Неделя 4. Стили. CSS.

1. Какими способами можно подключать CSS-стили? Найти еще один способ, не озвученный в уроке.

В уроке мы изучили подключение стилей следующими способами:

- Через атрибут style
- Встроить стили глобально через тег style
- Подключить внешние таблицы стилей через тег link

  Новый способ:
- Импортировать стили из внешних файлов:
  @import "style/header.css"
  @import "style/footer.css"

2. Зачем нужен Normalize.css?

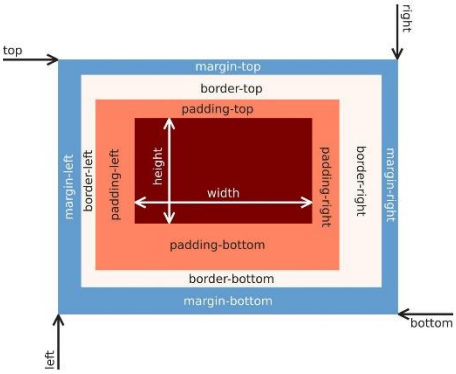
Normalize.css обеспечивает способность сайта отображаться и функционировать во всех часто используемых браузерах идентично, в стилях по умолчанию.
Когда элемент имеет различные стили по умолчанию в разных браузерах, normalize.css там, где это возможно, стремится сделать эти стили совместимыми и соответствующими современными стандартам. Кроме незаметных улучшений, Normalize.css может корректировать ошибки и основные несоответствия браузера.

3. Что такое CSS-директивы?

Есть CSS-правило (rule sets), которое состоит из селектора и перечня свойств и их значений, а есть так называемые «эт-правила» (at-rules), которые называют CSS-директивами - они показывают CSS как себя вести. Начинаются с символа @ и сопровождаются идентификатором. Например, @font-face, или вышеупомянутое импортирование стилей: @import - сообщает CSS включить внешнюю таблицу стилей.

4. В чем разница между margin и padding?

Margin, padding (еще есть border) – это части так называемой Блочной модели. Механизм Блочной модели следующий: Посередине есть зона контента, которую окружает padding, окруженный границей border, которая в свою очередь окружена полями margin.
Визуально представление Блочной модели выглядит так:



Или так:

То есть margin определяет пространство за пределами (снаружи) элемента и служит для создания внешнего отступа, то есть отступы будут делаться от элемента, а padding определяет пространство внутри элемента (внутренние отступы), то есть содержимое элемента будет удаленно от его внутренних границ на указанные значения.

5. Как в CSS определяются приоритеты? Какое из свойств будет приоритетнее: #link.main или span #login?

селектор	a	b	c	d	Число
#link .main	0	1	1	0	110
span #login	0	1	0	1	101

Приоритетнее #link .main

6. В чем разница между CSS1 и CSS3?

CSS3 — это последняя версия языка каскадных таблиц стилей, а цифра 3 — это не просто ссылка на новые функции в CSS, но и третий уровень в процессе разработки спецификации CSS. CSS3 — это модульная версия спецификации CSS с новыми функциями, которые позволяют вывести веб-дизайн на совершенно новый уровень, делая его более простым и гибким.

Разница:

1) Селекторы. CSS3 предлагает совершенно новый, надежный набор инструментов с расширенными селекторами, которые позволяют выбирать более конкретные элементы для стилизации, устраняя необходимость в произвольных идентификаторах и классах. CSS3 расширяет базовую функциональность селектора атрибутов, позволяя выбирать элементы на основе строк в значениях атрибутов.
2) Цвет. Раньше цвета объявлялись в шестнадцатеричном формате — системе нумерации, начинающейся с 00 и заканчивающейся FF. Спецификация цвета CSS3 определяет расширенный список ключевых слов цвета, которые поддерживаются веб-браузерами. Список теперь включает дополнительные 147 ключевых слов цвета и модель «RGBA» (добавлен альфа-канал), которая позволяет определять непрозрачность цвета. Наиболее значительным изменением является то, что теперь можно объявлять полупрозрачные цвета.
3) Скругление границ блока. Свойство CSS «border-radius» определяет закругленные углы любого элемента, что позволяет легко использовать закругленные углы в элементах дизайна. CSS3 также имеет дополнительные свойства фона, такие как возможность иметь несколько фонов и улучшения для управления размером фонового изображения, ориентацией и обрезкой. Свойство «box shadow» позволяет добавлять тени к элементам.
4) Форматирование текста. CSS3 предоставляет множество свойств форматирования текста, многие из которых уже присутствуют в CSS2, но с некоторыми дополнительными функциями, такими как новое свойство «text shadow», которое позволяет добавлять тени к отдельным символам в текстовых узлах.
5) Градиенты. Градиенты стали мощным дополнением к CSS, позволяющие установить градиентный цвет фона элемента.
6) В CSS3 есть еще одна интересная функция, называемая «transitions», которая позволяет управлять скоростью анимации при переходе от одного значения свойства CSS к другому. Например, можно анимировать высоту элемента от нуля до сотни пикселей, чтобы показать содержимое элемента. CSS3-анимации имеют свою собственную спецификацию, и они позволяют создавать ключевые кадры для управления анимацией и свойствами, которые позволяют управлять временем, длительностью и цикличностью анимации.

7. Что такое псевдоклассы? А псевдоэлементы?

Псевдоклассы и псевдоэлементы часто служат довольно специфическим целям.

Псевдокласс — это селектор, который выбирает элементы, находящиеся в специфическом состоянии, например, они являются первым элементом своего типа, или на них наведён указатель мыши.

Синтаксис выглядит так:
Селектор: псевдокласс {…}
Например, нам нужно установить особый шрифт/цвет первому абзацу текста.
Мы можем задать стиль по классу, добавив атрибут класс к нужному абзацу. Но, абзац может измениться, добавится новый и т.д. Тогда атрибут класс придется постоянно переносить. Чтобы этого не делать, можно установить стиль к нужному селектору через псевдокласс :first-child — он всегда будет нацелен на первый дочерний элемент в статье, и нам больше не нужно будет редактировать HTML.
Псевдоэлементы действуют так, как если бы я добавила в разметку целый новый HTML-элемент, а не применяла класс к существующим элементам. Псевдоэлементы начинаются с двойного двоеточия "::"
Например, я хочу добавить стиль к первой строке абзаца, я могла бы обернуть её в и использовать селектор элемента; но я не могу точно знать, какое количество слов войдет в эту строку при отображении на разных экранах, то есть это может не сработать. Для этой задачи я могу использовать псевдоэлемент ::first-line — если количество слов увеличивается или уменьшается, он всё равно будет выбирать только первую строку.

8. Изучите статью про «плохие теги» и пришлите список тегов, которые не желательно использовать

Строгое «нет»:
<u> <layer> <blink> и <marquee> <font>

Теги, у которых есть лучшая альтернатива:
 или свойство CSS вместо <b>,
 или свойство CSS вместо <i>,
<h1>, <h2> etc или font-size CSS вместо <big>,
font-size CSS вместо <small>,
border-top или border-bottom CSS вместо <hr>.

«Плохие» атрибуты:
text bgcolor в теге <body>, вместо них – CSS: color и background-color
background в теге <body>, вместо него – CSS: background-image
link, alink, vlink в теге <body>, вместо них – псевдоклассы CSS :link, : active, :visited
align, вместо него: CSS text-align
target – с осторожностью

9. Как можно подключать шрифты локально?

Можно подключить через правило (директиву) @font-face, когда на компьютере уже установлен нужный шрифт, то есть файлы со шрифтами в таком случае хранятся вместе с остальными ресурсами сайта в корне проекта. Обычно для этого создается отдельная папка, куда помещаются файлы со шрифтами, как правило, в форматах `woff` и `.woff2`, которые поддерживают любые браузеры.
После того, как шрифты добавлены в проект, их нужно подключить в CSS-файле. Правило @font-face будет включать:

1) Название шрифта, которое затем нужно использовать, чтобы задать элементам подключённый шрифт.
2) Адрес файла со шрифтом, который нужно подключить, и его формат. Если адресов несколько, их можно указать через запятую. В этом случае важен порядок — браузер будет последовательно пытаться подключить файлы. Первым должен быть самый подходящий формат, а далее — запасные варианты.
3) Также с помощью функции `local` можно добавить возможность перед загрузкой шрифта с сервера проверить, установлен ли он на компьютере пользователя. Если да, запроса к серверу за шрифтом не будет — при рендеринге используется локальная версия. Но у этого способа есть минус — шрифт

на компьютере пользователя может быть устаревшим, и тогда страница отобразится не совсем так, как было задумано.
4) Начертания: жирное, курсивное и так далее. Для каждого начертания нужно отдельное правило `@font-face`.
Базовый вариант правила:

```
@font-face {
  font-family: "Roboto";
  font-style: normal;
  font-weight: 400;
  /* Браузер сначала попробует найти шрифт локально */
  src: local("Roboto"),
       /* Если не получилось, загрузит woff2 */
       url("/fonts/roboto.woff2") format("woff2"),
       /* Если браузер не поддерживает woff2, загрузит woff */
       url("/fonts/roboto.woff") format("woff");
}

/* Теперь можно использовать шрифт */
body {
  font-family: "Roboto", "Arial", sans-serif;
}
```

Для улучшения производительности правило `@font-face` лучше всего прописывать в самом начале CSS-файла. Так браузер сможет раньше начать обработку шрифта.

10. Почему не стоит использовать сокращенную запись без необходимости? И если все же использовать, как это делать правильно?

Используя сокращённое свойство, можно писать более сжатые (и часто более читаемые) таблицы стилей, экономя время и энергию.

Однако нужно понимать во что разворачивается сокращенная запись, и что получится, если в ней будут пропущены какие-нибудь свойства.
При использовании короткой записи, свойства, которые не были указаны, сбрасываются к значениям по умолчанию, то есть какое-то свойство можно потерять. То есть сокращенная запись «сбросит» другие свойства, которые мы не собирались изменять.
А вот если описать свойства, например, background в обоих классах полными свойствами вместо сокращённых - они друг друга дополнят. А ещё это позволяет удобнее читать стили: порядок свойств в сокращённом свойстве background произвольный. Сокращённые свойства удобно писать, но сложно комбинировать и понимать, поэтому модульные стили лучше писать развёрнуто.
Чтобы не допускать ошибок, следует, во-первых, группировать свойства по смыслу, это позволит быстрее находить ошибки, а во-вторых, если нужно переопределить значения ранее заданных свойств, не использовать сокращённую запись.
Достаточно «безопасно» использовать сокращенную запись в следующих примерах:

```
.box {
  padding: 10px;
}
```

Здесь мы ничего не сбросим случайно, потому что хотим, чтобы все четыре стороны содержали 10px отступы. То есть сокращение имеет смысл, то же самое с margin.
Главное, что нужно помнить, это то, что сокращение — это плохо, когда оно влияет на свойства, которые мне на самом деле не нужно изменять.

11. Разберитесь самостоятельно, как сделать анимацию через CSS

1) CSS transitions

5 свойств, которые позволяют контролировать transition-анимацию:

transition-property;	указывает список свойств, которые будут анимироваться; свойства, которые здесь не указаны, будут изменяться обычным образом. Можно анимировать все свойства для конкретного элемента, указав значение all. Если не указано ни одного свойства, то по умолчанию используется значение all.
transition-duration;	задаёт значение продолжительности анимации, время можно указывать в секундах или миллисекундах.
transition-timing-function;	временная функция, указывает точки ускорения и замедления за определенный период времени для контроля изменения скорости анимации. Проще говоря, с помощью этого свойства можно указать поведение для анимации. Например, мы можем ускорить анимацию в начале и замедлить в конце, либо наоборот.
transition-delay;	задаёт задержку времени до начала анимации, можно указывать в секундах или миллисекундах.
transition;	это общее свойство, которое позволяет перечислить первые четыре свойства в порядке: property, duration, timing-function, delay.

2) CSS animations

Позволяют делать более сложные анимации, нежели CSS transitions.
Создание анимации начинается с установки ключевых кадров - правила @keyframes. Кадры определяют, какие свойства на каком шаге будут анимированы.
Правило @keyframes содержит имя анимации элемента, которое связывает правило и блок объявления элемента.

animation-name;	Указывается имя анимации, которое связывает правило @keyframes с селектором:
	`animation-name: my-animation;`
	`@keyframes my-animation {`
	`Список правил;`
	`}`

animation-duration; animation-timing-function; animation-delay;	Работают аналогично одноименных transition
animation-iteration-count;	задаёт количество повторов анимации, значение по умолчанию 1. Значение infinite означает, что анимация будет проигрываться бесконечно.
animation-direction;	Задает направление анимации
animation-play-state;	остановка и прогрывание анимации. Два значения: running (анимация проигрывается, по умолчанию) и paused (останавливает анимацию).
animation-fill-mode;	устанавливает, какие CSS-свойства будут приминены к объекту до или после анимации. Может принимать такие значения: none — анимируемые CSS-свойства применятся к объекту только во время воспроизведения анимации, по окончании объект возвращается в исходное состояние; forwards — анимируемые CSS-свойства применятся к объекту по окончании воспроизведения анимации; backwards — анимируемые CSS-свойства применятся к объекту до начала воспроизведения анимации; both — анимируемые CSS-свойства применятся к объекту и до начала, и после окончания воспроизведения анимации;