

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Практическая работа №1

Выполнила:

Жигалова Анастасия

К33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Практической задачей было пройти 3 игры. Научиться использовать Flexbox, Grid, Git.

Ход работы

Первой мной была пройдена игра FLEXBOX FROGGY. В ней рассматривались некоторые свойства:

`justify-content` - выравнивает flex-элементы вдоль главной оси и принимает следующие значения (*flex-start*: элементы выравниваются по левой стороне контейнера; *flex-end*: элементы выравниваются по правой стороне контейнера; *center*: элементы выравниваются по центру контейнера; *space-between*: элементы отображаются с одинаковыми отступами между ними; *space-around*: элементы отображаются с одинаковыми отступами вокруг них)

`align-items` - выравнивает flex-элементы вдоль пересекаемой оси и принимает следующие значения (*flex-start*: элементы выравниваются по верхнему краю контейнера; *flex-end*: элементы выравниваются по нижнему краю контейнера; *center*: элементы выравниваются вертикально по центру контейнера; *baseline*: элементы отображаются на базовой линии контейнера; *stretch*: элементы растягиваются, чтобы заполнить контейнер)

`flex-direction` - задает направление основной оси и принимает следующие значения (*row*: элементы размещаются по направлению текста; *row-reverse*: элементы отображаются в обратном порядке к направлению текста; *column*: элементы располагаются сверху вниз; *column-reverse*: элементы располагаются снизу вверх)

`order` - указывает порядок flex-элемента и принимает значения всех отрицательных и положительных целых чисел, 0 стоит по дефолту.

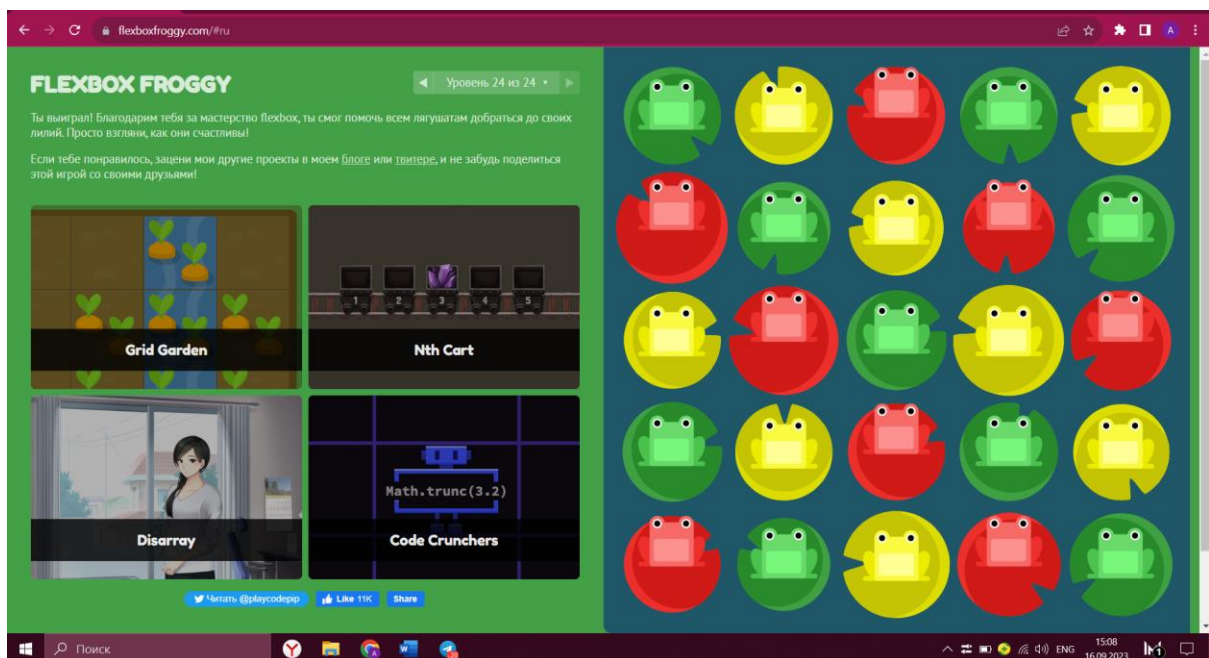
`align-self` - выравнивает flex-элемент вдоль пересекаемой оси, перекрывая значения свойства `align-items`, принимает значения, такие же, как и `align-items`.

`flex-wrap` - указывает, нужно ли чтоб элементы принудительно находились в одном ряду или автоматически переносились, и принимает следующие значения (*nowrap*: размеры элементов устанавливаются автоматически,

чтобы они поместились в один ряд; wrap: элементы автоматически переносятся на новую строку; wrap-reverse: элементы автоматически переносятся на новую строку, но строки расположены в обратном порядке.)

flex-flow – свойство, комбинирующее в себе flex-direction и flex-wrap. Это свойство принимает их значения, разделённые пробелом.

align-content - выравнивает ряды flex-контейнера внутри него (работает только, если элементы расположены больше чем в один ряд), принимает следующие значения (flex-start: ряды группируются в верхней части контейнера; flex-end: ряды группируются в нижней части контейнера. center: ряды группируются вертикально по центру контейнера; space-between: ряды отображаются с одинаковыми расстояниями между ними; space-around: ряды отображаются с одинаковыми расстояниями вокруг них; stretch: ряды растягиваются, чтобы заполнить контейнер равномерно)



Следующая игра - GRID GARDEN.

`grid-column-start` – определяет начальную `grid`-позицию внутри `grid`-столбцов (принимает значение целых чисел, в том числе отрицательных)

`grid-column-end` – определяет конечную `grid`-позицию внутри `grid`-столбцов (принимает значение целых чисел, в том числе отрицательные)

В `grid-column-start` и `grid-column-end` в качестве значений можно задавать необходимую ширину строк, используя `span`.

`grid-column` – комбинация `grid-column-start` и `grid-column-end`, значения указываются через `/`.

`grid-row-start` – определяет начальную `grid`-позицию внутри `grid`-строк (принимает значение целых чисел, в том числе отрицательных)

`grid-row-end` – определяет конечную `grid`-позицию внутри `grid`-строк (принимает значение целых чисел, в том числе отрицательные)

`grid-row` – комбинация `grid-row-start` и `grid-row-end`, значения указываются через `/`.

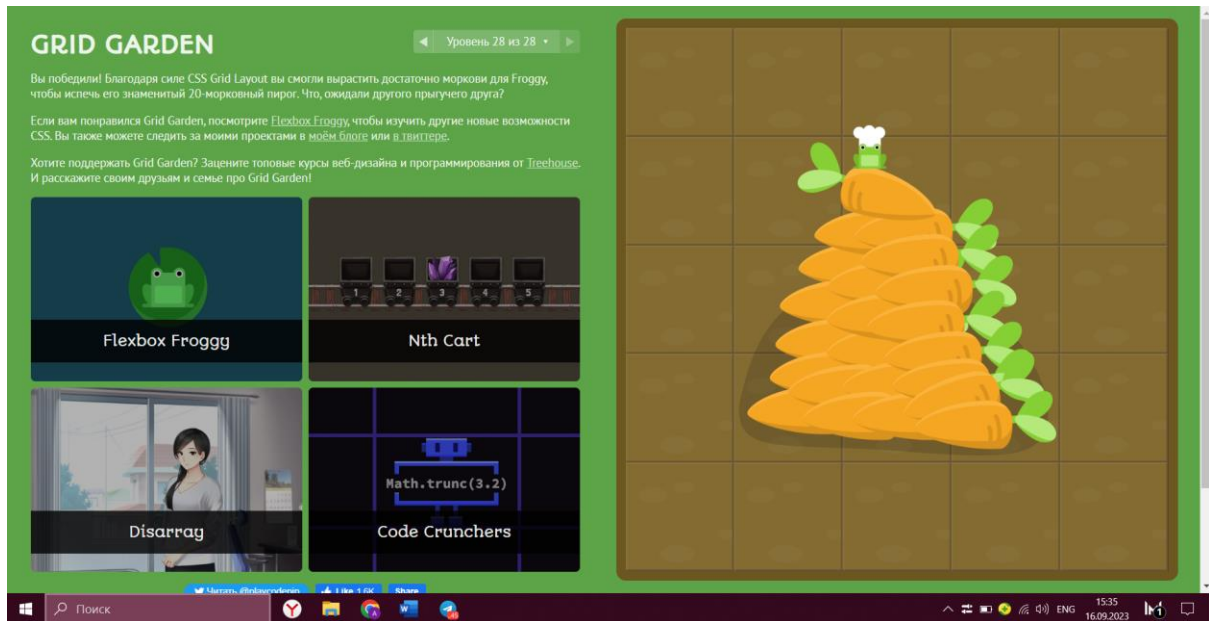
`grid-area` – комбинация `grid-row-start`, `grid-column-start`, `grid-row-end` и `grid-column-end`, записывается в этом же порядке через `/`.

`order` - указывает порядок `grid`-элемента и принимает значения всех отрицательных и положительных целых чисел, 0 стоит по умолчанию.

`grid-template-columns` – определяет размер и название для `grid`-столбцов. С помощью `repeat` можно записать несколько повторяющихся столбцов, не прописывая каждый. Значения принимаются в `px`, `fr` – выделяет часть свободного пространства, `%`, `em`: их можно комбинировать.

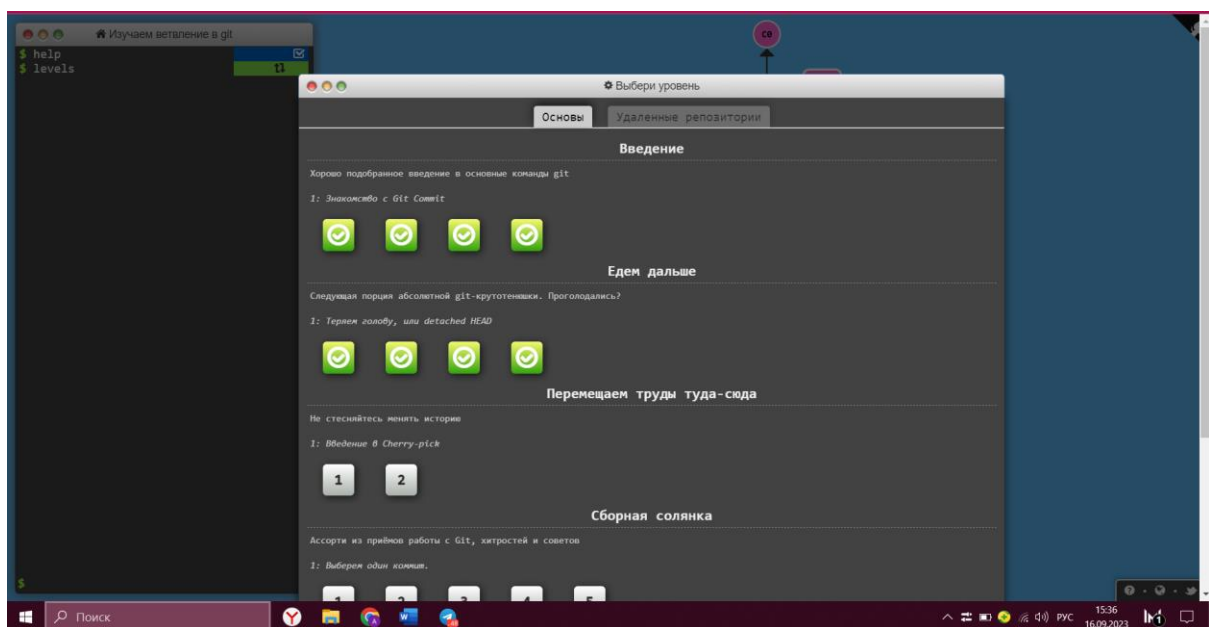
`grid-template-rows` работает точно так же, как и `grid-template-columns` и принимает такие же значения.

`grid-template` — сокращённый вариант комбинации `grid-template-rows` и `grid-template-columns`, записывается через `/`.



Последняя игра – основы Git.

В ней мной было повторено как создавать коммиты (`git commit`), как делать ветки (`git branch [branchName]`), как объединять изменения из своей ветки с общими (`git merge [branchName]`), отличие merge от rebase (`git rebase[branchName]`).



Вывод

В во время выполнения практической я впервые поработала с GRID CSS и FLEXBOX CSS, поняла, что Grid является макетом на основе контейнеров, а Flexbox на основе содержимого, и их можно комбинировать. Обновила свои знания о Git. Было интересно узнать новое, поиграв в игры.