Курс «Основы разработки сайтов»

Занятие №3

Продвинутые возможности CSS3

Время для вопросов

- Какие вопросы остались открытыми после предыдущего занятия?
- Какие проблемы возникли при выполнении дз?

Небольшой тест

- https://docs.google.com/forms/d/e/1FAIpQLScEEGkPx0REhQkzV4KavhJvGOmx1K3dBJiyC5cKBsbC3txZqA/viewform?usp=sf_link

Продвинутые возможности CSS3

План сегодняшнего занятия:

- CSS-свойство display и модели верстки (CSS Flexbox и CSS Grid).
- Использование CSS-медиазапросов для разработки адаптивного дизайна.
- Анимация и трансформация в CSS. Библиотеки анимации, CSS-генераторы.
- Использование препроцессоров на примере LESS

CSS3. Свойство display

- У каждого браузера есть свои стили по умолчанию. И для каждого элемента прописано свойство **display.** Оно указывает, как именно будет отображаться элемент.
- Значения свойства display:
  - **inline** - элемент будет отображаться как строчный. Примеры, для которых по умолчанию установлен inline - ,<a>,<img>.
  - **block -** элемент будет отображаться как блочный. Примеры, для которых по умолчанию установлен block - ,<h1> - <h6>,<p>,<form>,<header>,<footer>,<section>.
  - **none** - элемент не будет отображаться.
- **display:none** похож по смыслу со свойством **visibility:hidden;**, но есть глобальное отличие - при visibility:hidden на месте неотображаемого элемента будет пустое пространство, зарезервированное под этот элемент, а при display:none такого места не будет
- Пример display1

CSS3. Свойство display

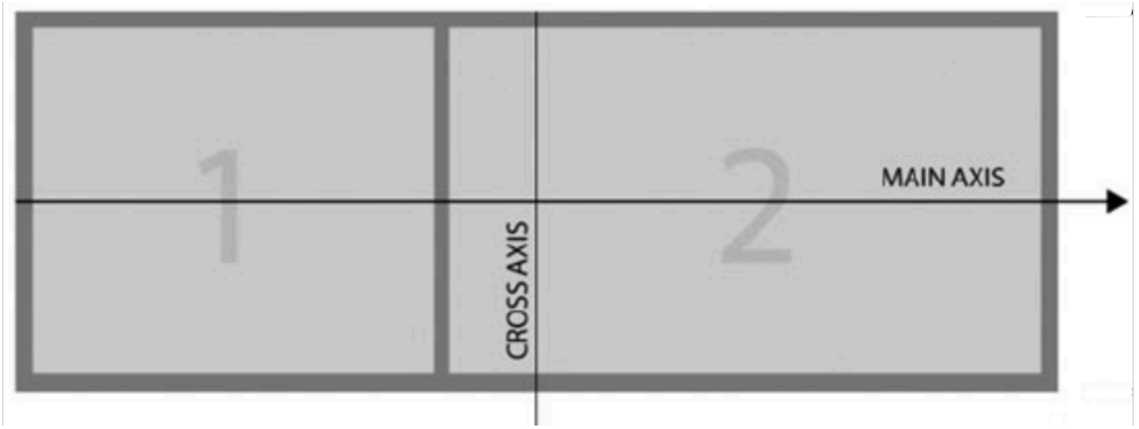
- Значение display: **inline-block** означает, что элемент будет иметь среднее значение между **inline** и **block**.
- По отношению к соседним внешним элементам такой элемент расценивается как строчный. То есть он не отделяется от соседних элементов переводом строки. Однако по отношению к вложенным элементам он рассматривается как блочный. И к такому элементу применяются свойства width, height, margin-top и margin-bottom.
- Пример display2
  - Задача. Не меняя HTML-кода, сделайте так, чтобы каждое предложение начиналось с новой строки (displayTask1)

CSS3. FlexBox. Основные понятия

- **Flexbox** - это общее название для модуля **Flexible Box Layout**, который имеется в CSS3. Данный модуль определяет особый режим компоновки/верстки пользовательского интерфейса.
- Благодаря Flexbox проще создавать сложные, комплексные интерфейсы, где с легкостью можно переопределять направление и выравнивание элементов, создавать адаптивные табличные представления.
- Основными составляющими компоновки flexbox являются **flex-контейнер** (flex container) и **flex-элементы** (flex items). Flex container представляет некоторый элемент, внутри которого размещены flex-элементы.

CSS3. FlexBox. Основные понятия

- Одно из ключевых понятий представляет **main-axis** или центральная ось. Это условная ось во flex-контейнере, вдоль которой позиционируются flex-элементы. Элементы в контейнере могут располагаться по горизонтали в виде строки и по вертикали в виде столбца.
- Кроме основной оси существует также поперечная ось или **cross axis**. Она перпендикулярна основной.



CSS3. FlexBox. Создание flex-контейнера

- Для создания **flex-контейнера** необходимо присвоить его стилевому свойству **display:flex;**
- Пример flex1
- Свойство **flex-direction** указывает направление главной оси и может иметь следующие значения:
  - **row**: значение по умолчанию, при котором элементы располагаются в виде строки слева направо
  - **row-reverse**: элементы также располагаются в виде стоки только в обратном порядке справа налево
  - **column**: элементы располагаются в столбик сверху вниз
  - **column-reverse**: элементы располагаются в столбик снизу вверх
- Пример flex2
- **display: inline-flex** - делает flex-контейнер inline элементом (в отличии от flex, который делает контейнер блочным)

CSS3. FlexBox. Перенос элементов

- А что будет если элементов в flex-контейнере будет очень много?
- Свойство **flex-wrap** определяет, будет ли перенос элементов flex-контейнера в следующие строки или столбцы в случае если его размеры недостаточны, чтобы вместить в один ряд все элементы.
- Значения:
  - **nowrap**: значение по умолчанию, которое определяет flex-контейнер без переноса
  - **wrap**: если элементы не помещаются во flex-контейнер, то создает дополнительные ряды в контейнере для размещения элементов. При расположении в виде строки создаются дополнительные строки, а при расположении в виде столбца добавляются дополнительные столбцы
  - **wrap-reverse**: то же самое, что и значение wrap, только элементы располагаются в обратном порядке
- Пример flex3
- Свойство **flex-flow** позволяет установить значения сразу для обоих свойств flex-direction и flex-wrap. Оно имеет следующий синтаксис: flex-flow: [flex-direction] [flex-wrap].

CSS3. FlexBox. Позиционирование элементов

- Для управления пустым пространством во flex-контейнере мы можем применять свойство **justify-content,** которое выравнивает элементы вдоль основной оси.
- Значения:
- **flex-start**: значение по умолчанию, при котором первый элемент выравнивается по началу главной оси
- **flex-end**: последний элемент выравнивается по концу главной оси
- **center**: элементы выравниваются по центру
- **space-between**: первый и последний элементы закрепляются в начале и конце главной оси, а все пустое пространство контейнера распределяется равным образом между всеми оставшимися элементами.
- **space-around**: элементы равным образом распределяют пространство между левым и правым краем контейнера, а расстояние между первым и последним элементом и границами контейнера составляет половину расстояния между элементами.
- Пример flex4

CSS3. FlexBox. Позиционирование элементов

- Свойство **align-items** также выравнивает элементы, но уже по поперечной оси
- Значения:
  - **stretch**: значение по умолчанию, при котором flex-элементы растягиваются по всей высоте (при расположении в строку) или по всей ширине (при расположении в столбик) flex-контейнера.
  - **flex-start**: элементы выравниваются по верхнему краю (при расположении в строку) или по левому краю (при расположении в столбик) flex-контейнера
  - **flex-end**: элементы выравниваются по нижнему краю (при расположении в строку) или по правому краю (при расположении в столбик) flex-контейнера
  - **center**: элементы выравниваются по центру flex-контейнера
- Также есть свойство **align-self,** которое позволяет переопределить значение свойства align-items для одного элемента. Оно может принимать все те же значения и указывается для flex-элемента
- Пример flex5

CSS3. FlexBox. Работа с размером элемента

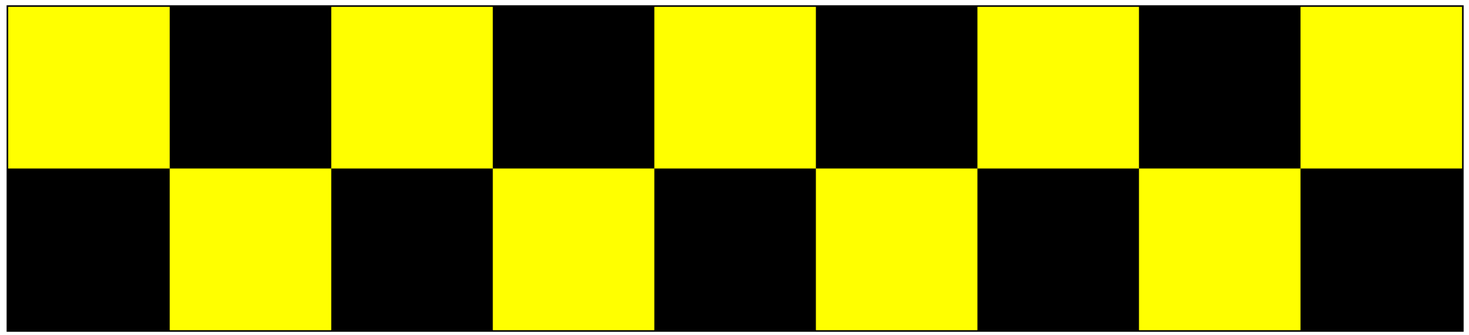
- Кроме свойств, устанавливающих выравнивание элементов относительно границ flex-контейнера, есть еще три свойства, которые позволяют управлять элементами:
- **flex-basis**: определяет начальный размер flex-элемента до того, как он начнет изменять размер, подстраиваясь под размеры flex-контейнера. Это свойство может принимать следующие значения: **auto -** начальный размер flex-элемента устанавливается автоматически; **числовое значение -** мы можем установить конкретное числовое значение для размеров элемента; Примечание: при установленном flex-basis с числовым значением свойство **width** игнорируется. Пример flex6
- **flex-shrink**: определяет, как flex-элемент будет уменьшаться относительно других flex-элементов во flex-контейнере. Если flex-контейнер имеет недостаточно места для размещения элемента, то мы можем указать коэффициент уменьшения. Пример flex7
- **flex-grow**: определяет, как flex-элемент будет увеличиваться относительно других flex-элементов во flex-контейнере. Работает обратно пропорционально свойству flex-shrink. Пример flex8
- Свойство **flex** является объединением свойств flex-basis, flex-shrink и flex-grow и имеет следующий формальный синтаксис: flex: [flex-grow] [flex-shrink] [flex-basis]
- Пример flex9

CSS3. FlexBox. Позиционирование элементов

- Для управления отступами между flex-элементами используется свойство **gap.**
- Оно может принимать одно значение (устанавливает внешний отступ между элементами как по горизонтали, так и по вертикали)
- Или же два значения (первое - вертикальный отступ, второе - горизонтальный отступ)
- Пример flex10

CSS3. FlexBox

- Задача. Нарисовать шашечки такси, используя разметку из flexTask1.



CSS3. Grid. Создание сетки

- **Grid Layout** представляет специальный модуль CSS3, который позволяет позиционировать элементы в виде сетки или таблицы **(display: grid;)**
- Грид образует сетку из строк и столбцов, на пересечении которых образуются ячейки. И для установки строк и столбцов в Grid Layout использовать следующие свойства CSS3
- **grid-template-columns:** настраивает столбцы. В качестве значения передается ширина столбцов. Сколько мы хотим иметь в гриде столбцов, столько и нужно передать значений этому свойству.
- **grid-template-rows:** настраивает строки. Передается высота каждой из строк.
- В то же время, если элементов больше, чем ячеек грида, то образуются дополнительные строки. А если ячеек грида больше, чем элементов, то все незанятые ячейки грида остаются пустыми.
- Если у нас столбцов и(или) строк много и они имеют одинаковые размеры, то есть смысл использовать специальную функцию **repeat(count, size)**
- Свойство **grid** объединяет свойства grid-template-rows и grid-template-columns и разом позволяет задать настройки для строк и столбцов в следующем формате:
  grid: grid-template-rows / grid-template-columns;
- Пример grid1

CSS3. Grid. Единицы измерения и отступы

- В качестве размеров строк и столбов можно указывать:
  - Фиксированные значения - **px, em, rem, %.**
  - Автоматические значения - **auto**. В этом случае ширина столбцов и высота строк вычисляются исходя из размеров содержимого
  - Пропорциональные размеры - **fr.** Вычисление пропорциональных размеров строк или столбцов производится по формуле:
    [fr для конкретного элемента] * [свободное место] / [сумма всех fr]
- Для создания отступов между столбцами и строками применяются свойства **grid-column-gap** и **grid-row-gap** соответственно. Если значения совпадают, то их можно объединить в одно свойство **gap.**
- Пример grid2

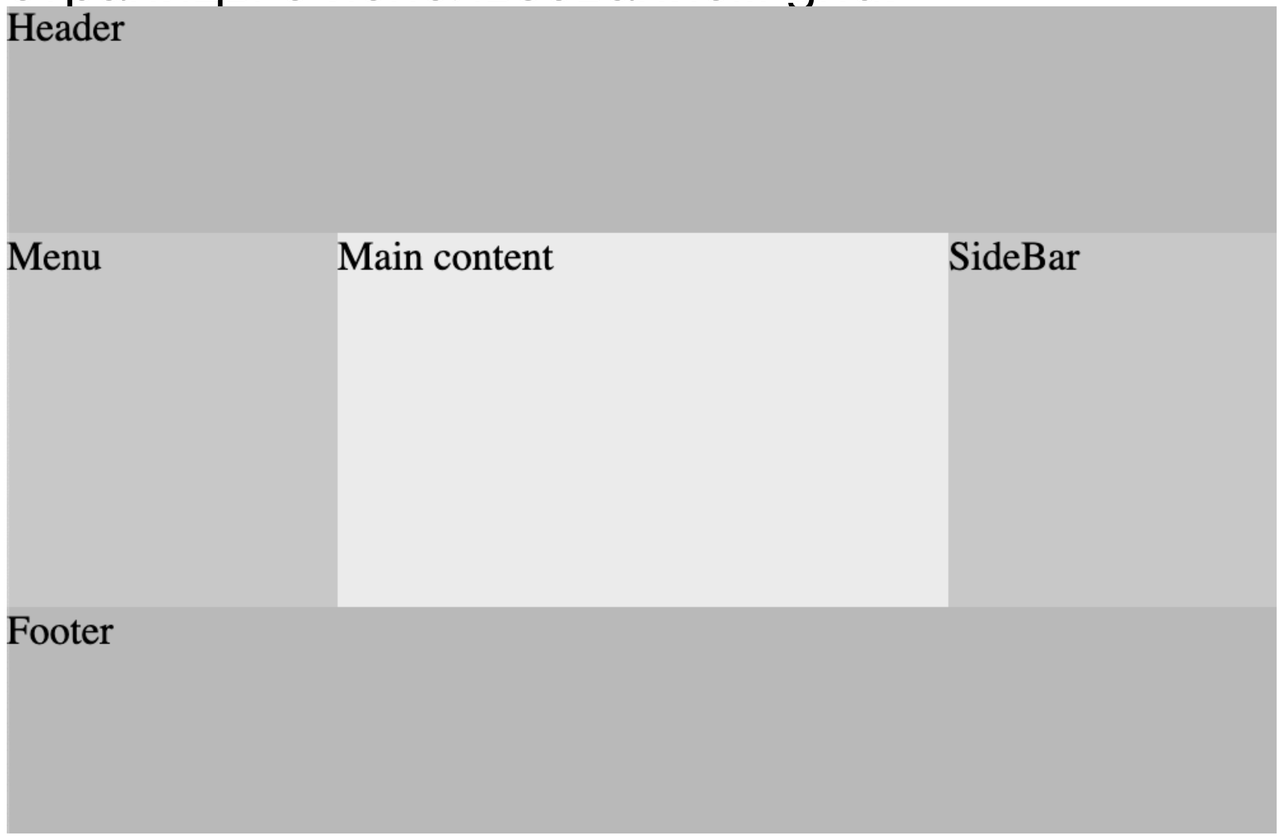
CSS3. Grid. Объединение ячеек

- Каждый элемент в гриде позиционируется в одну ячейку по порядку. Но мы можем более точно настроить расположение элемента в гриде с помощью ряда свойств
- **grid-row-start:** задает начальную строку, с которой начинается элемент
- **grid-row-end:** указывает, до какой строки не включительно надо растягивать элемент.
- Для их объединения используется одно свойство **grid-row: grid-row-start / grid-row-end;**
- **grid-column-start:** задает начальный столбец, от которого начинается элемент
- **grid-column-end:** указывает, до какого столба не включительно нужно растягивать элемент.
- Для их объединения используется одно свойство **grid-column: grid-column-start / grid-column-end;**
- С помощью специального слова **span** можно задать растяжение элемента на несколько ячеек. После слова span указывается, на какое количество ячеек надо растянуть элемент.
- Свойство grid-area объединяет свойства grid-column и grid-row, позволяя сократить их запись:
- grid-area: row-start / column-start / row-end / column-end
- Пример grid3

CSS3. Grid

- Для определения областей, которые будут занимать элементы, у grid-контейнера применяется свойство **grid-template-areas**, а для установки области у элементов задается свойство **grid-area**
- Пример grid4

- Задача. Реализовать верстку на всю высоту страницы с использованием grid.



CSS3. Медиазапросы для разработки адаптивного дизайна

- Адаптивная верстка позволяет правильно отображать сайт на различных разрешениях устройств, в том числе, на мобильных устройствах. Например, адаптивные макеты могут иметь три варианта отображения стилей в зависимости от ширины экрана: *меньше 1000px, 1000px-1479px, больше 1480px*.
- Для создания адаптивных дизайнов используют CSS-медиазапросы с помощью правила CSS ***@media***.
- Синтаксис:
  @media (min-width: 481px) and (max-width:768px) {}
- Пример adap1
  - Задача. Опишите медиа запросы для следующих трех устройств
    Телефон (ширина до 999px)
    Планшет (ширина от 1000 до 1479 px)
    Компьютер (ширина больше 1480 px)

CSS3. Трансформация в CSS.

- К трансформациям относятся такие действия, как вращение элемента, его масштабирование, наклон или перемещение по вертикали или горизонтали. Для создания трансформаций в CSS3 применяется свойство **transform**.
- Для поворота элемента свойство transform использует функцию rotate: **transform: rotate(угол_поворота deg);**
- Применение масштабирования имеет следующую форму: **transform: scale(величина_масштабирования);** Величина указывает во сколько раз увеличить или уменьшить элемент. Можно передать два значения через запятую для указания увеличения по горизонтали и вертикали.
- Для масштабирования по одной оси можно использовать **scaleX и scaleY.**

CSS3. Трансформация в CSS.

- Для перемещения элемента используется функция translate: **transform: translate(offset_X, offset_Y);** Аналогично можно указывать по отдельности **translateX** и **translateY.**
- Для наклона элемента применяется функции **skew(), skewX(),skewY()**: **transform: skew(X, Y);**
- Если надо применить к элементу сразу несколько преобразований, например поворот и перемещение, то мы можем их комбинировать, указывая через пробел.
- По умолчанию при применении трансформаций браузер в качестве точки начала преобразования использует центр элемента. Но с помощью свойства **transform-origin** можно изменить исходную точку, указав одно из значений: **left top, left bottom, right top, right bottom**
- Пример transform1

CSS3. Анимация в CSS. Transition.

- Переход (**transition**) представляет анимацию от одного стиля к другому в течение определенного периода времени. Для создания перехода необходимы прежде всего два набора свойств CSS: начальный стиль, который будет иметь элемент в начале перехода, и конечный стиль - результат перехода.
- Чтобы указать свойство как анимируемое, его название передается свойству **transition-property,** например:
  transition-property: background-color;
- При необходимости мы можем анимировать сразу несколько свойств CSS, указывая их через запятую в **transition-property** или передав значение **all** для всех.
- Далее идет установка времени перехода в секундах с помощью свойства **transition-duration: 2s;**
- В CSS для запуска перехода можно применять псевдоклассы.
- Пример transition1

CSS3. Анимация в CSS. Transition.

- Свойство **transition-timing-function** позволяет контролировать скорость хода и выполнение анимации.
- **linear**: линейная функция плавности, изменение свойства происходит равномерно по времени
- **ease**: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, предоставляя более естественное изменение
- **ease-in**: функция плавности, при которой происходит только ускорение в начале
- **ease-out**: функция плавности, при которой происходит только ускорение в конце анимации
- Свойство **transition-delay** позволяет определить задержку перед выполнением перехода. Значение - время.
- Свойство **transition** представляет сокращенную запись выше рассмотренных свойств **transition-property, transition-duration, transition-timing-function, transition-delay**
- Пример transition2

CSS3. Анимация в CSS. Keyframes

- Также для создания анимации есть более мощный инструмент под названием **keyframes**. Базовый синтаксис:

```
@keyframes название_анимации {
    from { /* начальные значения свойств CSS */ }
    to { /* конечные значения свойств CSS */ }
}
```

- Чтобы прикрепить анимацию к элементу, у него в стиле применяется свойство **animation-name**. Значение этого свойства - название применяемой анимации.
- Также с помощью свойства **animation-duration** необходимо задать время анимации в секундах или миллисекундах.
- Пример animation1

CSS3. Анимация в CSS. Keyframes

- Также animation позволяет нам иметь несколько промежуточных состояний, используя синтаксис:
  ```
  @keyframes название_анимации {
      from { /* начальные значения свойств CSS */ }
      25% { /* промежуточные значения свойств CSS */ }
      to { /* конечные значения свойств CSS */ }
  }
  ```
- Можно определить несколько отдельных анимаций, но применять их вместе, указывая через запятую.
- свойство **animation-iteration-count** определяет, сколько раз будет повторяться анимация. Если передать значение **infinite** - анимация будет длиться бесконечно.
- Как у transition, у animation есть аналогичные свойства **animation-delay** и **animation-timing-function**.
- Пример animation2

CSS3. Библиотеки анимации. CSS генераторы

- Для упрощения создания типовых эффектов анимации существуют готовые **библиотеки CSS**. Например Animate.css (https://animate.style/).
- Подключаются такие библиотеки как и обычный css, через элемент link. Они дают возможность использовать уже написанные в них классы.
- Также для упрощения разработки существуют онлайн ресурсы, позволяющие генерировать CSS-код в визуальном режиме. Такие ресурсы называются ***CSS-генераторы***. Например https://cssgrid-generator.netlify.app/

CSS3. Использование препроцессоров на примере LESS

- Поскольку CSS не обладает всеми возможностями языков программирования, придумали **CSS-препроцессор**, которые имеют возможность писать вложенные стили, использовать переменные, вспомогательные функции и вычисления при написании CSS кода.
- Процесс работы с less:
  - Предварительно должен быть скачен Node.js (https://nodejs.org/)
  - Проинициализировать проект с помощью npm init
  - Скачать less с помощью команды npm install less
  - Создать файл с расширением .less
  - После окончания работы с файлом less трансформировать его в обычный css путем запуска команды npx lessx [lessfile] [new_css_file]

CSS3. Основные возможности LESS:

- Возможность написания вложенных стилей, например:

```
.container{
    background-color: red;

    div {
        color: pink;
    }
}
```

- Объявление переменных с помощью **@name: value;**

- Переиспользование части css в разных местах (миксины)

```
.rounded_top(@value) {
    border-top-left-radius: @value;
    border-top-right-radius: @value;
}
.tab {
    background: #333;
    color:#fff;
    .rounded_top(6px);
}
.submit {
    .rounded_top(20px);
}
```

- Также миксины можно использовать с параметрами. Пример less

Итого

- Разобрали свойство display и познакомились с flexbox и grid
- Изучили создание адаптивный верстки с помощью медиа запросов
- Поговорили о возможности использования трансформаций и анимаций в css
- Попробовали на практике использование препроцессора less

Задачи

https://flexboxfroggy.com/#ru

Спасибо за внимание!

Ресурсы для изучения:

- http://htmlbook.ru/samcss
- https://webref.ru/practice
- https://flexboxfroggy.com/#ru