**ОТВЕТЫ НА ВОПРОСЫ К ЭКЗАМЕНУ ПО ДИСЦИПЛИНЕ**

**«КОМПЬЮТЕРНЫЕ ЯЗЫКИ РАЗМЕТКИ»**

1. Представление о макете веб-страницы.



HTML-элемент <header> представляет собой **вводный контент, обычно группу вводных или навигационных средств**. Он может содержать другие элементы-заголовки, а также логотип, форму поиска, имя автора и другие элементы.

HTML-элемент <footer> представляет собой **нижний колонтитул (футер, подвал) для своего ближайшего секционного контента или секционного корня**.

<nav> - элемент навигации HTML-элемент <nav> **определяет отдельную секцию документа, назначение которой обозначение ссылок навигации (как внутри текущего документа, так и ведущих на другую страницу)**.

HTML-элемент <main> **предназначен для основного контента (содержимого) <body> документа (страницы)**

HTML-элемент <article> **представляет самостоятельную часть документа, страницы, приложения или сайта, предназначенную для независимого распространения или повторного использования.**

HTML-элемент <section>  **задаёт раздел документа, может применяться для блока новостей, контактной информации, глав текста, вкладок в диалоговом окне и др**.

HTML-элемент <aside> **представляет собой часть документа, чьё содержимое только косвенно связанно с основным содержимым документа**.

1. Рекомендации для составления веб-страницы.

Любая Web-страница структурно разбивается на две части: заголовок и тело. В заголовке указывается служебная информация обо всей странице, а в теле Web-документа описывается ее содержимое вместе с правилами оформления.

При этом заголовок Web- ограничивается тэгами <head> и </head>, а тело документа обозначается тэгами <body> и </body>. По правилам хорошего стиля программирования перед заголовком ставится идентификатор применяемого стандарта HTML.

В заголовок документа могут входить: тэг, отображающий наименование Web-страницы, тэг стилевого оформления, тэг выполняемого сценария и так называемые метаданные.

Между тэгами **<title>** и **</title>** указывается название страницы, которое отображается в строке заголовка окна при просмотре странички в браузере.

Для внедрения метаданных в Web-страницу применяется тэг <META>. Тегов <META> может быть несколько.

В теле документа, между тэгами **<body>** и **</body>** заключается гипертекст, который определяет собственно WEB-страницу. Это та произвольная часть документа, которую разрабатывает автор и которая отображается браузером. Внутри элемента **<body>** можно использовать все элементы, предназначенные для дизайна WEB-страницы. Внутри начального элемента <body> могут располагаться атрибуты, обеспечивающие установки для всей страницы в целом. Синтаксис использования :

1. <body атрибут\_1=«значение1» атрибут\_2=«значение2»…>

**Что же необходимо для составления веб-страницы?**

**1. Нужно определиться с задачами и целями, которым будет служить сайт.**

Перед тем как взяться за непосредственное написание кода нужно определиться – зачем вообще нужен сайт, какие именно задачи он будет решать, какую пользу вы желаете извлечь из вашего ресурса сами и чем он будет полезен другим пользователям? Нужно отдавать себе отчет, что сайт - это продукт, развивающийся и улучшающийся со временем, поэтому перед его созданием нужно определиться в тех точках и аспектах, на которые будет сделан основной упор. От того, насколько будут проработаны основные моменты, будет зависеть насколько успешно станет развиваться сайт в дальнейшем.

**2. Полезно разработать схему будущего сайта.**

Не малой пользой при разработке индивидуального проекта будет служить создание схемы будущего сайт. Можно воспользоваться графическим редактором, для того, чтобы представить, как будет выглядеть планируемый сайт. В каких местах и какие именно блоки займут свое место. Какие сервисы понадобятся и на каких страницах будут расположены. Определиться с элементами, которые будут повторяться, а также с теми, которые станут изменяться динамически. Разделять основной блок принято на двенадцать колонок и все элементы располагать сообразно данным колонкам. Чтобы облегчить задачу, нужно изучить уже готовые сайты, проанализировать их структуру, обратить внимание на сильные и слабые моменты. Достоинства несомненно стоит реализовать на своем ресурсе. Тщательный анализ и подготовка перед программированием убережет от значительных переделок и сэкономит немало средств и времени.

**3. После составления схемы сайта можно приступить к воплощению его визуальной составляющей.**

Прежде чем нарисовать сайт рекомендуется изучить направления в современном веб-дизайне, благо на эту тему написано не мало статей. Начинающему веб-дизайнеру можно с уверенностью рекомендовать не изобретать велосипед заново и брать готовые, уже устоявшиеся решения у уже получивших значительный опыт мастеров. Рекомендуется не использовать большого обилия цветов, шрифтов, стилей оформлений изображений. В дизайне должно сохраняться единообразие.

**4. Разработчик создает сайт для людей.**

При написании своего ресурса в первую очередь необходимо думать о конечном пользователе, чтобы сайт был удобен в использовании и полезен. Не нужно забывать, что сайт должен быть настроен таким образом, чтобы посетители хотели возвращаться на него регулярно. Нужно использовать продуманную навигацию, как правило спокойные цвета в оформлении выглядят лучше, нежели яркие, броские, текст должен располагаться на светлом фоне и хорошо читаться. Не стоит использовать слишком вычурные шрифты.

Данные советы помогут организовать пространство вашего сайта таким образом, чтобы посетителю было комфортно пребывать на нем.

**5. Заранее подумайте о наполнении сайта.**

Помните, люди приходят в Интернет в поисках определенной информации, значит, у Вас на сайте, должно быть что-то интересное, причем интересное для посетителей сайта по выбранной Вами тематике. Подумайте, чем Вы будете наполнять свой сайт, какого рода Контент (содержание сайта) Вы будете размещать: текст, видео файлы, аудио файлы, софт и т.д. Подумайте, из каких источников Вы будете в дальнейшем черпать информацию для своего сайта.

Разбейте для себя Контент на первоочередной (то что Вы выложите сразу) и долгосрочный (чем будете наполнять сайт в дальнейшем).

**6. Подумайте, как Вы будете управлять сайтом в дальнейшем.**

По мере разрастания сайта перед Вами встанет вопрос обслуживания сайта, а именно: создавать новые страницы, редактировать или удалять существующие страницы, публиковать новости и т.д. Вы будете делать это сами или же поручите кому-то сопровождение Вашего сайта, но в таком случае будьте готовы к финансовым затратам, ведь никто не будет за спасибо постоянно обслуживать Ваш сайт.

Возможно, это будет просто небольшой сайт визитка (1-2 web-странички, может 10), не требующий постоянного сопровождения и обновления.

Перед запуском сайта в Интернете, обязательно проверьте работоспособность всех ссылок. Это важный момент.

**7. Протестируйте свой сайт сторонними посетителями.**

Что значит протестируйте? После размещения сайта в Интернете (хотя это можно сделать и локально на Вашем домашнем компьютере) и широкого рекламирования, попросите нескольких знакомых, побродить по сайту, посмотреть на структуру сайта, как у Вас все организовано и как все работает. И обязательно выясните мнение этих людей. Зачем? Помните Вы делаете сайт для людей и то что может быть понятно Вам как разработчику, может вызвать недоумение и как следствие отторжение других посетителей. Учтите мнение этих людей и обязательно внесите коррективы.

Всего 7 небольших пунктов, помогут Вам создать вполне достойный сайт. Все это не так сложно и трудно, как может показаться вначале, достаточно следовать этому простому плану. Конечно в каждом из пунктов есть свои нюансы, которые трудно постигнуть с ходу, но все приходит со временем и постепенно.

1. Инструменты для редактирования веб-страницы.

**Brackets**(Windows, OS X, Linux; brackets.io)

**Atom**(Windows, OS X, Linux; atom.io)

**jEdit**(Windows, OS X, Linux; jedit.org).

**EditPlus**(Windows, editplus.com)

**Visual Studio Code** (OS X, skedit.com)

**Sublime Text**(OS X, Windows, Linux; sublimetext.com)

**Dreamweaver**(OS X и Windows, adobe.com/ru/products/dreamweaver.html)

4. Основы HTML5: историческая справка, стандарты, структура html-документа, назначение

**HTML (HyperText Markup Language)** язык гипертекстовой разметки, который предназначен для создания веб-страниц и представляет собой набор правил.

**Основные версии языка:**

22 сентября 1995 г. версия HTML 2.0

14 января 1997 г. версия HTML 3.2

18 декабря 1997 г. версия HTML 4.0

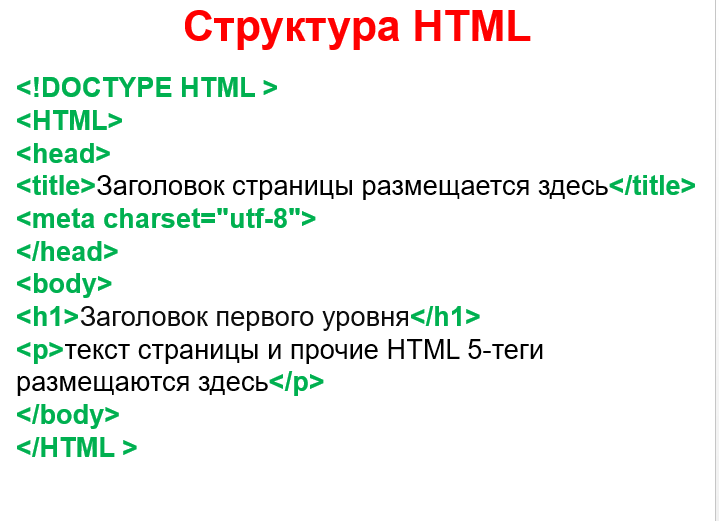
24 декабря 1999 г. версия HTML 4.01

15 мая 2000 г. версия ISO HTML

28 октября 2014 г. версия HTML5

1 ноября 2016 г. версия HTML 5.1



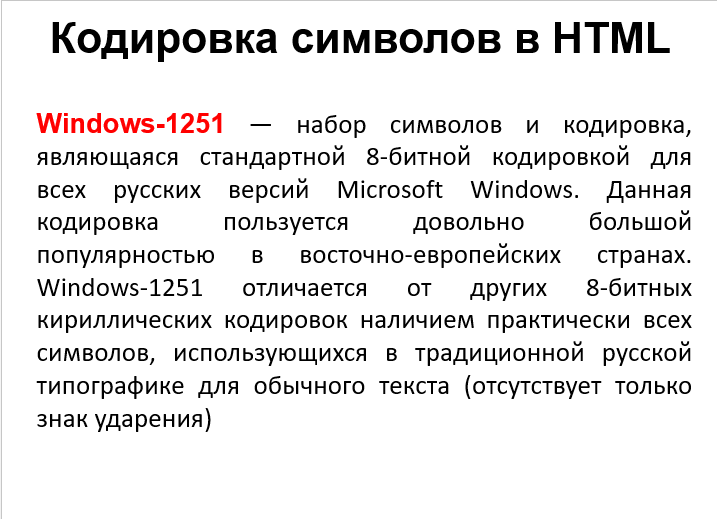


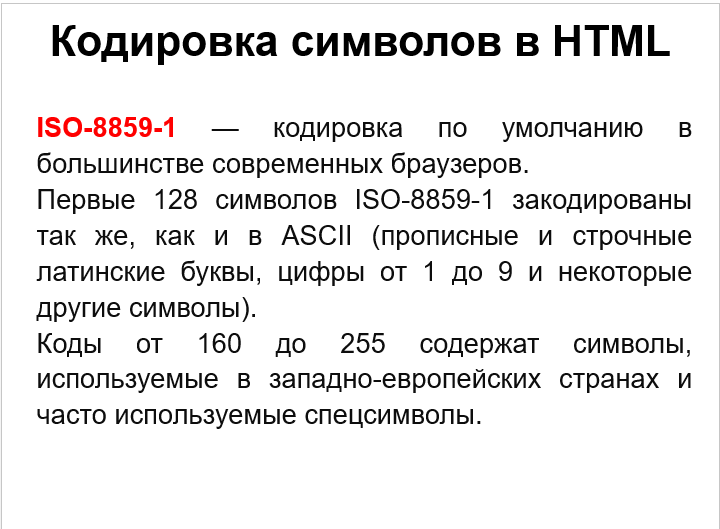


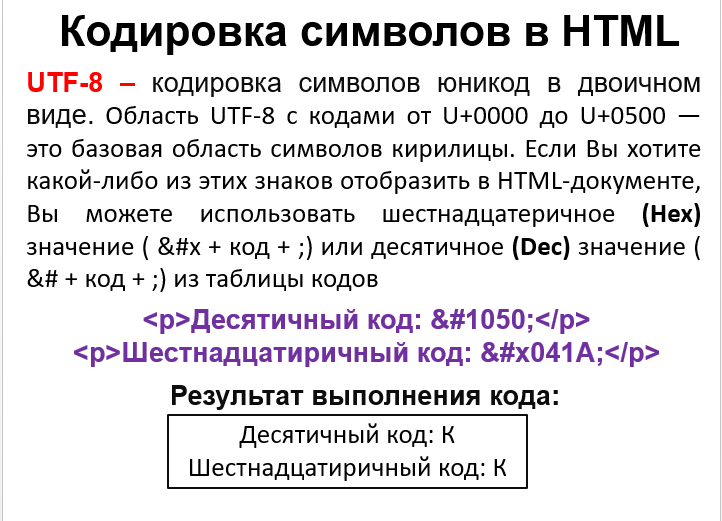
5. Кодировки (ASCII, Windows-1252, UTF-8, ISO-8859-1)

**Кодировка символов** международное представление букв, цифр и символов в веб-странице или любом другом файле, который хранится на компьютере и может быть передан через Интернет.









1. Служебные и структурные теги.

К **служебным** тегам относят:

* html;
* head;
* title;
* meta;
* style;
* body;
* script.

7. Семантические теги.

***Семантика*** – это изучение значений слов и фраз на языке.

Семантические элементы = элементы с смыслом

**Семантические теги** — это теги, которые предназначены для того чтобы компьютерные программы (поисковые системы, сборщики информации, речевые браузеры и т.д.), понимали какой тип информации заложен в данных тегах.

Примеры семантических тегов:

*header* устанавливает верхний колонтитул («шапку») для страницы или раздела

*nav* предназначен для создания блока навигации веб-страницы или всего веб-сайта, при этом не обязательно должен находиться внутри

*footer* устанавливает нижний колонтитул («шапку») для страницы или раздела

*section* представляет собой общий раздел документа или приложения. Раздел в этом контексте представляет собой тематическую группу контента, обычно с заголовком

*article* представляет собой полную или автономную композицию в документе, странице, приложении или сайте и, в принципе, может независимо распространяться или повторно использоваться, например в синдикации. Это может быть сообщение на форуме, статья в журнале или газете, запись в блоге, комментарий пользователя, интерактивный виджет или гаджет или любой другой независимый элемент контента.

*main* определяет уникальный основной контент для веб-страницы, в нем не должно быть элементов, располагаемых на всех страницах сайта, например, заголовка, нижнего колонтитула и основной навигационной панели

*aside* предназначен для оформления боковых колонок, рекламных блоков, группирования навигационных элементов и прочего содержимого, которое лишь косвенно касается основного контента страницы.

*details* используется для информации, которую можно скрыть или показать по желанию пользователя

*summary* определяет видимый заголовок для элемента

*figure*заключает в себе потоковый контент, который может сопровождаться заглавием <figcaption>. Предназначен для контента, который можно удалить со страницы не изменяя смысла остальной информации.

*mark* представляет фрагмент текста в документе, помеченный цветом или выделенный подсветкой для справочных целей

*time*используется для обозначения конкретной даты и времени

*address* контактная информация элемента <article> или <body>

8. Теги формы.

9. Теги списков, таблиц и изображений.

10. Теги для встраиваемого контента.

11. Теги для отображения и форматирования текста

12. Атрибуты HTML5. Специальные символы.

13. Глобальные атрибуты

14. Атрибуты обработчиков событий.

15. Атрибуты тегов формы

16. Основы CSS3

17. Создание стилей. Классификация селекторов.

18. Псевдоклассы и псевдоэлементы

19. Механизм наследования стилей.

20. Единицы измерения значений свойств CSS

21. Каскадность стилей. Приоритет стилей

22. Конфликты стилей. Устранение конфликтов стилей.

23. Использование селекторов потомков

24. Форматирование текста на CSS3: абзацев, символов и слов

25. Использование веб-шрифтов. Использование службы Google Fonts.

26. Понятие блочной модели.

27. Форматирование таблиц

28. Форматирование веб-форм

29. Добавление фоновых изображений. Использование градиентных фонов.

30. Построение навигационной системы сайта.

31. Форматирование ссылок

32. Использование ролловеров. Метод CSS-спрайт

33. CSS-переходы.

34. CSS-преобразования.

35. Функции 3D-преобразований.

36. CSS-анимация.

37. Форматирование списков перечисления.

38. Системы модульной верстки.

39. Система модульной верстки Skeleton.

40. Структурирование HTML-кода под модульную сетку и ее использование.

41. Основы Grid-верстки.

42. Медиазапросы. Создание адаптивного дизайна

43. Основы Flexbox-верстки.

44. Основы Sass

45. Основы Sass. Наследование свойств

46. Установка Sass

47. Примеси Sass

48. Переменные Sass

49. Операторы Sass

50. Основы Sass. Вложенные правила

51. Основы XML

XML-документ состоит из элементов, атрибутов, сущностей и инструкций обработки.

Первые строки до корневого элемента называются *прологом документа* и сообщает о том, что документ XML и какой версии.

Первая строка содержит объявление XML.

Существуют три атрибута, значения которых могут быть установлены в этом объявлении:

version

encoding

standalone

Вторая строка содержит объявление типа документа XML.

Например

<!DOCTYPE корневой элемент [

внутренне подмножество

]>

или

<!DOCTYPE note SYSTEM "note.dtd">

52. Правила документов XML.

53. Элементы и атрибуты XML

XML элементы должны следовать следующим *правилам написания имен*:

1. Имена могут содержать буквы, числа и другие символы
2. Имена не могут начинаться с цифры или символа пунктуации
3. Имена не могут начинаться с сочетания "xml" (или XML, или Xml и т.п.)
4. Имена не могут содержать пробельные символы

XML атрибуты могут иметь уникальные имена. Некоторые имена атрибутов зарезервированы рабочей группой XML для специальных целей использования в XML, и начинаются с префикса **xml:**

**xml:lang  —** классифицирует элемент по языку, на котором написано содержимое элемента

**xml:space  —** определяет должны ли сохраняться в содержимом пробелы

**xml:link  —** сообщает,что элемент является элементом ссылки

**xml:attribute  —** позволяет «переназначить» эти специальные атрибуты

54. Стандарты XML

Стандарты XML — это набор расширений, которые придают xml-файлам дополнительные возможности

Основные расширения XML:

1. XSL

2. XML Shema

3. RelaxNG

4. Schemarton

5. SAX

6. XSLT

7. Xpath

8. SVG

9. Xlinks

10. Xpointer

11. AJAX

55. Cхема DTD

**Document Type Definition (DTD)**  — это язык описания структуры XML-документа, который используется для проверки грамматики XML-документа и его соответствия определенному

DTD описывает:

• Какие элементы могут присутствовать в документе;

• Вхождение элементов (повторения и т.п.);

• Возможные атрибуты элементов;

• Обязательные / необязательные атрибуты;

• PCDATA и CDATA;

• Применяемые в документе сущности.

56. XML Schema

**XML-схема** - это основанная на XML современная альтернатива DTD, описывающая структуру XML-документа, в том числе:

• Определяет элементы, которые могут появляться в документе;

• Определяет атрибуты, которые могут появляться в документе;

• Определяет, какие элементы являются дочерними;

• Определяет последовательность, в которой появляются дочерние элементы;

• Определяет число дочерних элементов;

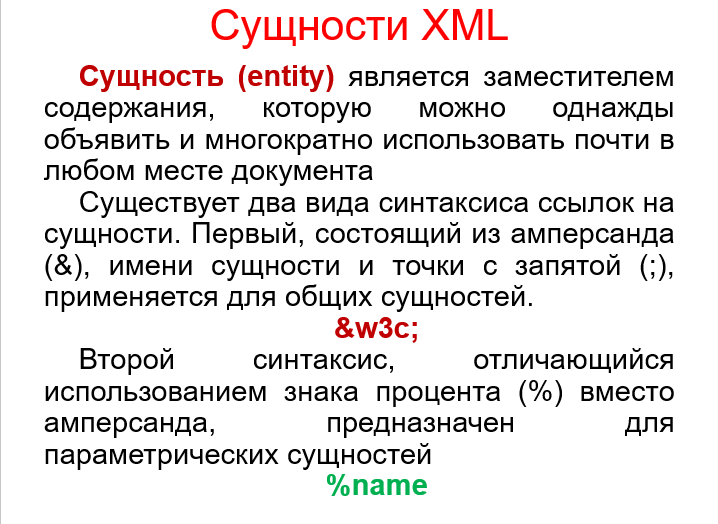
• Определяет пустой ли элемент или он может включать в себя текст;

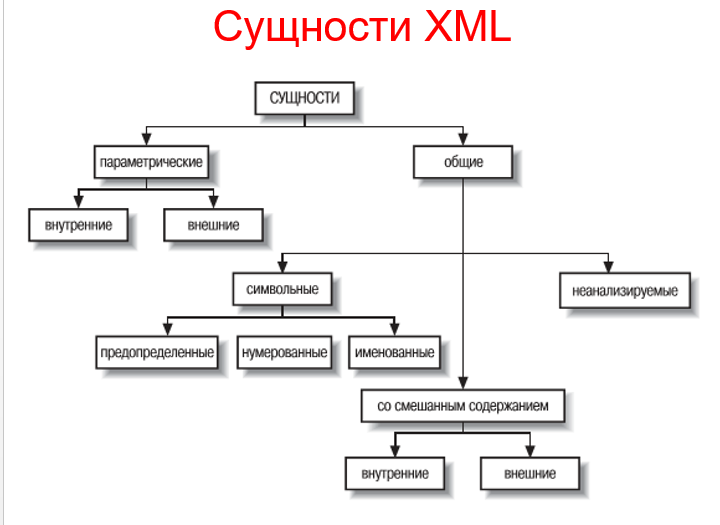
• Определяет типы данных элементов и атрибутов;

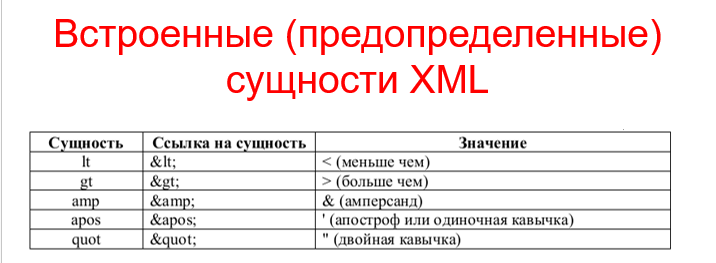
• Определяет значения атрибутов по умолчанию.

**Преимущества XML Schema:**

* XML схемы легко расширяются
* XML схемы пишутся на XML
* XML схемы поддерживают типы данных
* XML схемы поддерживают пространства имен

57. Сущности XML





58. Язык преобразований XSLT

59. Сортировка данных в XML

60. Язык переадресации XPath

61. Язык запросов XQuery

62. Введение в XLinks. Применение XPointer

63. Основы SVG

64. Атрибуты SVG

65. Анимация SVG

66. Основные элементы SVG

67. Работа с текстом SVG

68. Обзор и версии языков разметки веб-страниц

69. Понятие DOM. Основы JavaScript

70. Способы доступа к элементам DOM

71. Работа с формами с помощью JavaScript

72. Управление событиями на Javascript

Старший преподаватель Е.В. Барковский