE немного упростился. Если вы встретитесь с сайтами, созданными пять-десять лет назад, то сможете увидеть совершенно другие записи. Они были больше и напрямую влияли на то, как браузер обработает информацию. Неправильное указание элемента DOCTYPE могло привести к некорректному отображению. Сейчас такой проблемы нет, поэтому вы можете без всяких опасений использовать конструкцию, которая указана в данном уроке. Использование старых значений DOCTYPE необходимо только при разработке с поддержкой очень старых браузеров.

Парный тег html

Тег <html></html> является основой основ. Именно внутри него располагается вся информация. Благодаря этому тегу браузер понимает, где начинается контент, который необходимо обработать как HTML.
Важной частью тега html является наличие атрибута lang. В нём указывается язык, на котором отображается веб-страница. С помощью этого атрибута браузеры могут корректно считать множество специфичных символов, которые присутствуют в разных языках. Помимо этого, атрибут lang начинает использоваться и в CSS, с которым вы познакомитесь в следующих уроках. В новых стандартах CSS появляются свойства, которые опираются на данный атрибут. Например, позволяют корректно переводить слова в тексте.
В качестве значения атрибут lang принимает знакомые всем сокращения языков. Для русского — *ru*, для английского — *en*, для немецкого — *de*.

Парный тег head

Тег служит для хранения служебной информации. Здесь возможны самые разные сочетания тегов, которые подсказывают браузеру название страницы, описание, ключевые слова и так далее. Такая информация называется *метаинформацией*. В современном вебе она отвечает не только за служебную информацию для браузера, но и активно используется при продвижении сайта. Поисковые системы считывают всю эту информацию и на основе множества алгоритмов определяют место сайта при разных поисковых запросах.

Любые данные, которые указаны внутри тега head, не видны при отображении страницы в браузере. Это значит, что нет необходимости располагать там информацию, которая предназначена для отображения.

Хоть различной информации внутри head может быть множество, в этом уроке разберём несколько основных тегов, которые пригодятся при создании любой веб-страницы:

Метаинформация

Метатег <meta>. Он принимает множество разных атрибутов, с которыми вы познакомитесь при создании своих сайтов. В настоящее время важным является метатег <meta> с атрибутом charset. Он позволяет установить кодировку документа.
Кодировка — таблица символов. В ней каждый символ имеет уникальный код, благодаря чему программы, в том числе и браузеры, могут одинаково отображать один и тот же текст. У разных пользователей может стоять различная кодировка по умолчанию. Это приводит к тому, что у некоторых пользователей текст может отображаться в виде «кракозябр», хотя у вас он будет отображаться правильно. Универсальной кодировкой, которая содержит большинство необходимых символов из разных языков является кодировка *UTF-8*. Именно её рекомендуется устанавливать в качестве значения атрибута charset. Теперь браузер будет отображать все символы именно в этой кодировке.

Структура CSS.

Синтаксис CSS
Порядок написания свойств стилей довольно прост:
Селектор { свойство: значение; свойство2: значение; и. т. д.}

Селектор — определяет элемент HTML, к которому применяется форматирование.

Свойство — показывает параметр этого элемента HTML.

Значение — показывает какие изменения должен претерпеть данный параметр.

Пример:
Body { margin: 0; padding: 0 }
Селектором в данном примере служит тег body, свойствами — аргументы margin и padding, а их значения равны нулю. Все элементы после селектора заключаются в фигурные скобки. После «имени» свойства ставится двоеточие, и после пробела следует значение свойства. Значение замыкается точкой с запятой, по завершении описания стилей селектора точку с запятой можно не ставить. Данный пример написания стилей использует строчную запись, но когда свойств много, их удобнее расположить вертикально и разделить. Например:

```
Body {
margin: 0;
padding: 0;
font-size: 12px;
color: red;
}
```

При большом количестве значений одного свойства разрешается применение сокращённой записи. А вот и пример:

```
H2{
font-family: arial, sans-serif;
font-size: 20px;
font-style: italic;
font-weight: bold;
}
```

Сокращённая запись стилей заголовка H2 будет выглялеть так:

```
H2{ font: arial,sans-serif 20px italic bold; }
```

Как видите, после свойства FONT запись его значений идёт через пробелы в одну строчку. Браузер сам определяет подходящие значения и применяет их.

Для того, чтобы таблицы стилей работали с HTML-страницами их необходимо связать друг с другом. Это делается тремя способами:

Принципы создания сайта на HTML

Основные принципы создания Web-страниц. Язык HTML 5

Web-страницы выглядят зачастую очень пестро: разнокалиберные куски текста, таблицы, картинки, врезки, сноски и даже фильмы. Но описывается все это в виде обычного текста. Да-да, Web-страницы — суть текстовые файлы, которые можно создать с помощью хорошо знакомого нам редактора Блокнот, поставляемого в со- ставе Windows! (Разумеется, подойдет любой аналогичный текстовый редактор.)

Для форматирования содержимого Web-страниц применяется особый язык — *HTML* (HyperText Markup Language, язык гипертекстовой разметки). С помощью команд — *тегов* — этого языка создают и абзацы текста, и заголовки, и врезки, и даже таблицы.

Первая версия языка HTML появилась очень давно, еще в 1992 году. С тех пор по Сети утекло немало гигабайт… HTML также не стоял на месте. В данный момент готовится к выходу окончательная спецификация новой версии HTML под номером 5, и многие Web-обозреватели уже поддерживают некоторые ее возможности. Ее-то мы и будем изучать.

Освоение языка html и изучение **тегов html** лучше всего самостоятельно создавая свою первую интернет страничку. Для этого нам будет достаточно уже внедренного в Windows приложения — обыкновенный "Блокнот".

Но чтобы вы были в курсе: для написания Web-страниц разработано много спец. приложений — Web-редакторов. С их помощью, даже не представляя, что такое язык html, можно штамповать web-странички не заморачиваясь. К примеру всем знакомый редактор Adobe Dreamweaver.

Однако, на данном этапе, использование таких программ большей пользы нам не принесет. Пользуясь обыкновенным "Блокнотом" вы сумеете более основательно изучить язык html и **теги html**. Даже примеры html-кодов здесь поданы не как текст, а как изображение. Скопировать не удастся, только набирать с клавиатуры.

И так, открываем Блокнот и набираем в нем html-код, который видим в листинге 1.1.

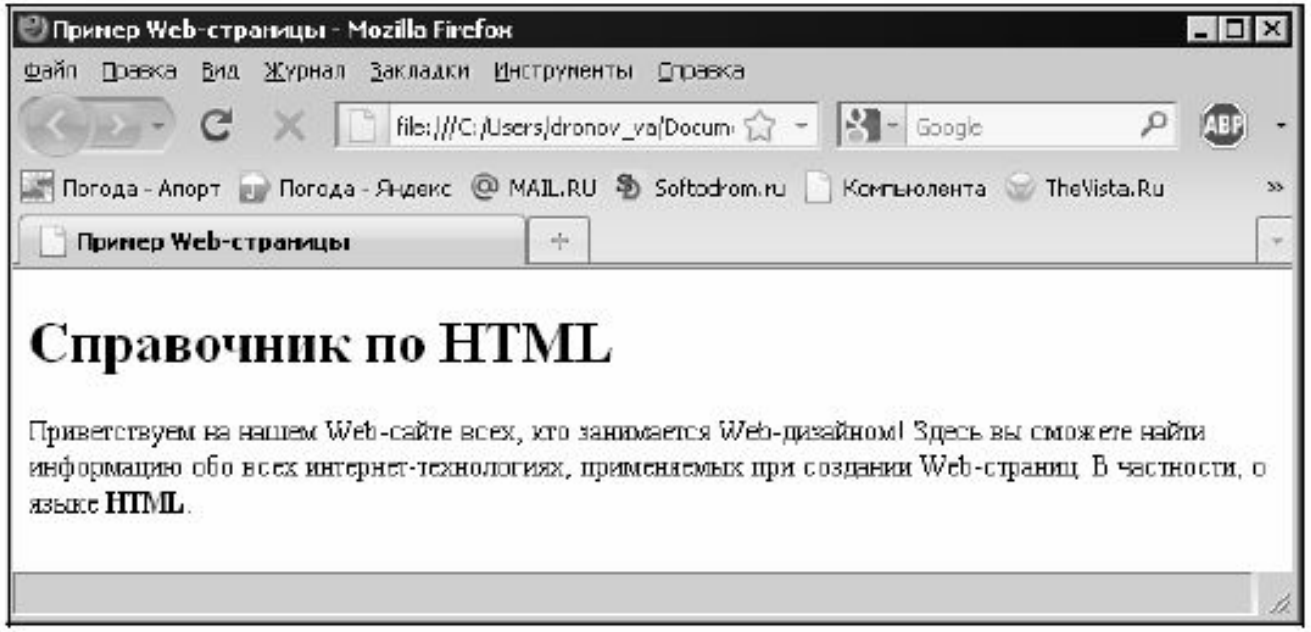
Листинг 1.1

```
<!DOCTYPE html>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
    <TITLE>Пример Web-страницы</TITLE>
  </HEAD>
  <BODY>
    <H1>Справочник по HTML</H1>
    <P>Приветствуем на нашем Web-сайте всех, кто занимается Web-дизайном!
    Здесь вы сможете найти информацию обо всех интернет-технологиях,
    применяемых при создании Web-страниц. В частности, о языке
    <STRONG>HTML</STRONG>.</P>
  </BODY>
</HTML>
```

После написания сохраняем созданный нами файл ("сохранить как"), но при этом дадим имя файлу, например, dokument1. Вместо расширения .txt проставим расширение .htm и сменим кодировку файла на UTF-8. Вот так мы создали файл dokument1.htm Осталось лишь проверить как он выглядит в браузере.

Но есть одно "но". Созданные вами страницы сайтов, по возможности, желательно проверять в нескольких более распространенных браузерах, так как не все web-обозреватели могут одинаково отображать html-документ. Так что могут всплывать непредвиденные ошибки.

Получили вот такой результат



Вот наша первая web-страничка. Мы видим крупный заголовок. Абзац, разделенный на строки автоматически и одно слово, выделенное полужирным шрифтом.

Давайте теперь разберемся, что же мы прописали в нашем файле dokument1.htm. Рассмотрим, пока, только небольшую часть html-кода (листинг 1.2)

Листинг 1.2

```
<H1>Справочник по HTML</H1>
<P>Приветствуем на нашем Web-сайте всех, кто занимается Web-дизайном!
Здесь вы сможете найти информацию обо всех интернет-технологиях,
применяемых при создании Web-страниц. В частности, о языке
<STRONG>HTML</STRONG>.</P>
```

Первым делом мы наблюдаем текст, который, при открытии файла dokument1.htm виден в браузере. А еще видим буквы и слова, заключенные в скобки угловые < и >. Вот они как раз и называются - теги HTML. Это, именно, с их помощью та или иная часть html-кода преобразуется в какой то элемент страницы.

Теги <h1> и </h1>. Текст, прописанный внутри этих двух тегов, на странице будет заголовком и имеет название "содержимое" тега. Тег <h1> отмечает начало и называется "открывающий" тег. </h1> завершает фрагмент текста и называется "закрывающий". Различие между ними лишь в наличии слеша /. Теги, имеющие и "открывающий" и "закрывающий" тег называют парными.

И так, мы выяснили, что все **html-теги** — это угловые скобки < и > внутри которых размещено имя тега. Именно оно определяет предназначение тега.

Продолжим. Далее мы видим html-тег <p>. Есть и открывающий тег и закрывающий — значит этот **тег html** парный. Текст, прописанный внутри него, становится абзацем web-страницы.

Парный тег вложили внутрь тега <p>. Значит все, что внутри тега будет частью абзаца (тега <p>). Как вы уже заметили, этот тег отвечает за написание текста полужирным шрифтом.

В завершение запомним три правила написания кода html:

- имя тега можно прописывать и строчными (маленькими) буквами и прописными (большими)
- внутри угловых скобок < и > и в самом имени тега не должно быть переноса строк и пробелов
- в прописанном тексте не допустимо присутствие символов угловых скобок < и >. Это, так званые, недопустимые символы.

Часть 1. Соблюдайте HTML и CSS стандарты

На всякий случай скажу, что кроме HTML и CSS мы ещё можем оптимизировать JS и PHP части наших сайтов, которые являются более ресурсоёмкими и значительнее замедляют скорость загрузки страниц. Однако кроме скорости загрузки страниц, JS и PHP составляющие никак не влияют на продвижение сайта (лишь бы не ломали его), в отличие от HTML разметки, то есть шаблона сайта.

1.1 Всегда закрывайте теги.

В том числе те, которые закрывать не обязательно, даже если вы ненавистник XHTML стандарта, лучше придерживаться этого правила. Пример:

```
<ul>
 <li>Пункт списка</li>
<li>Пункт списка</li>
 <li>Пункт списка</li>
 </ul>
```

Каждый тег li закрыт, хотя это и не обязательно.

1.2 Соблюдайте правильную вложенность тегов.

По принципу: первым открылся — последним закрылся. Пример как не правильно:
текст

Как правильно:

текст

1.3 Никогда не описывайте CSS и JS внутри HTML макета и атрибутов тегов.

Никогда! Другими словами, забудьте о существовании **атрибута style** и **тега style**:

текст

<!-- или -->

<style>

p{

color:red;

font-size:20px;

text-align:center

}

</style>

Описывайте все стили в **отдельном файле .css**

Можете использовать атрибут style, только динамически отрисовывая его с помощью яваскрипта по какому-нибудь действию. Например, если по клику на картинку, нужно изменить цвет всего текста на странице, то только в таком случае можете динамически создать атрибут style с нужными значениями для нужного тега (в данном примере – для body). Почему? Потому что поисковые роботы его всё равно не увидят и никто (в том числе пользователи) не загрузят лишний код, так как действие происходит уже после полной загрузки страницы.

Тем не менее, в таких случаях я всё равно рекомендовал бы добавлять заранее описанный CSS класс нужному тегу, нежели добавлять ему атрибут style. Это как минимум удобней для последующего редактирования.

Всё то же самое относиться и к JS, весь JS-код должен быть в **отдельном файле**, а не внутри вашего макета или ещё хуже — среди атрибутов тегов.

1.4 Забудьте о Caps Lock и заглавных буквах.

Пишите все теги, атрибуты и их значение строчными (маленькими) буквами, это же касается и таблиц стилей CSS.

Почти всё описанное выше есть в официальной спецификации HTML и CSS и относится к валидности документа.

Таким образом, я должен отметить, что придерживаясь W3C стандартов, то есть, соблюдая валидность документа — вы получаете следующие SEO преимущества:

Чистый код, а, следовательно, и дополнительное доверие поисковых систем к вашему сайту;

Ускоренная загрузка страниц, так как браузеру не приходится тратить время на отладку невалидного документа.

Однако не стоит зацикливаться на валидности, вы должны стараться максимально её придерживаться, однако не в ущерб всему остальному (времени, функциональности и тд.).

Часть 2. Ускоряем загрузку страниц – один из факторов ранжирования

2.1 Указывайте настоящие размеры картинок.

Здесь суть в двух вещах:

Обязательно указывайте атрибуты **width** и **height** для тега **img**:

Это ускорит загрузку изображений, так как браузер заранее будет знать, какой размер нужно отобразить.

Обязательно вставляйте картинку того же самого размера, который указали в атрибутах.

Если картинка больше чем вам необходимо, то не нужно экономить время на том, что её можно уменьшить по средствам HTML или CSS. Будьте любезны потратить время, зайти в Фотошоп или его аналоги и уменьшить картинку до тех размеров, которые вы хотите видеть на сайте, это может заметно ускорить загрузку страниц.

2.2 Используйте CSS3 вместо JS.

JS может значительно замедлить загрузку ваших страниц, а также продолжать создавать нагрузку уже после загрузки страницы (разнообразные эффекты, типа бегущей строки и другие динамические вещи), что может тормозить работу браузера пока вкладка с сайтом будет открыта.

Сегодня множество красивых и интересных эффектов можно достичь только благодаря использованию свойств CSS3 (например, transition, box-shadow, border-radius, opacity, transform, background-size) и умелой **комбинации селекторов**.

Всё сказанное выше можно подытожить одной фразой: **везде, где вы можете (подозреваете как) заменить JS на CSS3, используйте CSS3 не раздумывая!** Для поиска подобных фрагментов советую вам изучить новые возможности CSS3.

2.3 Меньше Photoshop – больше CSS 3.

Этот девиз уже давно используют многие вебмастера. Вы можете создавать красивые кнопки и элементы дизайна с плавным градиентом, сглаженными углами, тенями (внутренними и внешними) и красивым текстом сверху благодаря одному лишь CSS. Везде, где можно заменить графические элементы дизайна на CSS код – делайте это! Пример:Или моя кнопка "Подписаться!" в конце каждой статьи. Кстати, не забываем подписаться на оповещения о новых полезных статьях, никакого спама или рекламы.

Помимо перечисленных выше свойств CSS3 вам также доступен формат описания цвета RGBA.

Для создания кроссбраузерного градиента на CSS вы можете использовать бесплатные сервисы, например этот **colorzilla.com**.

2.4 Объединяйте изображения в CSS спрайты.

Те изображения, которые мы не смогли нарисовать с помощью CSS 3, нужно обязательно объединить в одно единственное (в идеале). Это нужно для того, чтобы сократить количество запросов к серверу при загрузке страницы. Этот пункт может значительно снизить нагрузку на ваш веб-сервер, а заодно и ускорить загрузку страниц, сразу 2 зайца!

Неплохо, не правда ли? Фоновая картинка одна на все элементы, мы только двигаем её и подставляем в фон определённого элемента нужную часть картинки благодаря свойству background-position, например , так:

```
subs,#left,.mail,.rss,.vk{background:url(/images/1.png) no-repeat}

#subs{background-position: -28px -120px;}

#left{background-position: -35px -20px;}

.mail{background-position: -43px -50px;}

.rss{background-position: -12px -8px;}

.vk{background-position: -34px -56px;}
```

Лучше всего будет составить спрайт вручную, с помощью Фотошопа, но это может показаться вам достаточно сложным занятием, поэтому вы можете использовать бесплатные сервисы, которые всё сделают за вас, в том числе даже напишут за вас CSS-код. Мне большего всех нравится сервис **SpriteMe**. Но я не устану повторять: всегда всё лучше делать вручную, в частности, так спрайт может получиться значительно компактней (по размерам и, следовательно, по весу), а значит более эффективным.

2.5 Размещайте JS файлы правильно.

Все подключаемые вашим сайтом JS файлы должны находится как можно ниже по коду, не нужно подключать их внутри «головы» сайта (между тегами head) – это значительно замедляет загрузку страниц. Самым лучшим вариантом будет подключить файлы JS перед закрывающимся тегом body, то есть в самом низу страницы, ниже уже некуда.

И ещё один очень полезный совет: в идеале все ваши файлы JS нужно объединить в один единственный, то есть вырезаем (Ctrl + X) код из всех файлов и копируем его в один, чтобы в итоге у вас внизу, перед закрывающимся тегом body подключался один единственный файл:

```
...

<script src='/scripts/all.js' type='text/javascript'></script>

</body>

</html>
```

Это может заметно ускорить загрузку ваших страниц, так как снизит количество запросов к вашему веб-серверу. Расположение JS файла внизу страницы отложит его загрузку и обработку, что ускорит вывод основной части страницы. И не забудьте сократить (сделать компрессию) ваш конечный JS файл.

2.6 Размещайте CSS файлы правильно.

Файл CSS, который, кстати говоря, тоже должен быть одним единственным (по тем же причинам), нужно размещать в коде наоборот, как можно выше!

Но не нужно размещать его выше тега title, он самый главный для SEO и лучше его оставить в самом верху. Лично я подключаю CSS файл сразу после тега title и всех мета-тегов, а уже после него можно подключить, например, фавикон (иконка сайта во вкладке браузера) и тд.

Редакторы кода.

Писать код при желании можно и в текстовом редакторе — ничто не мешает вам создать простейший сайт в «Блокноте», сохранив файл с расширением .html. Однако если вы хотите сделать процесс комфортнее и быстрее, стоит обратить внимание на интегрированные среды разработки (Integrated Development Environment, IDE) или продвинутые редакторы. В этой подборке мы собрали 10 популярных платформ, которые предлагают удобные функции для веб-разработчиков.

Что такое IDE и зачем она вам

Существует немало функций IDE, которые вы вряд ли встретите в более простых инструментах, особенно если работаете над созданием веб-приложения или довольно сложного сайта. Вам, скорее всего, пригодятся:

— **компилятор**: превращает ваш код в исполняемый файл;

— **интерпретатор**: запускает скрипты, которые не нужно компилировать;

— **отладчик**: позволяет находить проблемные места и ошибки в коде;

— **инструменты автоматизации**: помогают автоматизировать сборку проекта и ускорить процесс разработки.

В IDE все эти элементы обычно объединяются в единую платформу.

Несмотря на многие преимущества IDE, на самом деле они нужны не всегда. Если вы занимаетесь в основном разработкой веб-интерфейсов, вполне можно обойтись и стандартным редактором кода. Также IDE не стоит использовать для создания простых статических сайтов, иначе вы можете начать стрелять из пушки по воробьям: более сложные инструменты скорее замедлят процесс, чем сделают его эффективнее.

На что обратить внимание при выборе среды разработки

1. Поддержка нужной вам операционной системы (ОС). Особое внимание этому пункту стоит уделить, если вы работаете в команде. Лучше всего отдавать предпочтение кроссплатформенным решениям.

2. Возможности совместной разработки. Это опять же относится к командам, собирающимся работать с общим репозиторием. Многие платформы, которые мы рассмотрим ниже, интегрируются с Git.

3. Поддерживаемые языки (программирования, разумеется). Здесь не забывайте о долгосрочной перспективе — вдруг когда-нибудь вы решите добавить в проект возможности, реализуемые на каком-либо другом языке. Стоит выбрать среду, которая поддерживает несколько языков программирования.

4. Цена вопроса. Есть много бесплатных решений с открытым исходным кодом. Однако, как обычно бывает почти со всем подобным программным обеспечением, стоимость зависит от количества доступных функций.

Чтобы помочь вам определиться, мы собрали 10 лучших IDE и редакторов кода, которые поддерживают популярные языки для веб-разработки (HTML, CSS, JavaScript, PHP и Python). Сразу оговоримся, что это не топ, а список (первый — не значит лучший, последний — не значит самый плохой). Поэтому вы можете выбирать любой инструмент, исходя из своих нужд и предпочтений.

1. Visual Studio + Visual Studio Code

IDE от Microsoft, Visual Studio, доступна только для операционных систем Windows и macOS. Поддерживает Python, PHP, JavaScript, HTML, CSS и многие другие языки.

Visual Studio обладает всеми преимуществами IDE, включая удалённую отладку. Кроме того, платформа содержит:

— Умное дополнение кода IntelliSense, чтобы ускорить процесс написания программ;

— Инструменты для совместной работы: управление доступами и настраиваемые параметры редактора позволят писать код в едином стиле;

— Интеграцию с Git;

— Простое развёртывание благодаря встроенной интеграции с Azure.

К недостаткам Visual Studio можно отнести стоимость: цены на лицензии Professional, предназначенные для профессиональных команд разработчиков, начинаются от 45 $ в месяц. Корпоративная лицензия обойдётся в 1199 $ за первый год, продление — 799 $ в год.

Visual Studio Code

В качестве более простого решения можете рассмотреть бесплатный, но очень мощный и популярный редактор Visual Studio Code — он предлагает не так много возможностей, как IDE, зато позволяет писать код более чем на 72 языках и включает функции отладки. VS Code поддерживается не только на Windows и macOS, но и на Linux.

В редакторе есть умное автодополнение IntelliSense, встроенная интеграция с Git, а также огромная библиотека расширений.

А ещё разработчики GitHub собираются встроить VS Code прямо в браузер с помощью инструмента Codespaces, чтобы можно было вносить изменения в проект, не выходя из GitHub. Сейчас Codespaces находится на этапе бета-тестирования.

2. IntelliJ IDEA

IntelliJ IDEA — Java-ориентированная платформа для разработки от JetBrains. Несмотря на это, она позволяет работать со всеми языками, которые мы упоминали выше (HTML, CSS, JavaScript, PHP и Python). Из коробки вам будут доступны инструменты для написания кода на HTML, CSS и JavaScript (в версии Ultimate). Поддержку PHP и Python можно добавить с помощью плагинов.

IntelliJ IDEA доступна для систем Windows, macOS и Linux. Ключевые функции:

— Умное автодополнение, которое предлагает элементы кода исходя из текущего контекста;

— Встроенная отладка;

— Встроенная интеграция с системами контроля версий;

— Интеграция с инструментами сборки, такими как Apache Maven, Gradle и Webpack.

IntelliJ IDEA поставляется в трёх ценовых вариантах. Community-версия доступна бесплатно, однако она не включает себя поддержку JavaScript и работу с инструментами базами данных, что может быть критично для веб-разработки. Стоимость индивидуальной лицензии IntelliJ IDEA Ultimate — 149 $ в год, для организаций же цена составит 499 $ на пользователя в год. Также можно попробовать версию Ultimate бесплатно в течение 90 дней.

3. PyCharm

Если вы занимаетесь разработкой на Python, то присмотритесь к PyCharm — ещё одной IDE от JetBrains. Как и IntelliJ, она поддерживается всеми тремя основными операционными системами. Professional-лицензия включает поддержку HTML, JavaScript и CSS. Кроме того, вы всегда можете расширить функционал с помощью плагинов.

С PyCharm вам будут доступны:

— Автодополнение кода и автоматический поиск ошибок;

— Интеллектуальная навигация по проекту;

— Встроенные отладчик, профилировщик Python и терминал;

— Интеграция с популярными системами контроля версий, а также с Jupyter Notebook, Anaconda и другими библиотеками.

Как и IntelliJ IDEA, PyCharm имеет Community-версию с открытым исходным кодом, но с ограниченными функциями — в ней отсутствуют многие инструменты для веб-разработки, нет профилировщика Python и поддержки баз данных.

Professional лицензия стоит 89 $ за год для частных лиц и 199 $ в год для организаций (за одного пользователя).

4. PhpStorm

Если вам больше по душе PHP, то обратите внимание на PhpStorm от JetBrains. Эта IDE имеет много общего с IntelliJ IDEA и PyCharm. Вы можете использовать её на Windows, macOS и Linux, и она поддерживает разработку на JavaScript, CSS и HTML.

Кроме того, PhpStorm рекомендуется JetBrains для работы с популярными CMS:
например WordPress, Drupal, Joomla и другими.

Функциональность включает в себя:

— Автодополнение кода и рефакторинг;

— Эффективные функции навигации;

— Встроенная интеграция с системами контроля версий, инструменты командной строки, управление базами данных SQL;

— Визуальный отладчик и функция Live Edit, позволяющая сразу посмотреть, как будут выглядеть изменения в браузере.

Что касается цен, то PhpStorm, в отличие от других продуктов JetBrains, не предлагает бесплатную лицензию. Стоимость начинается от 89 $ в год для индивидуального использования и 199 $ для организаций. Также доступна 30-дневная пробная версия.

5. WebStorm

ля разработчиков на JavaScript JetBrains предлагает платформу WebStorm. Она поддерживает популярные фреймворки для фронтенда (Angular, React, Vue.js) и бэкенда (Node.js, Meteor). Среди преимуществ IDE можно выделить:

— Умное автодополнение кода;

— Встроенный отладчик;

— Инструменты для тестирования Karma, Mocha, Protractor и Jest;

— Интеграция с популярными системами контролями версий.

WebStorm не имеет бесплатной лицензии, цена для индивидуального использования — 59 $ в год, для компаний — 129 $ в год на пользователя.

6. Komodo IDE

Komodo IDE от ActiveState позиционирует себя как «одна IDE для всех языков». И это действительно так: платформа поддерживает JavaScript, HTML, CSS, Python, PHP и множество других языков программирования.

Ключевые особенности Komodo IDE:

— Интеллектуальная подсветка синтаксиса и автодополнение кода;

— Визуальный отладчик и инструменты для тестирования;

— Предварительный просмотр страниц: не нужно переключаться между IDE и браузером;

— Интеграция с Devdocs.io для удобного поиска документации;

— Поддержка популярных систем контроля версий;

— Профилирование кода на Python и PHP.

Community-лицензия Komodo для одного пользователя полностью бесплатна. Расширенная индивидуальная лицензия стоит 84 $ в год, а цены на тарифы для команд разработчиков и бизнеса стартуют от 228 $ в год за одного пользователя.

7. Sublime Text

Строго говоря, Sublime Text больше похож на редактор кода, но он содержит функции, аналогичные полноценным IDE. Sublime Text доступен для всех трёх основных операционных систем и поддерживает HTML, CSS, JavaScript, PHP, Python и другие языки. Кроме того, он включает в себя несколько функций, позволяющих ускорить и упростить редактирование кода, например:

— Навигация Goto Anything для быстрого перехода к файлам, строкам или словам;

— Быстрое внесение изменений сразу в нескольких местах;

— Интеграция с Git через Sublime Merge;

— Быстрое переключение между несколькими проектами с фиксацией изменений.

Если вы хотите расширить возможности Sublime Text, подключите к нему плагины для автозаполнения, отладки и других расширенных функций.

Sublime Text можно скачать бесплатно, однако для дальнейшего использования необходимо приобрести лицензию. Индивидуальный план стоит 80 $ в год, а цена на бизнес-лицензию зависит от числа людей в команде (минимально — 50 $ в год за пользователя для команды от 50 человек).

8. Brackets

Brackets — редактор с открытым исходным кодом, который отлично подойдёт для разработчиков веб-интерфейсов. Доступен для Windows, Linux и macOS. Из коробки поддерживает HTML, CSS и JavaScript, а PHP и Python можно подключить через LSP.

Brackets позволяет редактировать файлы в режиме реального времени: вы можете следить за внешним видом вашего проекта по мере внесения изменений без необходимости перезагрузки страницы.

Также в Brackets есть множество расширений для интеграции с Git, автодополнения кода и других полезных фич.

Хотя Brackets тоже больше относится к редакторам кода, чем к полноценным IDE, он полностью бесплатен и даёт простор для экспериментов: вы можете сами создавать расширения для него или даже переписать исходный код программы под себя.

9. Atom

Atom — редактор кода с открытым исходным кодом от GitHub. Как можно догадаться, он имеет встроенную интеграцию с Git и GitHub. Atom совместим с Windows, macOS и Linux, а также позволяет:

— Совместно редактировать код в режиме реального времени с помощью Teletype;

— Быстро писать код с умным автодополнением;

— Разделять интерфейс редактирования, чтобы одновременно работать с несколькими файлами или проектами;

— Устанавливать расширения с помощью встроенного менеджера пакетов.

Также вы можете попробовать Atom IDE — расширенную версию Atom, более похожую на полноценную IDE.

10. NetBeans

NetBeans — платформа с открытым исходным кодом от Apache, включающая множество функций, необходимых для веб-разработки. Совместима с системами Windows, Linux и macOS. NetBeans больше ориентируется на Java, но по умолчанию также доступны JavaScript, HTML и CSS. PHP и Python можно добавить с помощью плагинов.

Функциональность NetBeans включает:

— Умное редактирование и автодополнение кода;

— Настраиваемые сочетания клавиш для более быстрой работы;

— Визуальный отладчик;

— Интеграция с Git, Maven и другими платформами.

NetBeans полностью бесплатна и открыта для вклада разработчиков, желающих усовершенствовать платформу.

Бонус: какими IDE пользуются разработчики REG.RU

Разумеется, в этой подборке мы привели лишь малую часть существующих сред разработки и редакторов кода. Например, можно было бы упомянуть о Notepad++, Eclipse или активно развивающихся облачных IDE вроде Codeanywhere или Cloud9.

Тем не менее, мы в том числе перечислили инструменты, которыми пользуются разработчики в REG.RU. Например, те, кто так или иначе взаимодействует с Python, отдают предпочтение PyCharm. Разработчики интерфейсов больше склоняются к редакторам, чем к IDE, и выбирают Visual Studio Code и Sublime Text. Также встречаются и те, кто работает в WebStorm, Komodo, Atom. Многие, кстати, считают идеальным редактором Vim: конечно, в нём нет того обилия функций, которые предлагают IDE, однако его вполне можно превратить в удобную для работы среду с помощью многочисленных плагинов и расширений.