

## Level 1

`justify-content` property, which aligns items horizontally and accepts the following values:


- `flex-start`: Items align to the left side of the container.
- `flex-end`: Items align to the right side of the container.
- `center`: Items align at the center of the container.
- `space-between`: Items display with equal spacing between them.
- `space-around`: Items display with equal spacing around them.

For example, `justify-content: flex-end;` will move the frog to the right.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)



## Level 2


**FLEXBOX FROGGY** ◀ Level 2 of 24 ▶

Use `justify-content` again to help these frogs get to their lilypads. Remember that this CSS property aligns items horizontally and accepts the following values:

- `flex-start`: Items align to the left side of the container.
- `flex-end`: Items align to the right side of the container.
- `center`: Items align at the center of the container.
- `space-between`: Items display with equal spacing between them.
- `space-around`: Items display with equal spacing around them.

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

Next



## Level 3

## FLEXBOX FROGGY

Level 3 of 24

Help all three frogs find their lilypads just by using `justify-content`. This time, the lilypads have lots of space all around them.




If you find yourself forgetting the possible values for a property, you can hover over the property name to view them. Try hovering over `justify-content`.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



### Level 4

## FLEXBOX FROGGY

Level 4 of 24




Now the lilypads on the edges have drifted to the shore, increasing the space between them. Use `justify-content`. This time, the lilypads have equal spacing between them.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



### Level 5

## FLEXBOX FROGGY


Level 5 of 24

Now use `align-items` to help the frogs get to the bottom of the pond.  
This CSS property aligns items vertically and accepts the following values:

- `flex-start`: Items align to the top of the container.
- `flex-end`: Items align to the bottom of the container.
- `center`: Items align at the vertical center of the container.
- `baseline`: Items display at the baseline of the container.
- `stretch`: Items are stretched to fit the container.

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

Next



### Level 6


## FLEXBOX FROGGY

Level 6 of 24

Lead the frog to the center of the pond using a combination of `justify-content` and `align-items`.

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

Next



Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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

### Level 7

FLEXBOX FROGGY

Level 7 of 24


The frogs need to cross the pond again, this time for some lilypads with plenty of space around them. Use a combination of `justify-content` and `align-items`.

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

Next

Flexbox Froggy is created by [CodePen](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



## Level 8

FLEXBOX FROGGY


Level 8 of 24

The frogs need to get in the same order as their lilypads using `flex-direction`. This CSS property defines the direction items are placed in the container, and accepts the following values:

- `row`: Items are placed the same as the text direction.
- `row-reverse`: Items are placed opposite to the text direction.
- `column`: Items are placed top to bottom.
- `column-reverse`: Items are placed bottom to top.

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

Next



## Level 9

## FLEXBOX FROGGY


Level 9 of 24

Help the frogs find their column of lilypads using `flex-direction`. This CSS property defines the direction items are placed in the container, and accepts the following values:

- `row`: Items are placed the same as the text direction.
- `row-reverse`: Items are placed opposite to the text direction.
- `column`: Items are placed top to bottom.
- `column-reverse`: Items are placed bottom to top.

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

Next



### Level 10

## FLEXBOX FROGGY


Level 10 of 24

Help the frogs get to their own lilypads. Although they seem close, it will take both `flex-direction` and `justify-content` to get them there.

Notice that when you set the direction to a reversed row or column, start and end are also reversed.

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

Next



Flexbox Froggy is created by [CodePip](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

[Want to learn CSS and HTML? Play Grid Garden](#)

### Level 11

## FLEXBOX FROGGY

◀Level 11 of 24▶

Help the frogs find their lilypads using `flex-direction` and `justify-content`.


Notice that when the flex direction is a column, `justify-content` changes to the vertical and `align-items` to the horizontal.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



## Level 12

## FLEXBOX FROGGY

◀Level 12 of 24▶


Help the frogs find their lilypads using `flex-direction` and `justify-content`.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



## Level 13

## FLEXBOX FROGGY

◀Level 13 of 24▶


Help the frogs find their lilypads using `flex-direction`, `justify-content`, and `align-items`.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



### Level 14

## FLEXBOX FROGGY

◀Level 14 of 24▶

Sometimes reversing the row or column order of a container is not enough. In these cases, we can apply the `order` property to individual items. By default, items have a value of 0, but we can use this property to also set it to a positive or negative integer value (-2, -1, 0, 1, 2).


Use the `order` property to reorder the frogs according to their lilypads.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



### Level 15

## FLEXBOX FROGGY

Level 15 of 24


Use the `order` property to send the red frog to his lilypad.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



### Level 16

## FLEXBOX FROGGY

Level 16 of 24


Another property you can apply to individual items is `align-self`. This property accepts the same values as `align-items` and its value for the specific item.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



### Level 17



## FLEXBOX FROGGY

Level 17 of 24



Combine `order` with `align-self` to help the frogs to their destinations.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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



### Level 18

## FLEXBOX FROGGY

Level 18 of 24



Oh no! The frogs are all squeezed onto a single row of lily pads. Spread them out using the `flex-wrap` property, which accepts the following values:

- `nowrap`: Every item is fit to a single line.
- `wrap`: Items wrap around to additional lines.
- `wrap-reverse`: Items wrap around to additional lines in reverse.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)



### Level 19

## FLEXBOX FROGGY

Level 19 of 24

Help this army of frogs form three orderly columns using a combination of `flex-direction` and `flex-wrap`.

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

Next

Flexbox Froggy is created by [Codecademy](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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

### Level 20

## FLEXBOX FROGGY

Level 20 of 24

The two properties `flex-direction` and `flex-wrap` are used so often together that the shorthand property `flex-flow` was created to combine them. This shorthand property accepts the value of the two properties separated by a space.

For example, you can use `flex-flow: row wrap` to set rows and wrap them.

Try using `flex-flow` to repeat the previous level.

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

Next

### Level 21

- **flex-start**: Lines are packed at the top of the container.
- **flex-end**: Lines are packed at the bottom of the container.
- **center**: Lines are packed at the vertical center of the container.
- **space-between**: Lines display with equal spacing between them.
- **space-around**: Lines display with equal spacing around them.
- **stretch**: Lines are stretched to fit the container.

This can be confusing, but **align-content** determines the spacing between lines, while **align-items** determines how the items as a whole are aligned within the container. When there is only one line, **align-content** has no effect.

```

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

```

Next



## Level 22

### FLEXBOX FROGGY

Level 22 of 24

Now the current has bunched the lilypads at the bottom. Use **align-content** to guide the frogs there.

```

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

```

Next



## Level 23

## FLEXBOX FROGGY

Level 23 of 24

The frogs have had a party, but it is time to go home. Use a combination of `flex-direction` and `align-content` to get them to their lilypads.

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

Next

Flexbox Froggy is created by [CodePia](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

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

## Level 24

Bring the frogs home one last time by using the CSS properties you've learned:

- `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
```

Next