

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Фронт-энд разработка

Отчет

Домашняя работа №1

Выполнил:

Кащеев Максим

Группа

JS211

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

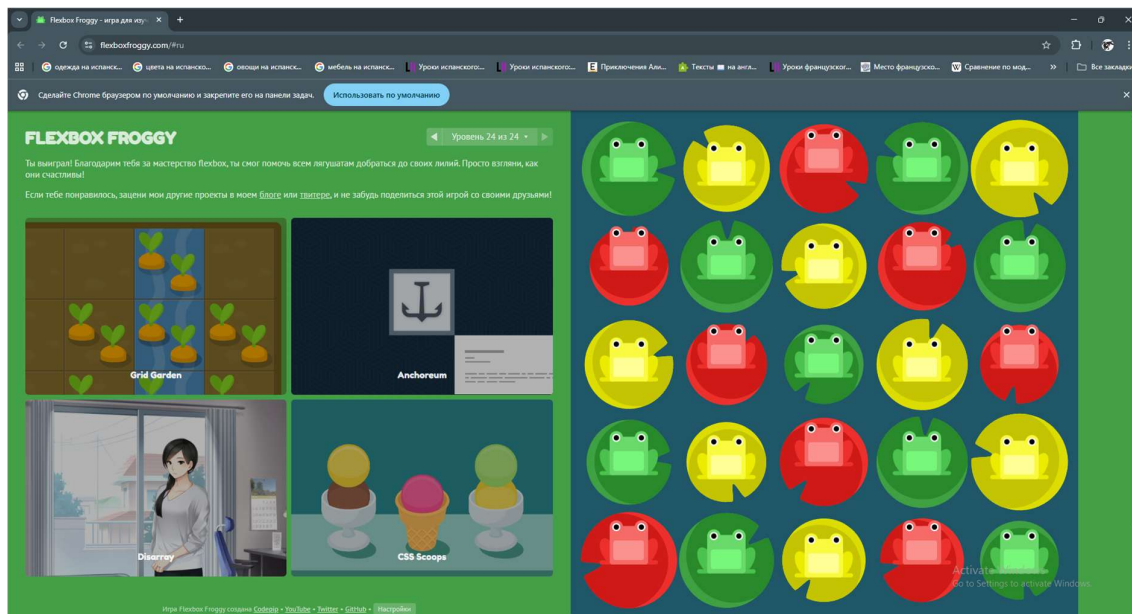
## Задача

Изучение CSS Flexbox, CSS Grid и git путём прохождения следующих материалов:

- <https://flexboxfroggy.com/#ru>
- <https://cssgridgarden.com/#ru>
- [https://learngitbranching.js.org/?locale=ru\\_RU](https://learngitbranching.js.org/?locale=ru_RU)

## Ход работы

В ходе выполнения домашнего задания были изучены базовые свойства CSS, а также CSS grid и git на примере трёх игр.



Во время прохождения игры flexbox froggy были изучены следующие свойства:

Свойство justify-content, которое выравнивает элементы горизонтально и принимает 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.

Свойство align-items, которое выравнивает элементы вертикально и принимает 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.

Свойство flex-direction, которое задаёт направление, в котором будут расположены элементы в контейнере и принимает 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.

Свойство Order, которое принимает целочисленные значения, в том числе отрицательные, при этом элементы сортируются по возрастанию значения order: чем меньше значение, тем раньше элемент будет расположен:

- **0** (значение по умолчанию): элементы отображаются в порядке, в котором они идут в исходном коде (если у всех order одинаковый)
- **Отрицательное число** (например, -1): элемент перемещается ближе к началу (раньше элементов с 0 и положительными значениями)
- **Положительное число** (например, 1, 2, 3): элемент перемещается ближе к концу (после элементов с меньшими значениями)

Свойство `align-self`, которое задаёт выравнивание конкретного flex-элемента по поперечной оси и тем самым определяет выравнивание, заданное `align-items` для всего контейнера:

- **auto**: значение по умолчанию; элемент наследует поведение выравнивания от `align-items` контейнера (то есть использует то же выравнивание, что и остальные элементы).
- **flex-start**: элемент прижимается к началу поперечной оси (например, при `flex-direction: row` - к верхнему краю контейнера).
- **flex-end**: элемент прижимается к концу поперечной оси (например, при `flex-direction: row` - к нижнему краю контейнера).
- **center**: элемент выравнивается по центру поперечной оси.
- **baseline**: элемент выравнивается по базовой линии (актуально, когда у элементов разный размер текста/шрифта).
- **stretch**: элемент растягивается, чтобы заполнить доступное пространство по поперечной оси (обычно работает, если у элемента не задан фиксированный размер по этой оси, например `height` для `row`).

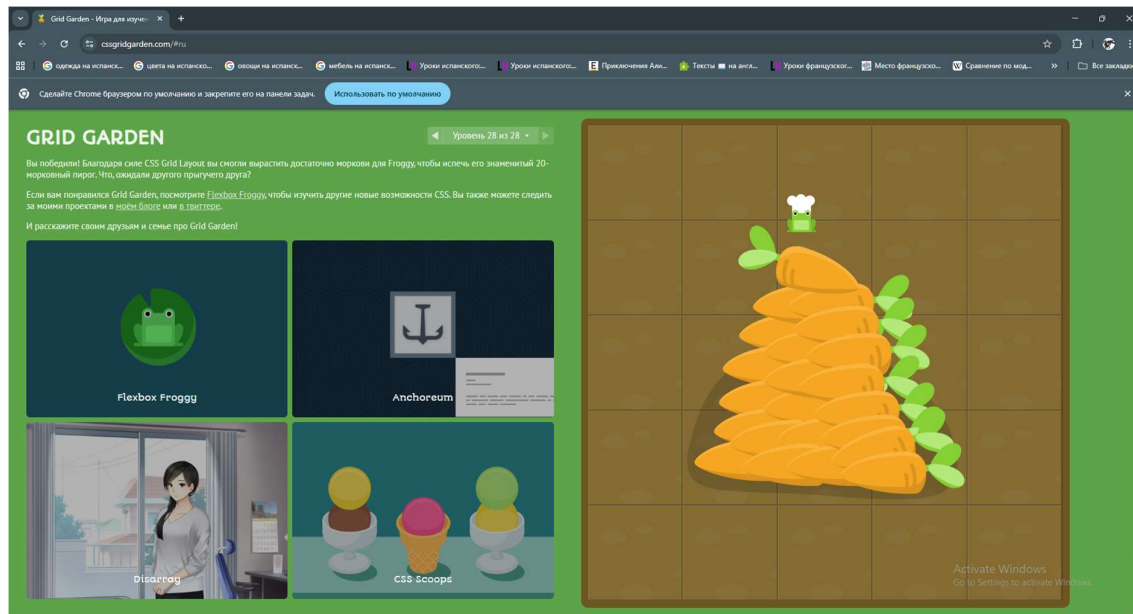
Свойство `flex-wrap`, которое определяет, будут ли flex-элементы располагаться в одну строку/колонок или им разрешено переноситься на новые строки/колоники, если им не хватает места в контейнере. Принимает следующие значения:

- **nowrap** (значение по умолчанию): все элементы стараются уместиться в одну линию; если места не хватает, они будут сжиматься (в рамках своих ограничений), чтобы не переноситься.
- **wrap**: элементы переносятся на следующую линию, если не помещаются; новые линии добавляются “вперёд” по поперечной оси.
- **wrap-reverse**: элементы тоже переносятся, но направление добавления новых линий меняется на противоположное (по поперечной оси).

Свойство `flex-flow`, которое одновременно задаёт направление главной оси и разрешение переноса `flex`-элементов, то есть объединяет `flex-direction` и `flex-wrap`. Принимает те же значения, что `flex-direction` + `flex-wrap`.

Свойство `align-content`, которое выравнивает строки/колонки `flex`-элементов внутри контейнера по поперечной оси, когда перенос включён и образовалось больше одной линии. Работает, только если у контейнера есть свободное место по поперечной оси и `flex-wrap` не равен `nowrap` (то есть несколько линий). Принимает значения:

- **stretch** (значение по умолчанию): линии растягиваются, чтобы занять всё доступное пространство по поперечной оси
- **flex-start**: все линии прижимаются к началу поперечной оси
- **flex-end**: все линии прижимаются к концу поперечной оси
- **center**: линии группируются по центру поперечной оси.
- **space-between**: линии распределяются равномерно, первая у начала, последняя у конца, между ними одинаковые промежутки.
- **space-around**: линии распределяются с одинаковыми промежутками вокруг каждой линии
- **space-evenly**: линии распределяются так, что промежутки между любыми соседними линиями и отступы до краёв контейнера одинаковые.



Во время прохождения игры css grid garden были изучены следующие свойства:

**grid-column-start** – свойство, которое задаёт, с какой вертикальной линией сетки начинается элемент по колонкам. Может принимать число, имя линии и span N (занять N колонок)

**grid-column-end** – свойство, которое задаёт, с какой вертикальной линией сетки заканчивается элемент по колонкам. Может принимать число, имя линии и span N (занять N колонок)

**grid-column** – сокращённое свойство, объединяющее grid-column-start и grid-column-end. Принимает значения: start / end, можно указать одно значение, тогда второе будет auto.

**grid-row-start** – свойство, которое задаёт, с какой горизонтальной линией сетки начинается элемент по строкам. Может принимать число, имя линии и span N (занять N колонок)

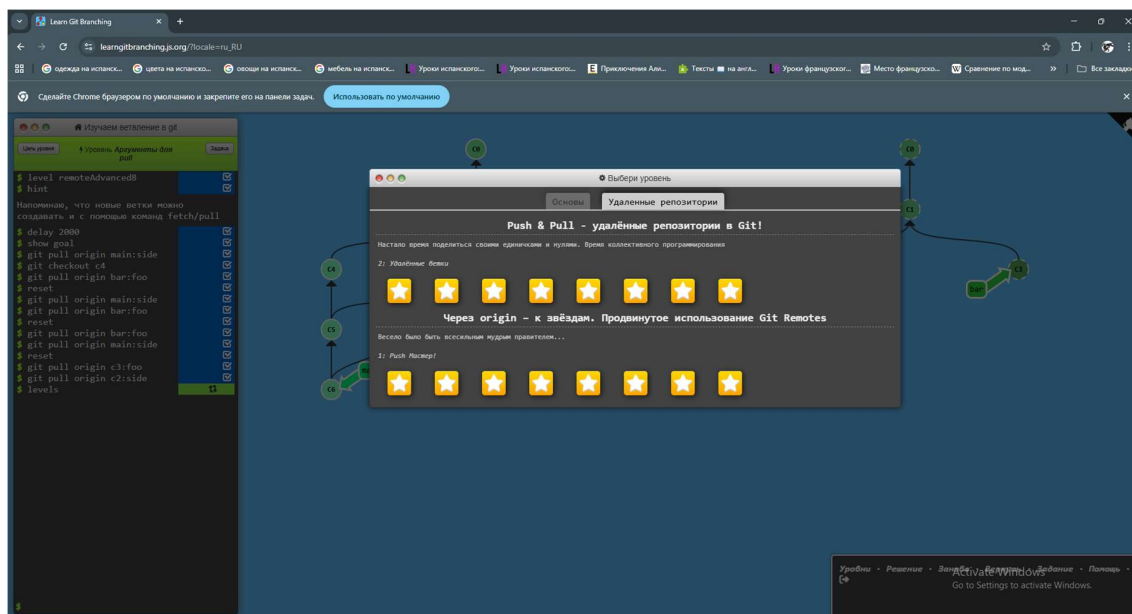
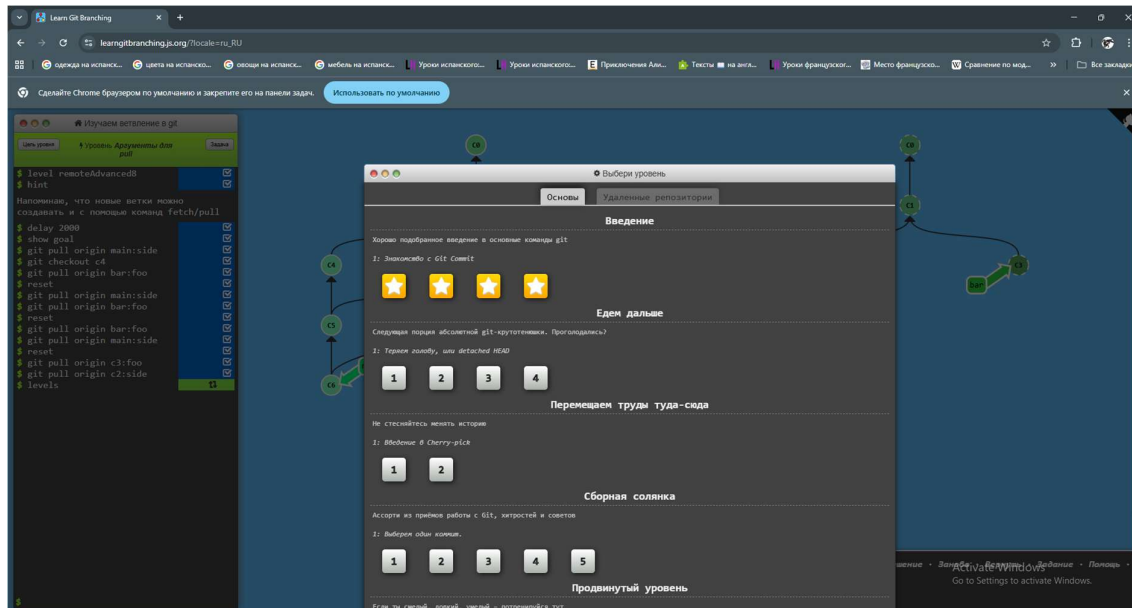
**grid-row-end** - свойство, которое задаёт, с какой горизонтальной линией сетки заканчивается элемент по строкам. Может принимать число, имя линии и span N (занять N колонок)

**grid-row** – сокращённое свойство, объединяющее grid-row-start и grid-row-end. Принимает значения: start / end

**grid-area** – принимает значения: row-start / column-start / row-end / column-end

**grid-template-columns** – свойство, которое задаёт количество и размеры колонок grid-контейнера. Принимает значения: список размеров, может включать именованные линии

**grid-template-rows** – свойство, которое задаёт количество и размеры строк grid-контейнера



Во время прохождения игры по изучению git были изучены основные команды для работы с локальными и удалёнными репозиториями.

## **Вывод**

В ходе выполнения домашнего задания были изучены базовые компоненты, необходимые для дальнейшего освоения курса. Например, изучение flex-свойств для дальнейшего форматирования и создания flex-контейнеров. Также в ходе работы были изучены свойства CSS Grid для построения сетки и настройки размеров её строк и столбцов с помощью различных единиц измерения. Важным аспектом изучения стало освоение git, которое поможет взаимодействовать с git(ом) непосредственно при работе с репозиториями, создавать коммиты и отправлять на проверку выполненные домашние задания/лабораторные работы.