

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

**Практическая работа 1 на тему: “Знакомство с CSS Grid,
Flexbox”**

**Выполнил: Безруков
А.И. (К33402)**

**Проверил:
Добряков Д. И.**

Санкт-Петербург

2021 г.

Задача

Познакомиться с основами CSS путем прохождения двух обучающих игр - Grid Garden, Flexbox Froggy

Ход работы

Grid garden

GRID GARDEN

◀ Уровень 1 из 28 ▶

Добро пожаловать в Grid Garden – место, где вы напишете CSS-код, чтобы вырастить морковный сад. Поливайте только те зоны, на которых есть морковь, используя свойство `grid-column-start`.

Например, `grid-column-start: 3` покроет водой зону, начинающуюся на третьей grid-линии по вертикали, это как сказать "третья вертикальная граница на grid-сетке слева".

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20% 20%;
4   grid-template-rows: 20% 20% 20% 20% 20%;
5 }
6
7 #water {
8   grid-column-start: 3
9 }
10
11
12
13
14
```

Следующий

GRID GARDEN

◀ Уровень 2 из 28 ▶

Oy, кажется, сорняки растут прямо в углу сада. Используйте `grid-column-start`, чтобы отравить их. Помните: сорняки начинаются на пятой вертикальной grid-линии.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20% 20%;
4   grid-template-rows: 20% 20% 20% 20% 20%;
5 }
6
7 #poison {
8   grid-column-start: 5 |
9 }
10
11
12
13
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 3 из 28 ▶

Когда вы используете только `grid-column-start`, grid-элемент по умолчанию "захватит" только один столбец. Однако вы можете увеличить элемент между несколькими столбцами, если добавите свойство `grid-column-end`.

Используя `grid-column-end`, полейте всю морковь, избегая пустые участки. Ведь мы не хотим использовать воду напрасно! Помните, что морковь начинается на первой вертикальной grid-линии и заканчивается на четвёртой.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column-start: 1;  
9   grid-column-end: 4  
10 }  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана Codepiper • GitHub • Twitter • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 4 из 28 ▶

Когда вы соединяли `grid-column-start` и `grid-column-end`, вы могли подумать, что конечное значение обязательно должно быть больше начального. Но это не так!

Попробуйте задать свойству `grid-column-end` значение меньше, чем 5, чтобы полить морковь.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column-start: 5;  
9   grid-column-end: 2  
10 }  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана Codepiper • GitHub • Twitter • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 5 из 28 ▶

Если вы хотите посчитать grid-строки справа налево вместо слева направо, вы можете задать `grid-column-start` и `grid-column-end` отрицательные значения. Например, вы можете присвоить значение, равное `-1`, чтобы указать первую grid-строку начиная справа.

Попробуйте присвоить `grid-column-end` отрицательное значение.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column-start: 1;  
9   grid-column-end: -2  
10 }  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 6 из 28 ▶

Теперь попробуйте присвоить `grid-column-start` отрицательное значение.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #poison {  
8   grid-column-start: -3|  
9 }
```

[Следующий](#)

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 7 из 28 ▶

Вместо определения grid-элемента на основе начальной и конечной позиции grid-строк вы можете задавать их с помощью необходимой вам ширины строк, используя `span`. Помните, что `span` работает только с положительными значениями.

Для примера, полейте эту морковь, используя свойство `grid-column-end: span 2`.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column-start: 2;  
9   grid-column-end: span 2;  
10 }  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 8 из 28 ▶

Попробуйте опять использовать `grid-column-end` вместе со `span`, чтобы полить морковь.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column-start: 1;  
9   grid-column-end: span 5;  
10 }  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 9 из 28 ▶

Вы также можете использовать `span` вместе с `grid-column-start`, чтобы присвоить значение ширины grid-элемента относительно конечной позиции.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20% 20%;
4   grid-template-rows: 20% 20% 20% 20% 20%;
5 }
6
7 #water {
8   grid-column-start: span 3
9   grid-column-end: 6;
10}
11
12
13
14
```

[Следующий](#)

Игра Grid Garden создана [Codepiper](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 10 из 28 ▶

Печатать каждый раз `grid-column-start` и `grid-column-end` может быть немного утомительно. К счастью, есть краткая форма `grid-column`, которая принимает оба значения сразу через косую черту: `/`.

Например, `grid-column: 2 / 4` скажет grid-элементу начаться на второй вертикальной grid-линии и закончиться на четвёртой.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20% 20%;
4   grid-template-rows: 20% 20% 20% 20% 20%;
5 }
6
7 #water {
8   grid-column: 4/6
9 }
10
11
12
13
14
```

[Следующий](#)

Игра Grid Garden создана [Codepiper](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 11 из 28 ▶

Попробуйте использовать `grid-column`, чтобы полить эту морковь. `span` также работает с этим сокращённым свойством, так что пробуйте!

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column: 2/ span 3  
9 }  
10  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана Codepen • GitHub • Twitter • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 12 из 28 ▶

Одна из вещей, которая отличает CSS Grid Layout от Flexbox'a, – это то, что вы можете легко позиционировать элементы в двух направлениях: в столбцах и в строках. `grid-row-start` работает почти как `grid-column-start`, но только по вертикальной оси.

Используйте `grid-row-start`, чтобы полить эту морковь.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-row-start: 3  
9 }  
10  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана Codepen • GitHub • Twitter • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 13 из 28 ▶

А теперь попробуйте использовать сокращённое свойство `grid-row`.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-row: 3/6  
9 }  
10  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана Codeipip • GitHub • Twitter • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 14 из 28 ▶

Используйте `grid-column` и `grid-row` одновременно, чтобы разместить элемент в двух направлениях.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #poison {  
8   grid-column: 2;  
9   grid-row: 5  
10 }
```

[Следующий](#)

Игра Grid Garden создана Codeipip • GitHub • Twitter • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 15 из 28 ▶

Вы также можете использовать `grid-column` и `grid-row` вместе, чтобы охватить более крупные зоны внутри grid-сетки. Пробуйте!

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-row: 1/6;  
9   grid-column: 2/6;  
10 }  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана [Codeipip](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 16 из 28 ▶

Если вас также утомляет печатать `grid-column` и `grid-row`, есть сокращённое свойство и для этого. `grid-area` принимает 4 значения, разделенные косой чертой `/`: `grid-row-start`, `grid-column-start`, `grid-row-end` и `grid-column-end`.

Пример: `grid-area: 1 / 1 / 3 / 6;`

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-area: 1/2/4/6;  
9 }  
10  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана [Codeipip](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 17 из 28 ▶

Как насчет множества элементов? Вы можете накладывать их друг на друга без проблем. Используйте `grid-area`, чтобы определить вторую зону, которая покроет всю не политую морковь.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water-1 {  
8   grid-area: 1 / 4 / 6 / 5;  
9 }  
10  
11 #water-2 {  
12   grid-area: 2/3/5/6;  
13 }  
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 18 из 28 ▶

Если grid-элементы не имеют конкретного расположения с `grid-area`, `grid-column`, `grid-row` и т.д., они автоматически размещаются, следуя порядку, написанному в коде. Мы можем изменить это с помощью свойства `order`, которое является одним из преимуществ CSS Grid Layout перед табличной разметкой.

Изначально все grid-элементы имеют `order`, равный `0`, но этому свойству можно присвоить любое положительное или отрицательное значение, примерно как у `z-index`.

Сейчас морковь во втором столбце отправлена и сорняки в последнем столбце поливаются. Измените значение свойства `order` для яда, чтобы исправить это прямо сейчас!

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 .water {  
8   order: 0;  
9 }  
10  
11 #poison {  
12   order: 5;  
13 }  
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 19 из 28 ▶

Теперь вода и яд меняются местами, хотя все сорняки находятся в начале сада. Присвойте правильное значение `order` для яда, чтобы исправить это.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 .water {  
8   order: 0;  
9 }  
10  
11 .poison {  
12   order: -1;|  
13 }  
14
```

[Следующий](#)

Игра Grid Garden создана Codeipip • GitHub • Twitter • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 20 из 28 ▶

До этого момента сад имел grid-сетку с пятью столбцами по 20% ширины и пятью строками, каждая по 20% высоты.

Это было сделано свойствами `grid-template-columns: 20% 20% 20% 20% 20%`; и `grid-template-rows: 20% 20% 20% 20% 20%`; Каждое свойство имеет пять значений, которые создают пять столбцов, где ширина каждого равна 20% от общей ширины сада.

Но вы можете изменить grid-сетку как вам вздумается. Присвойте `grid-template-columns` новое значение, чтобы полить морковь. Вам необходимо поставить значение ширины первого столбца равным 50%.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 50%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column: 1;  
9   grid-row: 1;  
10 }  
11  
12  
13  
14
```

[Следующий](#)

Игра Grid Garden создана Codeipip • GitHub • Twitter • [Русский](#)

GRID GARDEN

◀ Уровень 21 из 28 ▶

Определять несколько столбцов с одинаковой шириной может быть утомительно. К счастью, есть функция `repeat`, которая призвана помочь в этом.

Например, раньше мы определяли пять столбцов по 20% ширины с помощью `grid-template-columns: 20% 20% 20% 20% 20%`; Это можно упростить до `grid-template-columns: repeat(5, 20%)`.

Используй `grid-template-columns` вместе с функцией `repeat`, создайте 8 столбцов по 12.5% ширины каждый. Таким образом, вы не будете использовать больше воды, чем нужно.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns : repeat(8, 12.5%);  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-column: 1;  
9   grid-row: 1;  
10 }  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана [Codepiper](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

GRID GARDEN

◀ Уровень 22 из 28 ▶

`grid-template-columns` принимает значения не только в процентах, но и в единицах длины, например `px` или `em`. Вы даже можете комбинировать разные единицы измерения.

Присвойте трем столбцам значение `100px`, `3em` и `40%` соответственно.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 100px 3em 40%;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7  
8  
9  
10  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана [Codepiper](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 23 из 28 ▶

CSS Grid Layout также вводит новую единицу измерения: дробный `fr` (англ. *fraction*). Каждый `fr` выделяет одну часть свободного пространства. Например, если двум элементам задано `1fr` и `3fr` соответственно, то пространство будет поделено на 4 одинаковые части. Первый элемент займет $\frac{1}{4}$, а второй оставшиеся $\frac{3}{4}$ пространства.

Сейчас сорняки занимают левую $\frac{1}{6}$ часть вашей первой строки, а морковь оставшиеся $\frac{5}{6}$. Создайте два столбца с такой же шириной, используя `fr`.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 1fr 5fr;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7  
8  
9  
10  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 24 из 28 ▶

Когда одни столбцы определены с помощью пикселей, процентов или em, а любые другие столбцы – с помощью `fr`, то последние просто разделят всё возможное оставшееся пространство.

Сейчас морковь занимает 50 пикселей слева, а сорняки 50 пикселей справа. С помощью `grid-template-columns` создайте два столбца и используйте `fr`, чтобы сделать ещё 3 столбца, которые займут оставшееся пространство между ними.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 50px 1fr 1fr 1fr 50px;  
4   grid-template-rows: 20% 20% 20% 20% 20%;  
5 }  
6  
7 #water {  
8   grid-area: 1 / 1 / 6 / 2;  
9 }  
10  
11 #poison {  
12   grid-area: 1 / 5 / 6 / 6;  
13 }
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 25 из 28 ▶

Теперь у нас есть столбец сорняков с шириной 75 пикселей на левой стороне сада. Оставшегося пространства занимает растущая морковь, а заполонили сорняки.

Используйте `grid-template-columns` с комбинацией `px` и `fr`, чтобы создать необходимые столбцы.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 75px 3fr 2fr;  
4   grid-template-rows: 100%;  
5 }  
6  
7  
8  
9  
10  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 26 из 28 ▶

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

Используйте `grid-template-rows`, чтобы полить всё, кроме верхних 50 пикселей сада. Помните, что вода заполняет только пятую строку, так что вам нужно создать именно пять строк.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20% 20% 20%;  
4   grid-template-rows: 50px 0 0 0 1fr;  
5 }  
6  
7 #water {  
8   grid-column: 1 / 6;  
9   grid-row: 5 / 6;  
10 }  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 27 из 28 ▶

`grid-template` – сокращённый вариант комбинации `grid-template-rows` и `grid-template-columns`.

Например, `grid-template: 50% 50% / 200px` создаст grid-сетку с двумя строками по 50% каждая и одним столбцом шириной 200 пикселей.

Попробуйте использовать `grid-template`, чтобы полить зону, включающую в себя верхние 60% и левые 200 пикселей сада.

```
1 #garden {  
2   display: grid;  
3   grid-template: 60% 40%/ 200px;  
4 }  
5  
6 #water {  
7   grid-column: 1;  
8   grid-row: 1;  
9 }  
10  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

GRID GARDEN

◀ Уровень 28 из 28 ▶

Сад выглядит прекрасно. Здесь вы оставили 50-пиксельную дорожку снизу и заполнили всё оставшееся пространство морковью.

К сожалению, левые 20% сада заполнили сорняки. Используйте CSS Grid Layout в последний раз, чтобы очистить сад.

```
1 #garden {  
2   display: grid;  
3   grid-template: 1fr 50px / 1fr 4fr;  
4 }  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14
```

Следующий

Игра Grid Garden создана [Codepen](#) • [GitHub](#) • [Twitter](#) • [Русский](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

Flexbox Froggy

FLEXBOX FROGGY

Добро пожаловать в Flexbox Froggy. Игра, в которой тебе нужно помочь лягушонку Фроги и его друзьям написанием CSS кода! Направь этого лягушонка на лилию справа используя свойство `justify-content`, которое выравнивает элементы горизонтально 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.

Например, `justify-content: flex-end;` сдвинет лягушонка вправо.

```
1 #pond {  
2   display: flex;  
3   justify-content: flex-end;  
4 }  
5  
6  
7  
8  
9  
10
```

[Следующий](#)

Игра Flexbox Froggy создана [Codepiper](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

Want to learn CSS grid? Play [Grid Garden](#).

FLEXBOX FROGGY

Уровень 2 из 24 ▾

Используй **justify-content** еще раз, чтобы помочь этим лягушатам попасть на их лилии. Помни, что это свойство CSS выравнивает элементы горизонтально 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.

```
1 #pond {  
2   display: flex;  
3   justify-content:center;  
4 }  
5  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 3 из 24 ▾

Помоги всем трем лягушатам найти их лилии, просто используя justify-content. В этот раз, у лилий много пространства вокруг.

Если ты чувствуешь, что забыл возможные значения свойства, ты можешь навести курсор на название свойства, чтобы посмотреть их. Попробуй навести курсор на justify-content.

```
1 #pond {  
2   display: flex;  
3   justify-content: space-around;  
4 }  
5  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 4 из 24 ▾

Теперь лилии по краям уплыли к берегам, увеличив пространство между ними. Используй justify-content. В этот раз, у лилий одинаковое расстояние между ними.

```
1 #pond {  
2   display: flex;  
3   justify-content: space-between;  
4 }  
5  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 5 из 24

Теперь используй **align-items** чтобы помочь лягушатам добраться к нижней части пруда. Это CSS свойство выравнивает элементы вертикально 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.

```
1 #pond {  
2     display: flex;  
3     align-items: flex-end;  
4 }  
5  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 6 из 24 ▾

Направь лягушонка в центр пруда, используя **justify-content** и **align-items** вместе.

```
1 #pond {  
2   display: flex;  
3   justify-content: center;  
4   align-items: center;  
5 }  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 7 из 24 ▾

Лягушатам снова нужно пересечь пруд. В этот раз к лилиям, с достаточно большим пространством вокруг них. Используй комбинацию **justify-content** и **align-items**.

```
1 #pond {  
2   display: flex;  
3   justify-content: space-around;  
4   align-items: flex-end;|  
5 }  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 8 из 24 ▾

Лягушатам нужно выстроиться в порядке их лилий, используя **flex-direction**. Это CSS свойство задает направление, в котором будут расположены элементы в контейнере, 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.

```
1 #pond {  
2     display: flex;  
3     flex-direction: row-reverse;● ▲ ▼  
4 }  
5  
6  
7  
8  
9  
10
```

[Следующий](#)

FLEXBOX FROGGY

Уровень 9 из 24 ▾

Помоги лягушатам расположиться на своих лилиях используя **flex-direction**. Это CSS свойство задает направление, в котором будут расположены элементы в контейнере 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.

```
1 #pond {  
2   display: flex;  
3   flex-direction: column;|  
4 }  
5  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY



Уровень 10 из 24



Помоги лягушатам попасть на свои лилии. Хоть и кажется, что они почти на своих местах, все же придется применить и **flex-direction** и **justify-content**, чтобы поместить их на свои лилии.

Заметь, что когда ты устанавливаешь направление в обратном порядке для ряда или колонки, начало (start) и конец (end) тоже меняются местами.

```
1 #pond {  
2   display: flex;  
3   flex-direction: row-reverse;  
4   justify-content: flex-end;  
5 }  
6  
7  
8  
9  
10
```



Следующий

FLEXBOX FROGGY

◀ Уровень 11 из 24 ▶

Помоги лягушатам найти их лилии с помощью **flex-direction** и **justify-content**.

Заметь, когда в качестве направления выбрана колонка, то **justify-content** влияет на вертикальное выравнивание, а **align-items** на горизонтальное.

```
1 #pond {  
2   display: flex;  
3   flex-direction: column;  
4   justify-content: flex-end;|  
5 }  
6  
7  
8  
9  
10
```



Следующий

FLEXBOX FROGGY

◀ Уровень 12 из 24 ▶

Помоги лягушатам найти их лилии с помощью **flex-direction** и **justify-content**.

```
1 #pond {  
2   display: flex;  
3   flex-direction: column-reverse;  
4   justify-content: space-between;|  
5 }  
6  
7  
8  
9  
10
```



Следующий

FLEXBOX FROGGY

Уровень 13 из 24

Помоги лягушатам найти их лилии с помощью **flex-direction**, **justify-content** и **align-items**.

```
1 #pond {  
2   display: flex;  
3   flex-direction: row-reverse;  
4   justify-content: center;  
5   align-items: flex-end;  
6 }  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 14 из 24

Иногда изменения порядка отображения элементов в контейнере недостаточно. В таких случаях мы можем применить свойство **order** для конкретных элементов. По умолчанию, значение этого свойства у элементов равно 0, но мы можем задать положительное или отрицательное целое число этому свойству.

Используй свойство **order**, чтобы разместить лягушат на своих лилиях.

```
1 #pond {  
2   display: flex;  
3 }  
4  
5 .yellow {  
6   order: 2;|  
7 }  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

◀ Уровень 15 из 24 ▶

Используй свойство order, чтобы отправить красного лягушонка на его лилию.

```
1 #pond {  
2   display: flex;  
3 }  
4  
5 .red {  
6   order:-2;  
7 }  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

◀ Уровень 16 из 24 ▶

Еще одно свойство, которое ты можешь применить к определенному элементу это align-self. Это свойство принимает те же значения, что и align-items.

```
1 #pond {  
2   display: flex;  
3   align-items: flex-start;  
4 }  
5  
6 .yellow {  
7   align-self: flex-end;|  
8 }  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 17 из 24

Используй `order` и `align-self` вместе, чтобы помочь лягушатам добраться к своим целям.

```
1 #pond {  
2   display: flex;  
3   align-items: flex-start;  
4 }  
5  
6 .yellow {  
7   order: 2;  
8   align-self: flex-end;  
9 }  
10
```

Следующий

FLEXBOX FROGGY

◀ Уровень 18 из 24 ▶

О нет! Лягушат сплющило на одном ряду лилий. Раздвинь их с помощью свойства **flex-wrap**, которое 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.

```
1 #pond {  
2   display: flex;  
3   flex-wrap: wrap;|  
4 }  
5  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

◀ Уровень 19 из 24 ▶

Помоги этой армии лягушат выстроиться в три колонки с помощью комбинации **flex-direction** и **flex-wrap**.

```
1 #pond {  
2   display: flex;  
3   flex-direction: column;  
4   flex-wrap: wrap;  
5 }  
6  
7  
8  
9  
10
```



Следующий

FLEXBOX FROGGY

◀ Уровень 20 из 24 ▶

Два свойства **flex-direction** и **flex-wrap** используются так часто вместе, что было создано свойство **flex-flow** для их комбинирования. Это свойство принимает значения двух этих свойств, разделенные пробелом.

Например, ты можешь использовать **flex-flow: row wrap**, чтобы элементы располагались в ряд и автоматически переносились на новую строку.

Попробуй использовать **flex-flow**, чтобы повторить предыдущий уровень.

```
1 #pond {  
2   display: flex;  
3   flex-flow: column wrap;  
4 }  
5  
6  
7  
8  
9  
10
```



Следующий

FLEXBOX FROGGY

Уровень 21 из 24

Лягушат раскидало по всему пруду, но лилии сгруппированы в верхней части. Ты можешь использовать **align-content**, чтобы указать, как несколько рядов должны отделяться друг от друга. Данное свойство 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.

Это может запутать, но **align-content** отвечает за расстояние между рядами, в то время как **align-items** отвечает за то, как элементы в целом будут выровнены в контейнере. Когда только один ряд, **align-content** ни на что не влияет.

```
1 #pond {  
2   display: flex;  
3   flex-wrap: wrap;  
4   align-content: flex-start;  
5 }  
6  
7
```

FLEXBOX FROGGY

Уровень 22 из 24

Теперь течение сгруппировало лилии в нижней части. Используй align-content, чтобы направить лягушат туда.

```
1 #pond {  
2   display: flex;  
3   flex-wrap: wrap;  
4   align-content: flex-end;|  
5 }  
6  
7  
8  
9  
10
```

Следующий

FLEXBOX FROGGY

Уровень 23 из 24 ▾

Лягушата были на вечеринке, но уже пора возвращаться домой. Используй комбинацию **flex-direction** и **align-content**, чтобы отправить их к своим лилиям.

```
1 #pond {  
2   display: flex;  
3   flex-wrap: wrap;  
4   flex-direction: column-reverse;  
5   align-content: center;  
6 }  
7  
8  
9  
10
```



Следующий

FLEXBOX FROGGY



Уровень 24 из 24



Доставь лягушат по домам в последний раз, используя CSS свойства, которые ты выучил:

- justify-content
- align-items
- flex-direction
- order
- align-self
- flex-wrap
- flex-flow
- align-content

```
1 #pond {  
2   display: flex;  
3   flex-flow: column-reverse wrap-reverse;  
4   align-content: space-between;  
5   justify-content: center;  
6 }  
7  
8  
9  
10
```



Следующий

Вывод

Grid и Flexbox - два современных стандарта для идеальной CSS-вёрстки. Flexbox чаще работает в одном измерении, а Grid задействует сразу два - ряды и колонки. Благодаря этим играм я убедился в удобстве данных технологий.