**HTML**

**html\_1. HTML5 теги, как сделать верстку семантичной и зачем это нужно?**

Семантика языка — это смысловое значение слов. В программировании — начальное смысловое значение операторов, основных конструкций языка и т. п.

Чтобы сделать верстку семантичной, нужно отказаться от ‘дивной’ верстки, и использовать вместо див и спан HTML-теги, передающих смысл собственного содержимого, например, заголовки, навигация, списки и т.д.

Семантика HTML-элементов передаётся через их роли, состояния и свойства. Роли описывают основной смысл элемента, а состояния и свойства добавляют больше деталей: выбран или не выбран чекбокс, активно или неактивно поле в форме и так далее. Теги и их встроенные роли перечислены в спецификации ARIA в HTML (ARIA in HTML). Все HTML-теги, которых больше 100, семантические.

Зачем нужна семантика?

1.Семантический код легче читать и поддерживать, в нем проще ориентироваться и другим, т.к. сразу видно логику отдельных элементов и общую структуру интерфейса.

2. Когда верстка семантическая, нужно писать меньше кода, а чем меньше кода – тем быстрее грузится страница. Например, у <button> и <input> уже встроены обработчики событий: их не нужно писать руками. Чтобы воспроизвести нативное поведение кнопки, нужно слушать события keydown при нажатии на Enter и keyup для пробела и Enter. Размеры файлов со стилями и скриптами напрямую влияют на *производительность сайта (perfomance)*. Чем больше вес файлов, тем дольше грузится страница.

3. Семантические теги повышают доступность для пользователей с инвалидностью. Они используют скринридеры, экранные лупы и другие вспомогательные технологии для взаимодействия со страницами.

К примеру, скринридер расскажет пользователям, что <button> — это кнопка. Пользователи голосового управления могут кликнуть по элементу <button> с помощью команды «кнопка, клик». У чекбоксов есть состояния checked и unchecked. Благодаря им люди могут услышать, какой чекбокс выбран, а какой нет. В <h1>−<h6> встроено свойство для уровня заголовка — level. Скринридеры зачитывают уровень заголовка вместе с ролью: заголовок первого уровня или заголовок четвёртого уровня. Так структуру страницы смогут представить те, кто слушает интерфейсы.

Другой плюс продуманной HTML-разметки — навигация для скринридеров. Теги вроде <header> и <footer> — ориентиры. Это значит, что пользователи могут перемещаться по отдельным блокам страницы с помощью горячих клавиш.

Пользователи скринридеров могут перемещаются и по заголовкам.

4. Сами браузеры, различные расширения и другие технологии тоже используют семантику.

Например во все браузеры встроен *режим чтения (reading mode)*. В нём со страницы удаляются все лишние элементы и остаётся самое важное: заголовки, текст, картинки и видео. Если страница, например, свёрстана без <h1>, <h2> и заголовков других уровней, пользователи получат большой кусок неструктурированного текста.

Часто различные браузерные расширения, которые изменяют стиль страниц, могу основываться на разметке.

Умные часы и голосовые используют голосовые команды, которые лучше работают с семантическими тегами.

5. Семантика позитивно влияет на поисковую оптимизацию сайтов — SEO (search engine optimization)

**html\_2. Что такое doctype и зачем он нужен?**

<!DOCTYPE> или «доктайп» — это сокращение от document type - «тип документа» — это декларация типа документа. Сегодня доктайп по HTML спецификации нужен для того, чтобы браузер работал в стандартном режиме (standards mode). Сейчас единственный стандартный доктайп <!DOCTYPE html>, все остальные устарели.

Если не написать doctype браузеры переключаются в режим обратной совместимости со старыми версиями языка (quirks mode).

**html\_3. Какие есть отличия у тегов div, p, span, a?**

- роли - блочность\строчность – атрибуты - категория контента – модель контента

**html\_4. Что такое инлайновые элементы (например b, strong, em)?**

Это элементы, которые занимают часть строки, и свойство display установлено как inline.

Например, ссылки нужны для оборачивания слов и фраз. Значит, они должны быть строчными, чтобы не разорвать окружающий текст.

Элементы, для которых это значение задано по умолчанию, — <span>, <a>, <q>, <code> и др., в основном они используются для изменения вида текста или его смыслового выделения.

Особенности строчных элементов:

- не применяются свойства, связанные с размерами (width, height). Их размеры определяет содержимое.

- применяются только горизонтальные margin, padding применится и горизонтальный, и вертикальный, но отобразится странно.

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

**html\_5. Как создавать таблицы в html? Для чего вообще нужны таблицы?**

Таблицы предназначены для структурирования данных. Часто в таблицах размещают однотипные данные. Для создания таблиц существует набор семантических тегов:

<table> - определяет начало и конец таблицы. Всё содержимое таблицы должно находиться между <table></table>. Встречая этот тег в разметке, браузер понимает, что дальше будет таблица.

<thead> - отвечает за шапку таблицы. Внутри этого тега могут располагаться одна или более строк с заголовками таблицы. <thead> должен располагаться в разметке сразу за открывающим <table> или после <caption>, но строго до <tbody> и <tfoot>.

<tbody> — определяет тело таблицы. Можно использовать несколько <tbody> внутри таблицы, разделяя тем самым данные на отдельные блоки. Этот тег схож по семантике с <main>, но в пределах таблицы. Если этот тег не используется - браузер самостоятельно добавит <tbody> при отрисовке таблицы.

<tfoot> —определяет некую строку с итогом данных таблицы. В таблице может быть только один блок <tfoot>. Браузер всегда отрисовывает <tfoot> внизу таблицы, даже если этот блок идёт в разметке не последним. Если по какой-то причине не используется <thead> или <tbody>, то футер будет отрисован после всех <tr>. По семантике этот тег похож на <footer>, только в пределах таблицы.

<th> — определяет заголовок столбца или строки таблицы. «th» расшифровывается как «table header» и переводится «заголовок таблицы». Как «td» только для заголовка.

<tr> — определяет строку таблицы. «tr» расшифровывается как «table row» и переводится «ряд таблицы».

<td> — определяет ячейку таблицы. Используется внутри тега<tr>. Ячейки формируют из себя столбцы. В HTML нет специального тега для столбцов. «td» расшифровывается как «table data» и переводится «данные таблицы».

Ну или можно создать таблицу из любых тегов при помощи css свойств display.

**html\_5.1 border-spacing, border-collapse**

CSS свойство border-collapse определяет, будут ли ячейки внутри иметь общие или отдельные границы. По умолчанию, границы в таблице отображаются отдельно друг от друга. Это значит, что если у неё или её ячеек есть границы, то между ними будет пространство (размерами которого можно управлять с помощью border-spacing). Если задать свойству border-collapse значение collapse, то все смежные границы будут отображаться как одна.

CSS свойство border-spacing задаёт расстояние между ячейками таблицы. Это свойство работает только для таблиц со свойством border-collapse в значении separate.

Отступ также будет появляется вокруг таблицы, поэтому расстояние между ячейками таблицы и её рамкой будет складываться из значений border-spacing и padding таблицы.

Разрешается указывать одно или два неотрицательных значения:

* border-spacing: 1px — расстояние между ячейками со всех сторон одинаковое.
* border-spacing: 1px 2px — первое значение задаёт расстояние между ячейками по горизонтали, второе — по вертикали.

**html\_5.2 colspan, rowspan**

Оба атрибута предназначены для объединения ячеек. colspan нужен для объединения ячеек из 2 или более столбцов, а rowspan для объединения ячеек из 2 или более рядов. <td rowspan="3">…</td>

**html\_5.3 table-layout**

CSS свойство, которое применяется к <table> и определяет, как браузер должен вычислять ширину ячеек таблицы. Оно может принимать следующие значения: auto, fixed или inherit.

Auto - значение по умолчанию. Ширина столбца задается самым широким неразрывным содержимым ячейки. Данный алгоритм может быть медленным, так как браузер должен прочитать все содержимое в таблице, прежде чем определить её окончательный макет. По умолчанию в браузерах для отображения таблицы используется стандартное поведение: браузер рассчитывает ширину строки как сумму ширины всех ячеек. Следом, по тому же принципу рассчитывает ширину 2-й строки, и если размеры какой-либо ячейки превышают размеры вышестоящей ячейки, перестраивает всю таблицу. Далее определяет ширину 3-й ячейки, 4-й и т.д. В случае если таблица имеет много, очень много рядов, расчет ширины может немного затянуться…

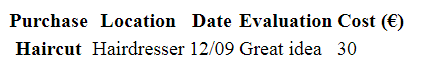
Fixed - горизонтальная разметка таблицы будет зависеть от ее ширины и ширины колонок, а не от контента ячеек. Ширина столбцов определяется шириной их заголовков. в случае применения этого режима, расчет ширины столбцов идет по первой строчке и все последующие используют это значение. Это может ускорить время отрисовки по сравнению с auto.

**html\_5.4 caption**

Тег используется для добавления подписи к таблице. В нем лучше всего написать обобщающую информацию. Тег должен идти сразу после открывающего тега <table>. По умолчанию подпись визуально располагается сразу перед таблицей. Но её положением можно управлять свойством caption-side. Вне зависимости от визуального расположения подписи скринридер прочитает её перед таблицей. Наличие подписи таблицы важно для людей, использующих скринридеры. Вместо того чтобы слушать табличные данные, которые могут быть ненужны удобнее прослушать подпись к таблице и решить, важна ли она. Внутри тега можно располагать любые другие теги, нужные для группировки и оформления подписи.

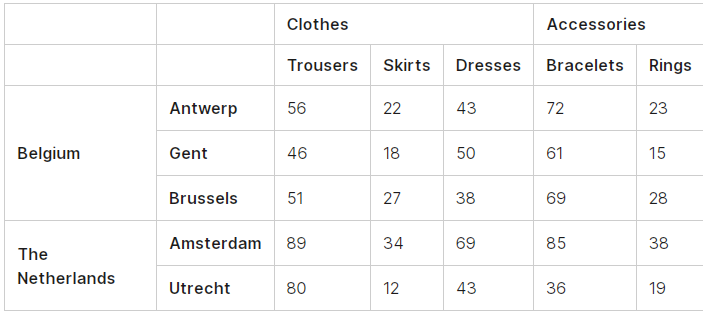
**html\_5.5 scope**

Атрибут, который может быть добавлен к элементу <th> он сообщает скринридеру какие ячейки точно являются заголовками. Для заголовков столбцов используется значение “col”, для заголовков строк - ”row”. Скринридер распознает разметку, структурированную таким образом, что позволяют пользователям прочитать весь столбец или строку целиком.



Атрибут scope имеет ещё два возможных значения — colgroup и rowgroup. Эти значения указывают, что заголовок <th> относится не к одному столбцу или строке, а к группе столбцов. То есть заголовок описывает не отдельную строку, а целую группу строк.

Они используются для заголовков, которые располагаются вверху столбцов или строк. Например, ячейка с "Clothes" находится над ячейками "Trousers", "Skirts" и "Dresses". "Clothes" должна получить атрибут scope="colgroup", тогда как другие получат атрибут scope="col".

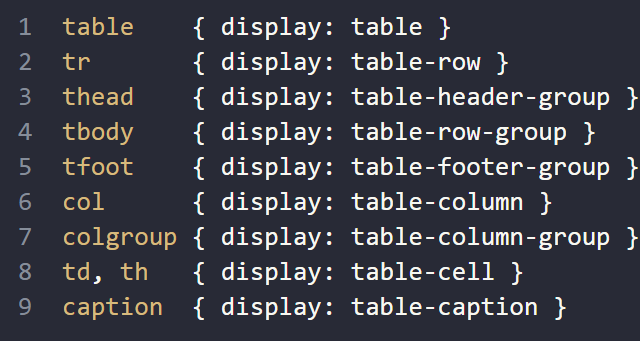
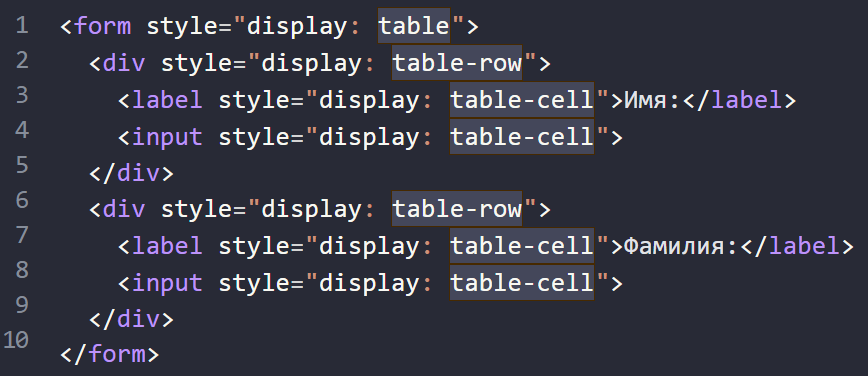


**html\_5.6 display: table, table-cell, table-row**

Вообще можно создать таблицу любыми элементами, если поставить им соответствующие значения display. Для таблицы целиком table, для строки – table-row, для ячейки – table-cell и т.д. Это будет полноценная таблица. Используются табличные алгоритмы вычисления ширины и высоты элемента, описанные в стандарте.

Это хорошо для семантической вёрстки и позволяет избавиться от лишних тегов.

С точки зрения современного CSS, обычные <table>, <tr>, <td> и т.д. – это просто элементы с предопределёнными значениями display:



**html\_6. В чем заключается отличие чекбокса от радиокнопок? Как связать радиокнопки в группу?**

C помощью радиокнопок можно выбрать только один элемент, а с помощью чекбоксов – любое количество, также они имеют разную формую.

Группа радиокнопок определяется путём присвоения каждой радиокнопке в данной группе одного и того же значения атрибута name. Выбор любой радиокнопки в этой группе автоматически отменяет выбор другой радиокнопки в той же группе.

**html\_7. Что такое тег <label> и как с ним работать?**

Элемент <label> используется для создания подписи к элементу формы, например к input. Помимо текстовой подписи создаётся программная связь между подписью и элементом формы. Это сильно облегчает взаимодействие с формами пользователям, использующим скринридеры. Когда фокус попадает на элемент формы, с которым связан <label>, скринридер автоматически зачитывает текст подписи, и пользователь понимает, какие данные нужно ввести или какие данные представлены в текущем элементе формы.

Связать <label> с элементом формы можно 2мя способами:

* Задать элементу формы атрибут id, и такое же значение задать атрибуту for тега <label>.
* Обернуть элемент формы в тег <label>. В этом случае связь создаётся автоматически и нет необходимости в атрибутах id и for.

for — значение этого атрибута должно соответствовать значению атрибута id связываемого элемента. Первый же элемент в документе, чей id будет совпадать со значением атрибута for, становится связанным с нашим <label>. Единственное условие — элемент должен принадлежать к группе связываемых элементов: <button>, <input>, <meter>, <output>, <progress>, <select> и <textarea>.

Если элемент с нужным id не является связываемым, то связь не создаётся, и даже если дальше по документу найдётся связываемый элемент с таким же id, то он уже не будет учитываться.

По умолчанию <label> является строчным элементом.

Один элемент формы может быть связан с несколькими <label>, но один <label> может быть связан ровно с одним элементом формы.

**html\_8. Что такое БЭМ и в чем его польза?**

БЭМ — методология Яндекса в основе которой лежит принцип разделения интерфейса на независимые блоки.

Польза БЭМ:

* Код становится читабельнее. Сразу видно из каких блоков состоит страница, какие элементы имеются в каждом блоке, и какие модификаторы в них применены.
* Удобно работать одновременно над разными частями интерфейса, т.к. блоки не зависят друг от друга, и ничего не сломается.
* Также благодаря независимости блоков предотвращается взаимное влияние компонентов друг на друга.
* Код можно переиспользовать.

**html\_9. Что такое блок, когда следует создавать?**

Блок — это логически и функционально независимый компонент страницы.

Делать блок следует, если фрагмент кода может использоваться повторно и не зависит от реализации других компонентов страницы.

**html\_9.1 Может ли блок содержать другие блоки?**

Блоки можно вкладывать в любые другие блоки.

**html\_9.2 Почему не стоит задавать внешнюю геометрию блокам?**

Блоки в БЭМ должны быть максимально независимыми и универсальными. Если блок содержит внешние отступы или позиционирование, его использование в разных контекстах становится сложнее, так как внешний вид блока будет зависеть от конкретного расположения.

Блок не должен влиять на свое окружение, т. е. блоку не следует задавать внешнюю геометрию (в виде отступов, границ, влияющих на размеры) и позиционирование. Таким образом обеспечивается независимость, при которой возможно повторное использование или перенос блоков с места на место.

**html\_10. Что такое элемент, когда следует создавать?**

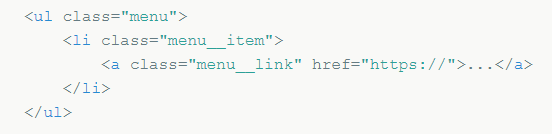
Элемент — это составная часть блока, которая не может использоваться в отрыве от него.

Делать элемент следует, если фрагмент кода не может использоваться самостоятельно, без родительской сущности (блока).

**html\_10.1 Может ли элемент содержать другие элементы?**

Элементы можно вкладывать друг в друга. Допустима любая вложенность элементов.

В примере ниже пункты меню представлены ссылками. Такая структура блока реализуется за счет вложенности элементов:



**html\_10.2 Может ли элемент содержать блоки (блок, к которому он относится, или другие блоки)?**

Может.

**html\_10.3 Можно ли создавать элемент элемента?**

Нельзя. Один из принципов работы с элементами – это вложенность. Элемент — всегда часть блока, а не другого элемента. Наличие элементов элементов ограничивает возможность изменять внутреннюю структуру блока. Элементы нельзя поменять местами, удалить или добавить без корректировки существующего кода.

**html\_10.4 Может ли элемент использоваться вне блока?**

Один из принципов работы с элементами – это принадлежность. Элемент — всегда часть блока и не должен использоваться отдельно от него. Элементы не существуют вне блока. Каждый элемент может принадлежать только одному блоку.

**html\_10.5 Может ли существовать блок без элементов?**

Один из принципов работы с элементами – это необязательность. Элемент — необязательный компонент блока. Не у всех блоков должны быть элементы.

**html\_11.1 Модификатор. Что такое, когда следует использовать?**

БЭМ-сущность, определяющая внешний вид, состояние или поведение блока либо элемента.

**html\_11.2 Когда следует использовать булевый модификатор, а когда модификатор ключ-значение?**

Булевый использовать, когда важно только наличие или отсутствие модификатора, а его значение несущественно. Например, «отключен»: disabled. Считается, что при наличии булевого модификатора у сущности его значение равно true.

Ключ-значение использовать, когда важно значение модификатора. Например, «меню с темой оформления islands»: menu\_theme\_islands.

**html\_11.3 Является ли корректным следующий код:**

*<div class="button\_red"> Click me </div>*

Нет, т.к. модификатор нельзя использовать самостоятельно. С точки зрения БЭМ-методологии модификатор не может использоваться в отрыве от модифицируемого блока или элемента. Модификатор должен изменять вид, поведение или состояние сущности, а не заменять ее.

**html\_12 Почему в CSS по БЭМ не рекомендуется использовать селекторы по id и по тегам?**

Использованием классов обеспечивается независимость, при которой возможно повторное использование. Объединение тега и класса повышает специфичность CSS-правил, что усложняет задачу их переопределения. Это приводит к войнам значимости, при которых таблицы стилей загружаются излишне сложными селекторами.

**html\_14 Какие альтернативы есть БЭМ и в чем сила/слабость каждого? Знать хотя бы 2 альтернативы данному подходу, плюсы/минусы подходов.**

Все подходы к организации CSS можно разделить на две условные группы:

* Чистота CSS при усложнённом HTML
* Чистый HTML при усложнённом CSS

1. OOCSS — Object Oriented CSS или объектно-ориентированный CSS.

Основные принципы:

* Разделение структуры и внешнего вида.
* Повторное использование классов.

В традиционном CSS структура и внешний вид часто смешиваются в одном классе. В OOCSS они разделяются - структура может быть описана в одном классе, а внешний вид — в другом. К структуре относится всё, что влияет на расположение контента – высоту, ширину, внутренние и внешние отступы. К стилям оформления относятся – фон, шрифты и др.

Плюсы:

- Уменьшение количества кода за счет повторного его использования (принцип DRY).

- Код становится более логичным и простым для понимания за счет разделение структуры и внешнего вида

- масштабируемость - компоненты легко адаптируются под новые требования. Например, базовые классы кнопок могут быть использованы для разных тем или размеров, без необходимости переписывать код.

Минусы:

- HTML может быть перегружен из-за необходимости написания множества классов.

- Необходимость строгой дисциплины. Нарушение принципов приводит к утрате его преимуществ.

- Сложность внедрения в готовые проекты. Переход с традиционного CSS на OOCSS требует времени и усилий.

1. Atomic CSS — это методология верстки, в которой используются маленькие атомарные CSS-правила. Часто они применяет одно CSS-свойство (например, меняет цвет текста), но не обязательно одно. Основные принципы:

* **Переиспользуемость:** Классы могут повторно использоваться в разных частях сайта, что снижает дублирование кода.
* **Малые единицы стилей:** Каждый класс отвечает за одно свойство (например, bg-red для красного фона).
* **Комбинируемость:** Стили комбинируются через использование нескольких классов на одном элементе.

Плюсы:

- Не нужно думать про уникальные названия сущностей, БЭМ-блок это или БЭМ-элемент, какую делать структуру каталогов и etc

- Меньше CSS. С определенного момента разработки, стили перестают добавляться. Переиспользуются те, которые уже есть.

- Стили пишутся быстрее. Особенно, если используются короткие названия. Плюс - меньше нужно переключаться между файлами.

Минусы:

- HTML может быть перегружен из-за необходимости написания множества классов.

- наименования классов представляют собой описательные названия свойств, не описывая семантическую сущность элемента.

1. **html\_15 Что такое миксы в БЭМе и какие особенности работы с ними есть?**

Способ использования разных БЭМ-сущностей на одном DOM-узле.

Миксы позволяют:

* совмещать поведение и стили нескольких БЭМ-сущностей без дублирования кода;
* создавать семантически новые компоненты интерфейса на основе имеющихся БЭМ-сущностей.

Микс используется:

- чтобы позиционировать один блок в составе другого блока;

- чтобы одинаково стилизовать несколько разных блоков на странице (допустим тексту внутри блоков article и copyright необходимо задать один и тот же цвет и шрифт. Для этого нужно примиксовать блок text, стили которого определяют цвет и шрифт текста, к блокам article и copyright.)

**html\_17. Зачем нужны метатеги?**

Метаданные — это, в самом простом определении, данные, что описывают данные.

HTML-документ — это данные, и он может содержать метаданные в элементе <head>, в метатегах.

В теге <meta> хранится краткое описание страницы, ключевые слова и другие данные, которые могут понадобиться браузерам и поисковым системам.  Информация в метатегах называется метаданными. Пользователь не видит их содержимое на странице.

Метатеги различаются набором атрибутов и их значений. Один тег содержит одно сообщение: например, описание страницы <meta name="description">, ключевые слова <meta name="keywords" content="...">, кодировку страницы <meta charset="UTF-8"> и др.

<meta name="viewport"> - говорит браузеру, как масштабировать и отображать страницу на разных устройствах, особенно на смартфонах и планшетах. Если не использовать viewport, то сайт может выглядеть как уменьшенная версия десктопного сайта на мобильном экране, что делает его трудночитаемым и неудобным для использования. У viewport пять основных параметров — width, initial-scale, maximum-scale, minimum-scale, user-scalable. Параметр width определяет ширину области просмотра. Чаще всего используется значение device-width, что означает «использовать ширину экрана текущего устройства».

<meta name="viewport" content="width=device-width">

Параметр initial-scale задаёт начальный масштаб страницы. Значение 1 означает, что страница будет отображена без масштабирования. Записывается так: initial-scale=1. Параметры maximum-scale и minimum-scale определяют, как сильно пользователь может увеличивать или уменьшать масштаб. Параметр user-scalable имеет значения yes или no и определяет, может ли пользователь масштабировать страницу.

Самый популярный meta viewport: <meta name="viewport" content="width=device-width, initial-scale=1" />

Рекомендуется вставить как минимум метатеги description, viewport, charset. Они помогут оптимизировать сайт для браузера пользователя и поисковых систем.

**html\_18. Что такое data-атрибуты, и привести примеры, когда они полезны?**

Data-атрибут – это любой атрибут, чьё имя начинается с data-. Они позволяют хранить дополнительную информацию в стандартных элементах HTML, и потом с этой инфой можно работать из JS и CSS.

Они полезны, чтобы помечать HTML-элементы для JavaScript или CSS, для передачи пользовательских данных из HTML в JavaScript или CSS. Дата-атрибуты можно использовать в качестве селекторов по атрибуту – например можно выбрать только те элементы у которых есть атрибут [data-length] {}, или элементы, у которых атрибут имеет заданное значение: [data-age="46"] {}.

**html\_19. Когда стоит использовать тег img, а когда background-image?**

Тег img для картинок несущих смысловую нагрузку или каким-либо образом относящиеся к содержанию страницы (картинки в тексте, баннеры, картинки слайдера, аватарки и тд). Свойство background-image для элементов оформления: различного рода разделители, маркеры, фоны и тд.

При печати страницы сайта теги img печатаются, а background-ы нет.

Все, что желательно к индексации поисковиками выводим картинками, все что хотим скрыть- фонами.

Браузеры разработанные для людей с ограниченными возможностями читают альты картинок, потому если изображение несет важный смысл, оно обязательно должно быть оформлено html тегом.

Некоторые старые браузеры не поддерживают свойство background-size и для создания масштабируемого фона в них используется тег, а не свойство фона.

Img с анимацией меньше нагружает браузеры.

*Для замены текста изображением лучше использовать background-image (к примеру, если нужно вывести заголовки красивыми картинками).*

**html\_20. Как стилизовать svg? Какие основные атрибуты и CSS-свойства используются для этого?**

Стилизовать можно через HTML, CSS.

Fill – заливка элемента. Возможные значения: none, ключевые слова, цвета, паттерны и градиенты. Значение по умолчанию — black.

fill-rule - свойство определяет, как будут заливаться сложные фигуры, имеющие пересечения внутри себя. Для простых фигур ни на что не влияет. Возможные значения: nonzero, evenodd

fill-opacity - управляет прозрачностью заливки. Можно задавать значения от 0.0 до 1.0 или в процентах.

stroke - цвет обводки. Значения по умолчанию нет.

stroke-width - толщина обводки, можно задавать в единицах длины или в процентах. Значение по умолчанию: 1.

stroke-linecap - свойство определяет, как будут выглядеть концы линий. Возможные значения: butt, round, square. Значение по умолчанию: butt.

stroke-linejoin - определяет как будут выглядеть соединения линий на углах. Возможные значения: miter, round, bevel. Значение по умолчанию: miter.

stroke-dasharray - управляет видом пунктирной обводки. Можно задавать в единицах длины или в процентах. Если задано одно значение, второе значение считается равным первому. Например, stroke-dasharray="1" нарисует линию из отрезков длиной одну единицу разделенных пробелами такой же ширины.

stroke-dashoffset - свойство позволяет задать смещение пунктирной обводки относительно первоначального положения. Значение задается в единицах длины или в процентах, значения могут быть отрицательными. Значение по умолчанию: 0.

stroke-opacity - прозрачность обводки. Задаются значения от 0.0 до 1.0.

**html\_21. Что такое специальные html-сущности (неразрывный пробел, тире, которое не дефис) и как с ними работать?**

HTML-сущности — это конструкции - части текста ("строки"), которые начинаются с символа амперсанда (&) и заканчиваются точкой с запятой (;). Это коды для отображения специальных символов: знаков авторского права, дефисов, длинных тире, кавычек и т. д., а также для отображения зарезервированных символов в HTML. Если, например, использовать знаки меньше (<) или больше (>) в тексте, браузер может смешать их с тегами.