UNIVERSIDAD PRIVADA
DOMINGO SAVIO

DESAFÍO COMPLEMENTARIO

Estudiantes: Carlos Miguel Colque Montoya

Docente: Ing. Angela Graciela Rosales Ugarte

Modulo: Diseño Web I

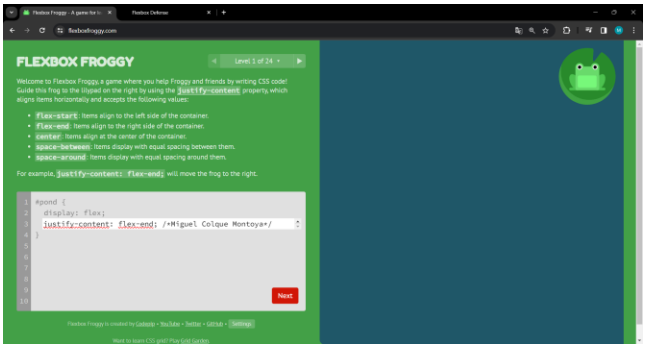
Carrera: Ingeniería de Sistemas

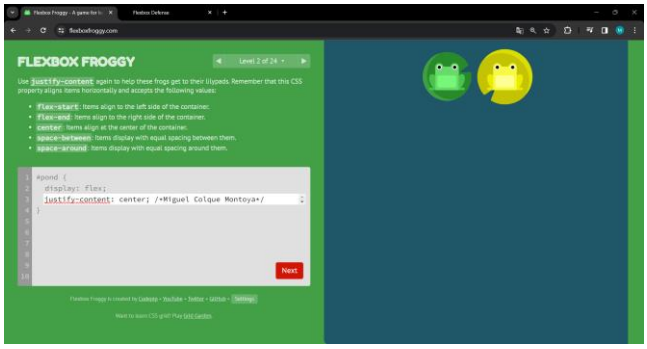
Año: 2024

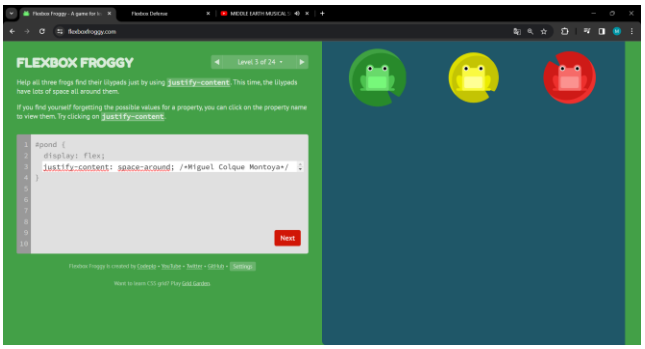
Sucre – Bolivia

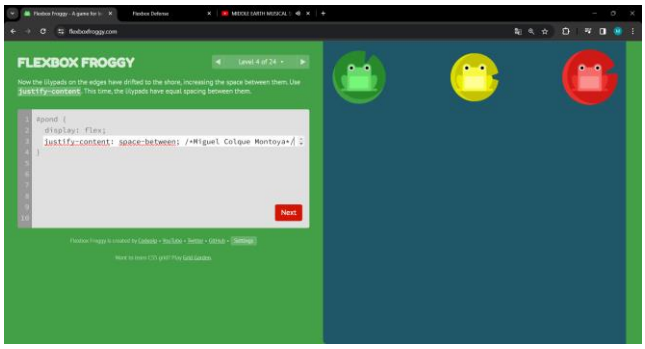
Desafío complementario

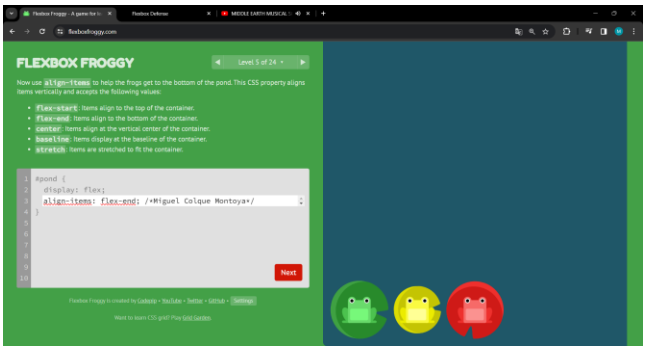
Flexbox Froggy

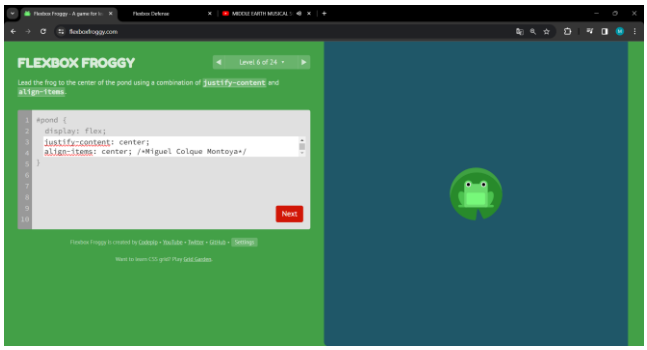


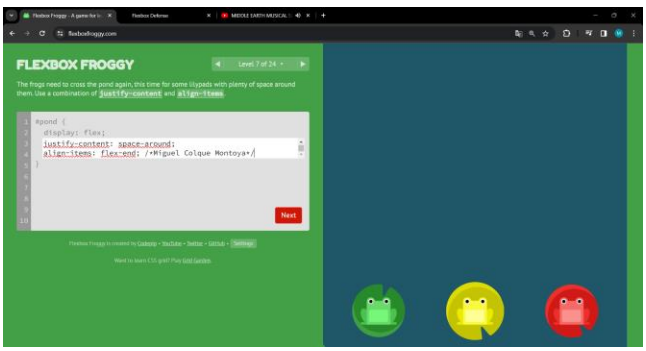


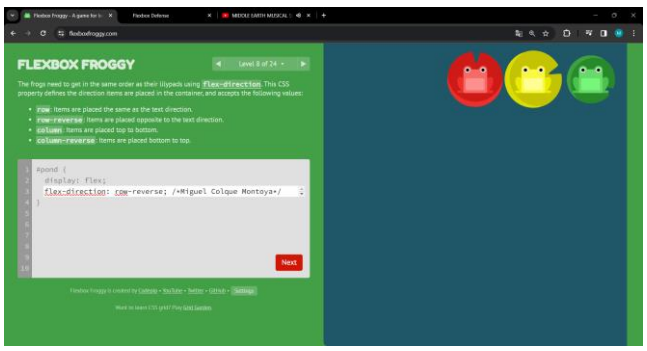


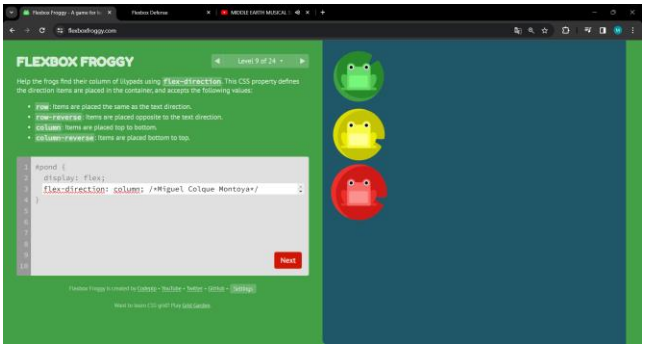


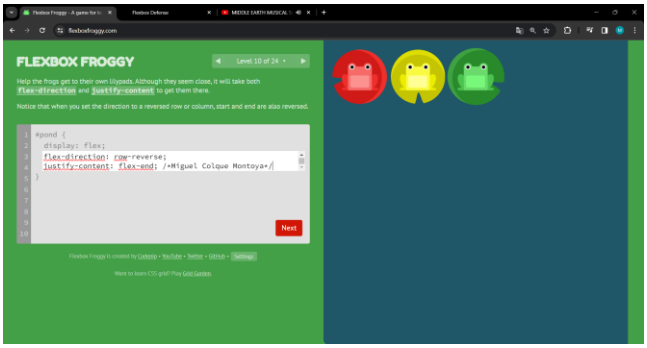


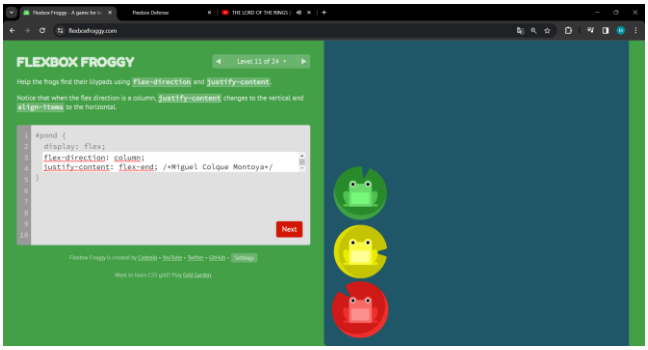


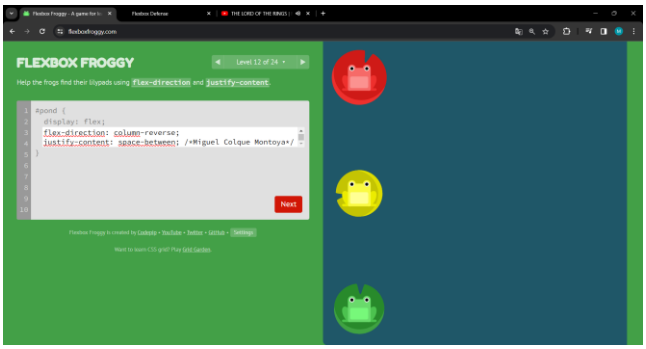


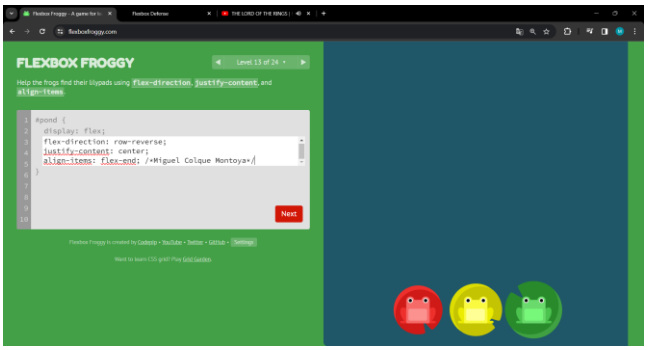


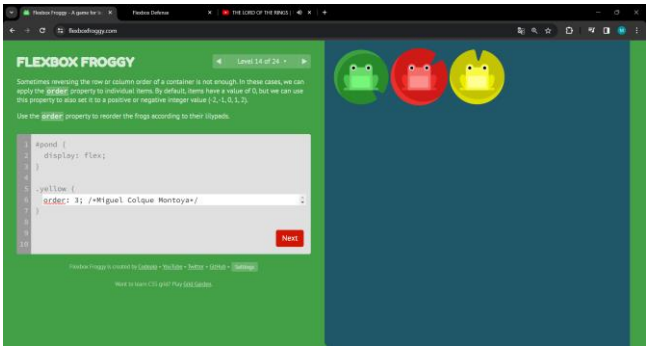

FLEXBOX FROGGY
Level 9 of 24
Help the frogs find their column of lilypads using flex-direction. This CSS property defines the direction items are placed in the container, and accepts the following values:
row: Items are placed the same as the text direction.
row-reverse: Items are placed opposite to the text direction.
column: Items are placed top to bottom.
column-reverse: Items are placed bottom to top.
#pond {
display: flex;
flex-direction: column; /*Miguel Colque Montoya*/
}
Next

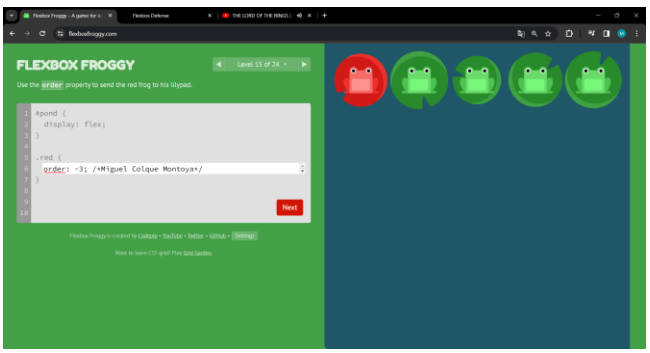

FLEXBOX FROGGY
Level 10 of 24
Help the frogs get to their own lilypads. Although they seem close, it will take both flex-direction and justify-content to get them there.
Notice that when you set the direction to a reversed row or column, start and end are also reversed.
#pond {
display: flex;
flex-direction: row-reverse;
justify-content: flex-end; /*Miguel Colque Montoya*/
}
Next

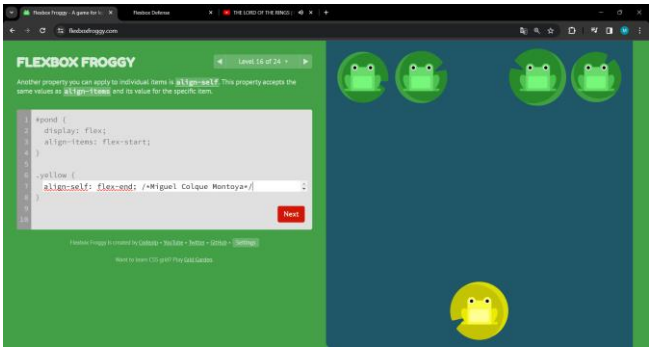

FLEXBOX FROGGY
Level 11 of 24
Help the frogs find their lilypads using flex-direction and justify-content.
Notice that when the flex direction is a column, justify-content changes to the vertical and align-items to the horizontal.
#pond {
display: flex;
flex-direction: column;
justify-content: flex-end; /*Miguel Colque Montoya*/
}
Next

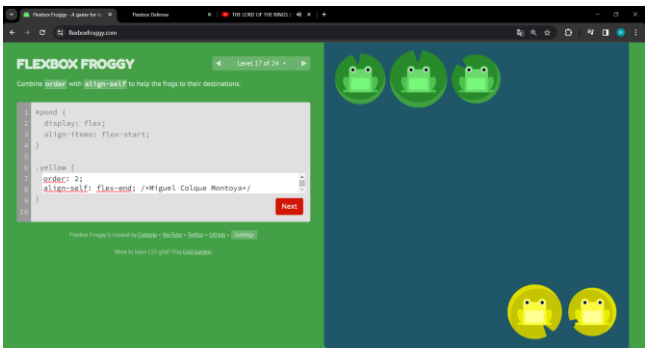

FLEXBOX FROGGY
Level 12 of 24
Help the frogs find their lilypads using flex-direction and justify-content.
#pond {
display: flex;
flex-direction: column-reverse;
justify-content: space-between; /*Miguel Colque Montoya*/
}
Next

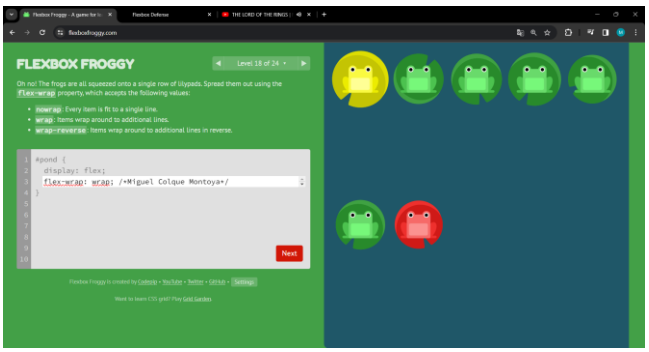

FLEXBOX FROGGY
Level 13 of 24
Help the frogs find their lilypads using flex-direction, justify-content, and align-items.
#pond {
display: flex;
flex-direction: row-reverse;
justify-content: center;
align-items: flex-end; /*Miguel Colque Montoya*/
}
Next

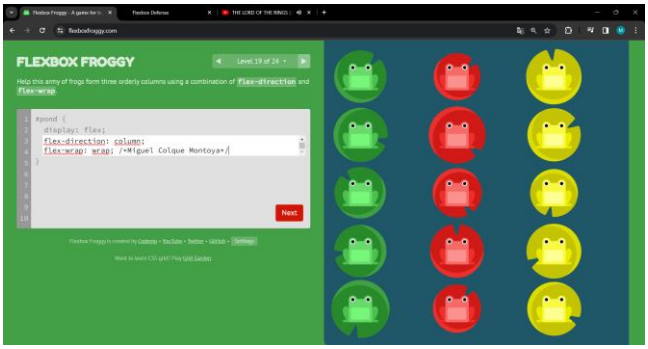

FLEXBOX FROGGY
Level 14 of 24
Sometimes reversing the row or column order of a container is not enough. In these cases, we can apply the order property to individual items. By default, items have a value of 0, but we can use this property to also set it to a positive or negative integer value (-2, -1, 0, 1, 2).
Use the order property to reorder the frogs according to their lilypads.
#pond {
display: flex;
}
.yellow {
order: 3; /*Miguel Colque Montoya*/
}
Next

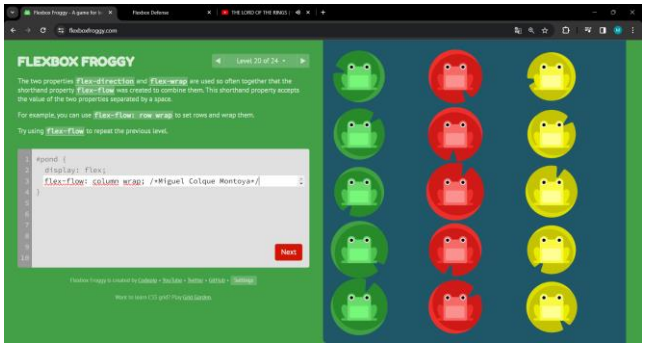

FLEXBOX FROGGY
Level 15 of 24
Use the order property to send the red frog to his lilypad.
#pond {
display: flex;
}
.red {
order: -3; /*Miguel Colque Montoya*/
}
Next

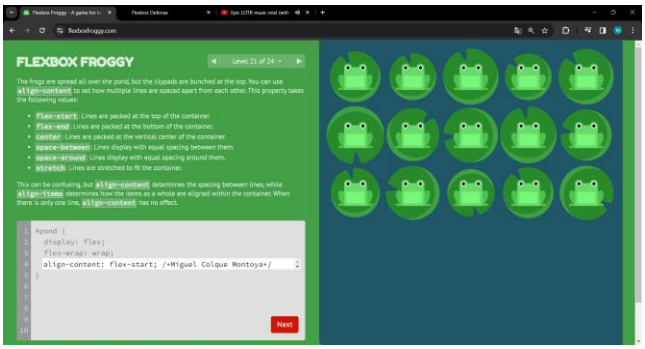

FLEXBOX FROGGY
Level 16 of 24
Another property you can apply to individual items is align-self. This property accepts the same values as align-items and its value for the specific item.
#pond {
display: flex;
align-items: flex-start;
}
.yellow {
align-self: flex-end; /*Miguel Colque Montoya*/
}
Next

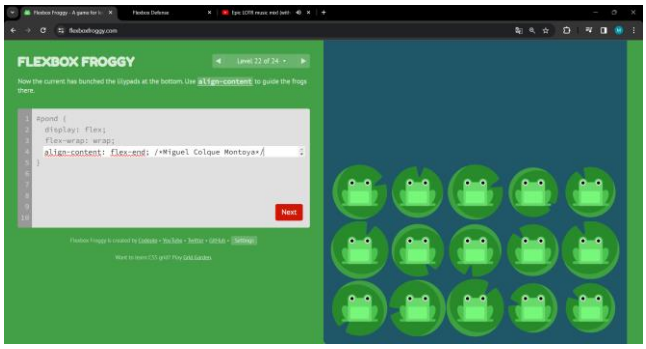

FLEXBOX FROGGY
Level 17 of 24
Combine order with align-self to help the frogs to their destinations.
#pond {
display: flex;
align-items: flex-start;
}
.yellow {
order: 2;
align-self: flex-end; /*Miguel Colque Montoya*/
}
Next

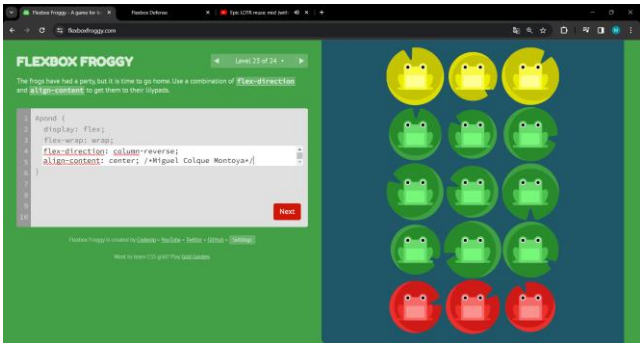

FLEXBOX FROGGY
Level 18 of 24
Oh no! The frogs are all squeezed onto a single row of lilypads. Spread them out using the flex-wrap property, which accepts the following values:
nowrap: Every item is fit to a single line.
wrap: Items wrap around to additional lines.
wrap-reverse: Items wrap around to additional lines in reverse.
#pond {
display: flex;
flex-wrap: wrap; /*Miguel Colque Montoya*/
}
Next

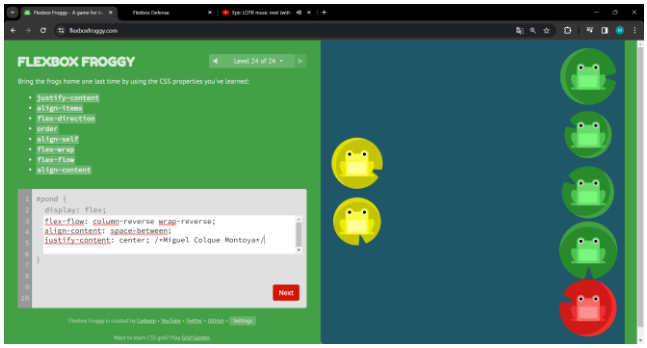

FLEXBOX FROGGY
Level 19 of 24
Help this army of frogs form three orderly columns using a combination of flex-direction and flex-wrap.
#pond {
display: flex;
flex-direction: column;
flex-wrap: wrap; /*Miguel Colque Montoya*/
}
Next

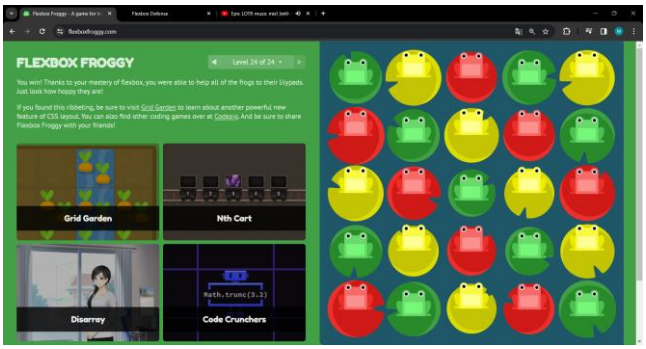

FLEXBOX FROGGY
Level 20 of 24
The two properties flex-direction and flex-wrap are used so often together that the shorthand property flex-flow was created to combine them. This shorthand property accepts the value of the two properties separated by a space.
For example, you can use flex-flow: row wrap to set rows and wrap them.
Try using flex-flow to repeat the previous level.
#pond {
display: flex;
flex-flow: column wrap; /*Miguel Colque Montoya*/
}
Next


FLEXBOX FROGGY
Level 21 of 24
The frogs are spread all over the pond, but the lilypads are bunched at the top. You can use align-content to set how multiple lines are spaced apart from each other. This property takes the following values:
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.
#pond {
display: flex;
flex-wrap: wrap;
align-content: flex-start; /*Miguel Colque Montoya*/
}
Next

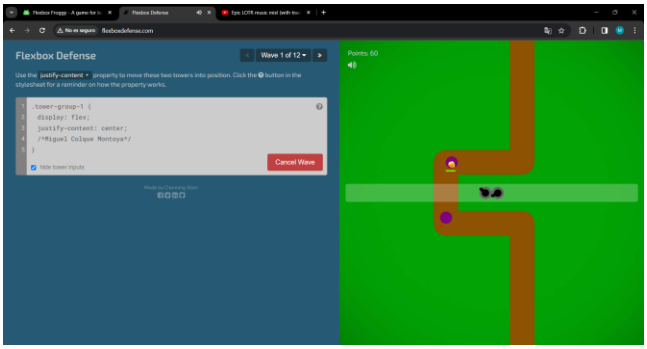

FLEXBOX FROGGY
Level 22 of 24
Now the current has bunched the lilypads at the bottom. Use align-content to guide the frogs there.
#pond {
display: flex;
flex-wrap: wrap;
align-content: flex-end; /*Miguel Colque Montoya*/
}
Next

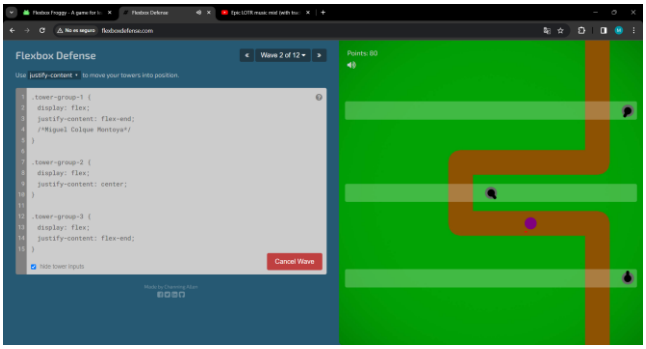

FLEXBOX FROGGY
Level 23 of 24
The frogs have had a party, but it is time to go home. Use a combination of flex-direction and align-content to get them to their lilypads.
#pond {
display: flex;
flex-wrap: wrap;
flex-direction: column-reverse;
align-content: center; /*Miguel Colque Montoya*/
}
Next

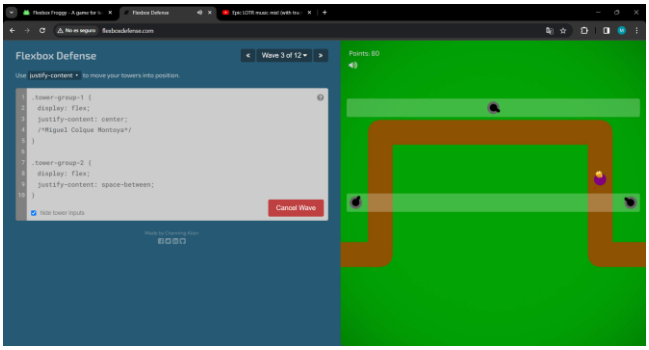

FLEXBOX FROGGY
Level 24 of 24
Bring the frogs home one last time by using the CSS properties you've learned:
justify-content
align-items
flex-direction
order
align-self
flex-wrap
flex-flow
align-content
#pond {
display: flex;
flex-flow: column-reverse wrap-reverse;
align-content: space-between;
justify-content: center; /*Miguel Colque Montoya*/
}
Next

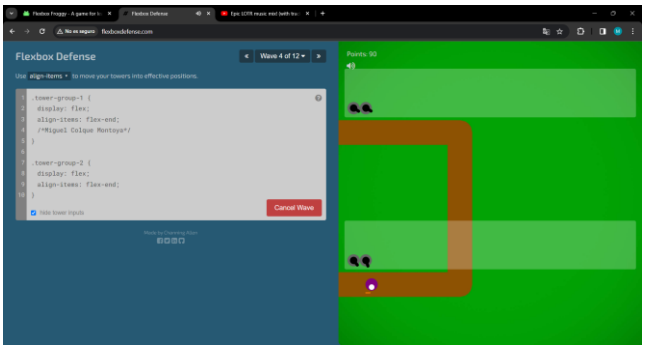

FLEXBOX FROGGY
Level 24 of 24
You win! Thanks to your mastery of flexbox, you were able to help all of the frogs to their lilypads. Just look how happy they are!
If you found this ribbeting, be sure to visit Grid Garden to learn about another powerful, new feature of CSS layout. You can also find other coding games over at Codepip. And be sure to share Flexbox Froggy with your friends!
Grid Garden
Nth Cart
Disarray
Math.trunc(3.2)
Code Crunchers

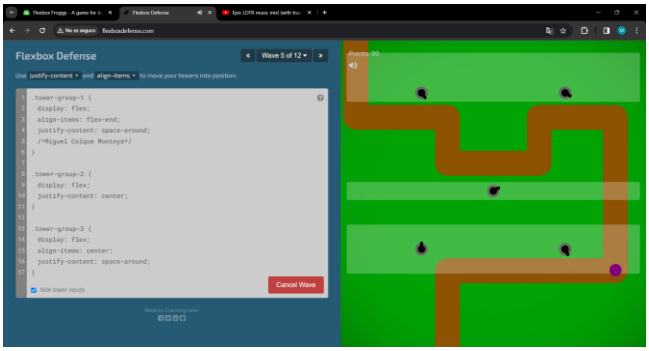
Flexbox Defense

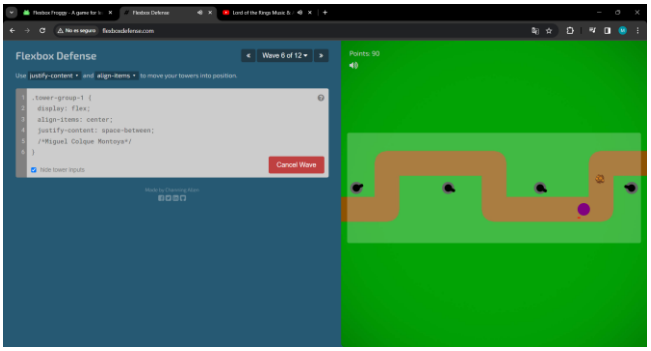


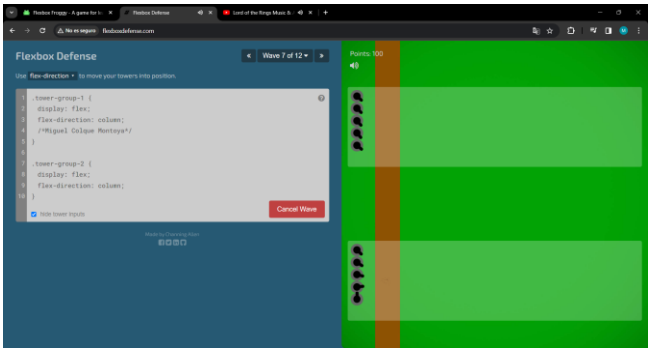


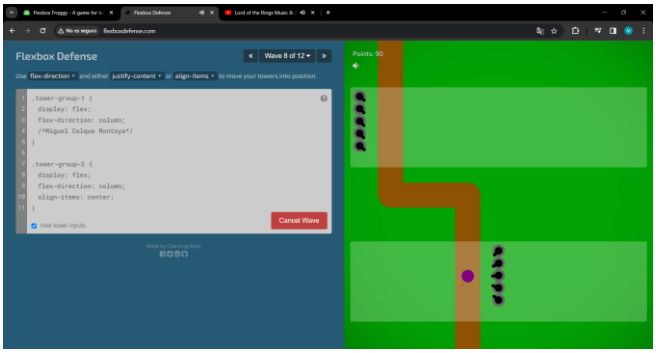


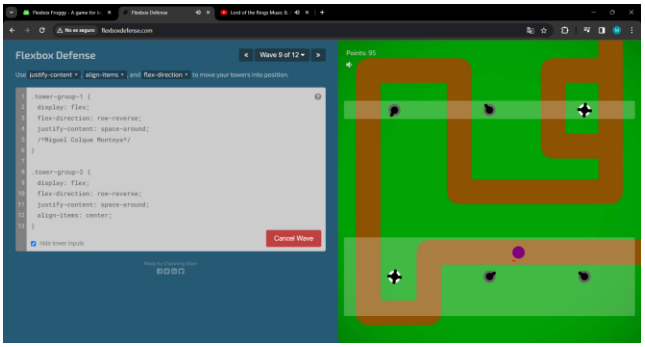


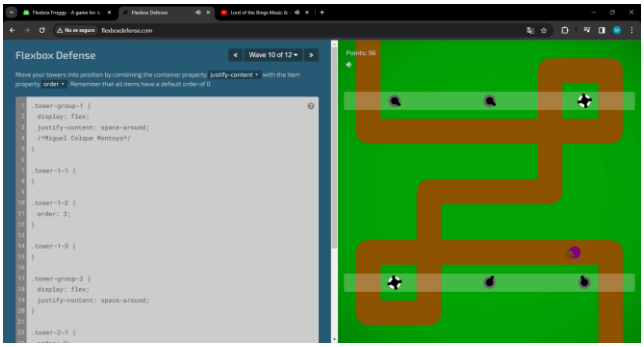


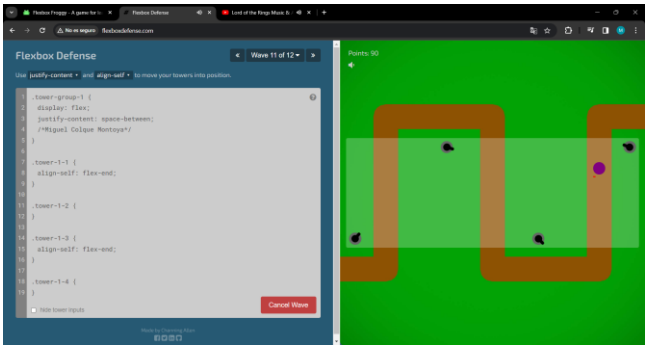


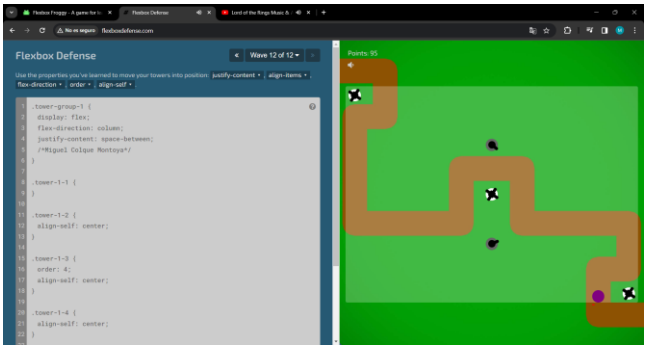


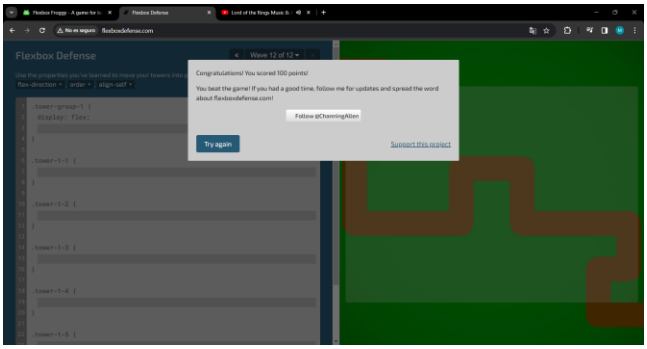



Flexbox Defense
Wave 9 of 12
Points: 95
Cancel Wave

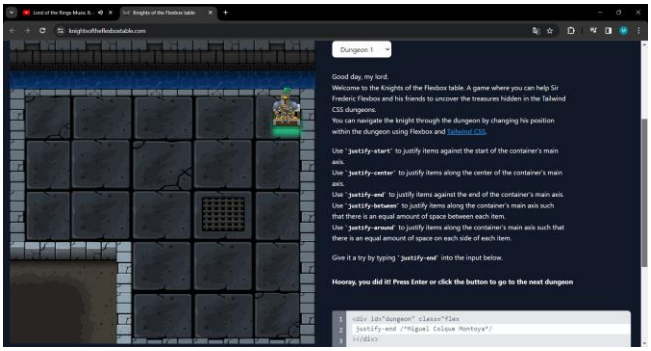

Flexbox Defense
Wave 10 of 12
Points: 95

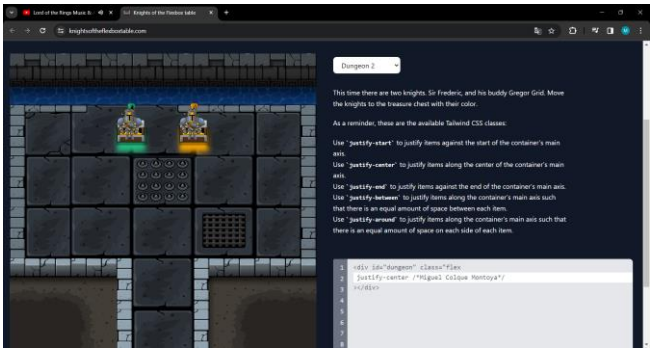

Flexbox Defense
Wave 11 of 12
Points: 90
Cancel Wave

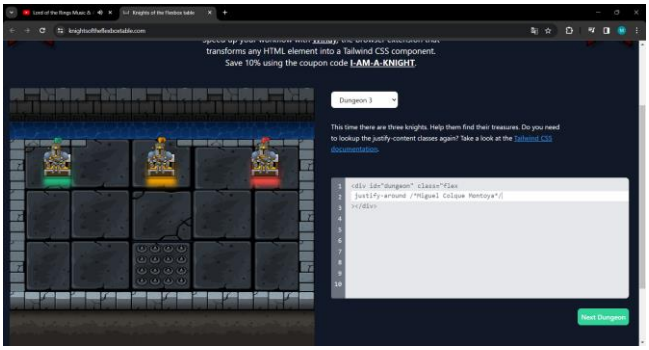

Flexbox Defense
Wave 12 of 12
Points: 95

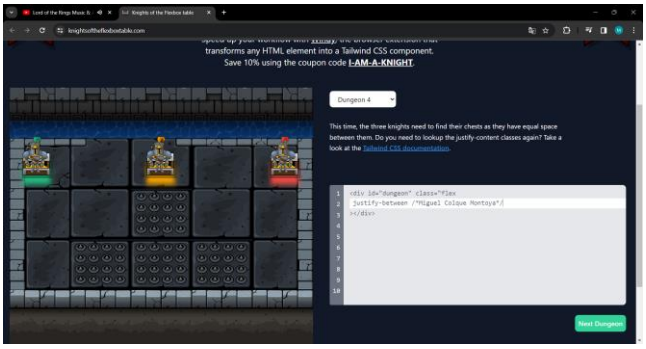

Flexbox Defense
Wave 12 of 12
Congratulations! You scored 100 points!
You beat the game! If you had a good time, follow me for updates and spread the word about flexboxdefense.com!
Follow @ChanningAllen
Try again
Support this project

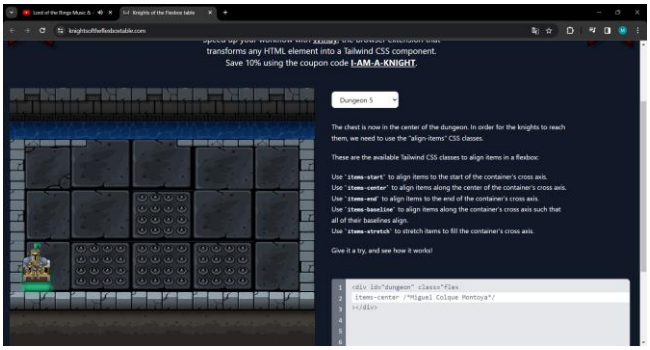
Knights Of The Flexbox Table

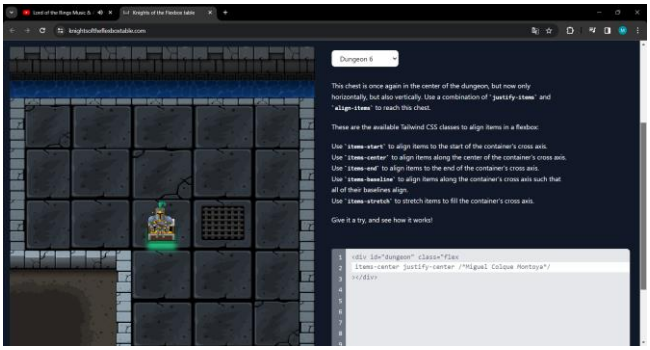


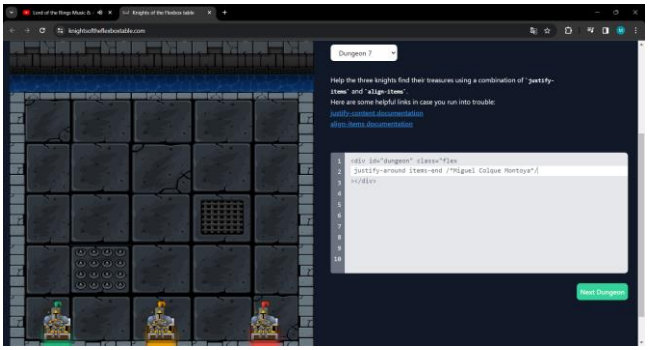


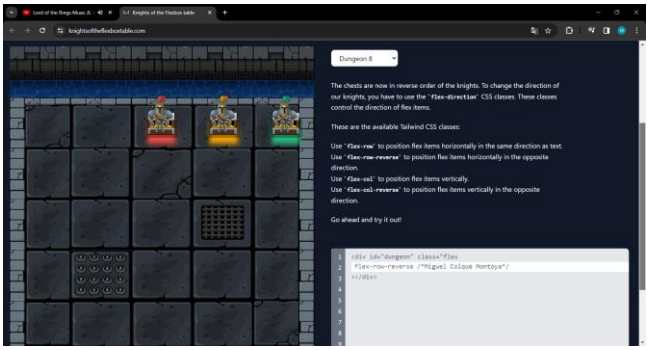


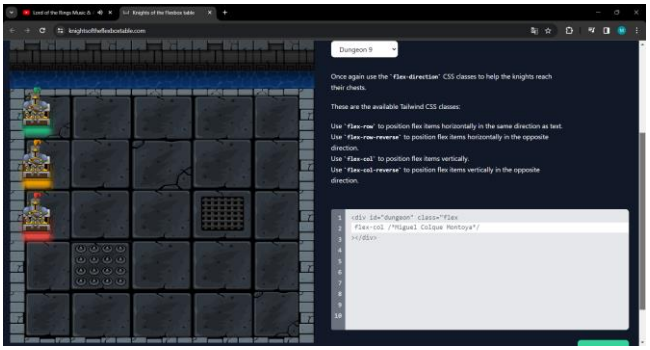


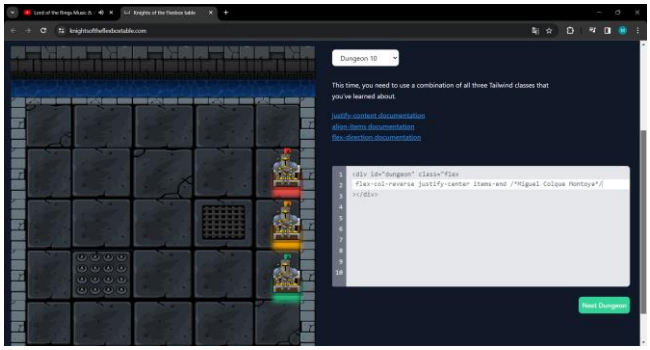


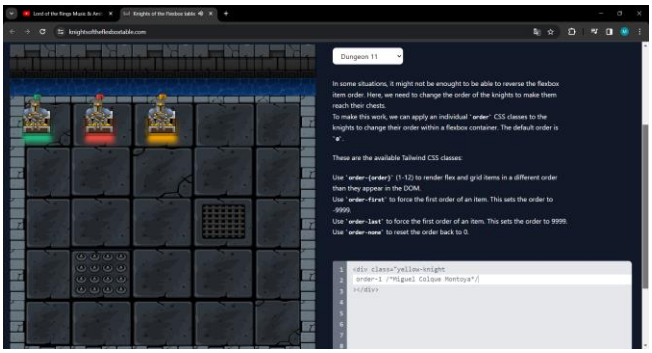


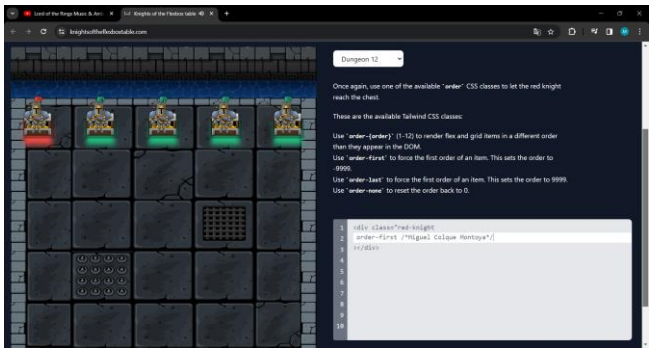


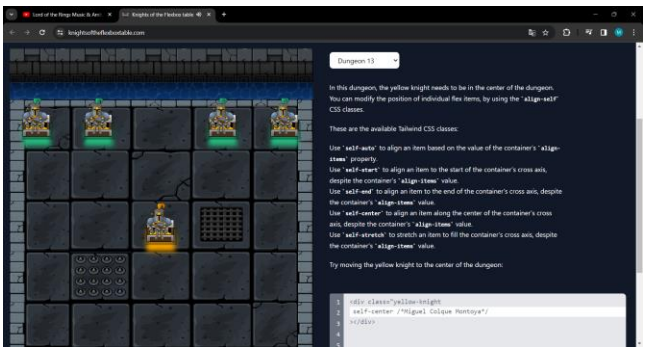


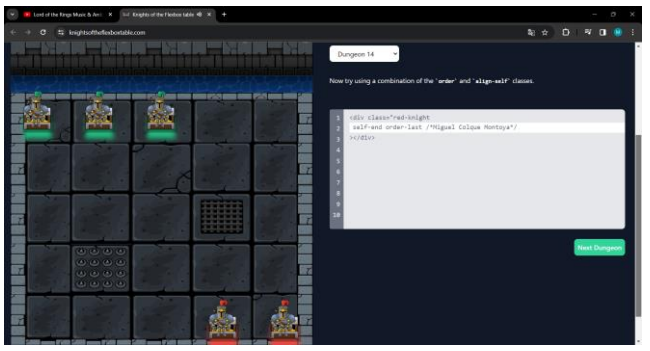

Dungeon 9
Once again use the `flex-direction` CSS classes to help the knights reach their chests.
These are the available Tailwind CSS classes:
Use `flex-row` to position flex items horizontally in the same direction as text.
Use `flex-row-reverse` to position flex items horizontally in the opposite direction.
Use `flex-col` to position flex items vertically.
Use `flex-col-reverse` to position flex items vertically in the opposite direction.
<div id="dungeon" class="flex
flex-col /*Miguel Colque Montoya*/
>

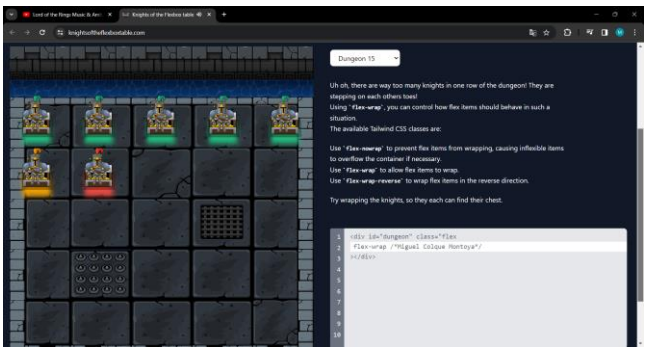

Dungeon 10
This time, you need to use a combination of all three Tailwind classes that you've learned about.
justify-content documentation
align-items documentation
flex-direction documentation
<div id="dungeon" class="flex
flex-col-reverse justify-center items-end /*Miguel Colque Montoya*/
>
Next Dungeon

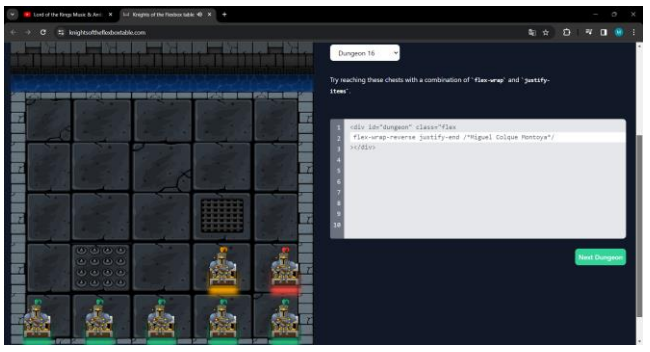

Dungeon 11
In some situations, it might not be enought to be able to reverse the flexbox item order. Here, we need to change the order of the knights to make them reach their chests.
To make this work, we can apply an individual `order` CSS classes to the knights to change their order within a flexbox container. The default order is `0`.
These are the available Tailwind CSS classes:
Use `order-[order]` (1-12) to render flex and grid items in a different order than they appear in the DOM.
Use `order-first` to force the first order of an item. This sets the order to -9999.
Use `order-last` to force the first order of an item. This sets the order to 9999.
Use `order-none` to reset the order back to 0.
<div class="yellow-knight
order-1 /*Miguel Colque Montoya*/
>

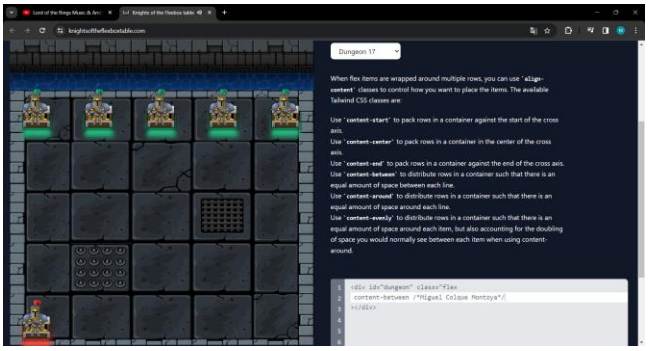

Dungeon 12
Once again, use one of the available `order` CSS classes to let the red knight reach the chest.
These are the available Tailwind CSS classes:
Use `order-[order]` (1-12) to render flex and grid items in a different order than they appear in the DOM.
Use `order-first` to force the first order of an item. This sets the order to -9999.
Use `order-last` to force the first order of an item. This sets the order to 9999.
Use `order-none` to reset the order back to 0.
<div class="red-knight
order-first /*Miguel Colque Montoya*/
>

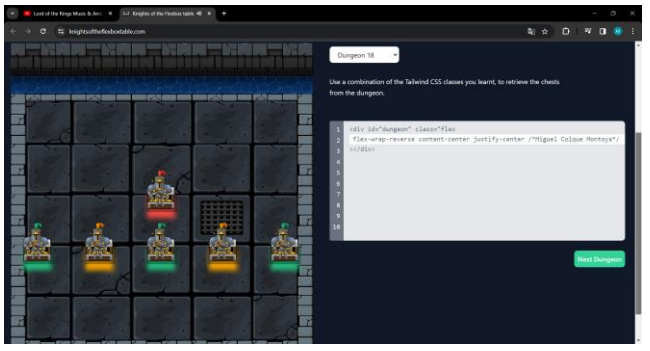

Dungeon 13
In this dungeon, the yellow knight needs to be in the center of the dungeon.
You can modify the position of individual flex items, by using the `align-self` CSS classes.
These are the available Tailwind CSS classes:
Use `self-auto` to align an item based on the value of the container's `align-items` property.
Use `self-start` to align an item to the start of the container's cross axis, despite the container's `align-items` value.
Use `self-end` to align an item to the end of the container's cross axis, despite the container's `align-items` value.
Use `self-center` to align an item along the center of the container's cross axis, despite the container's `align-items` value.
Use `self-stretch` to stretch an item to fill the container's cross axis, despite the container's `align-items` value.
Try moving the yellow knight to the center of the dungeon:
<div class="yellow-knight
self-center /*Miguel Colque Montoya*/
>

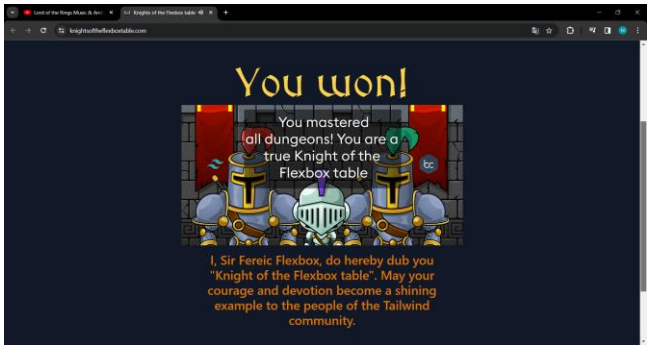

Dungeon 14
Now try using a combination of the `order` and `align-self` classes.
<div class="red-knight
self-end order-last /*Miguel Colque Montoya*/
>
Next Dungeon


Dungeon 15
Uh oh, there are way too many knights in one row of the dungeon! They are stepping on each others toes!
Using `flex-wrap`, you can control how flex items should behave in such a situation.
The available Tailwind CSS classes are:
Use `flex-nowrap` to prevent flex items from wrapping, causing inflexible items to overflow the container if necessary.
Use `flex-wrap` to allow flex items to wrap.
Use `flex-wrap-reverse` to wrap flex items in the reverse direction.
Try wrapping the knights, so they each can find their chest.
<div id="dungeon" class="flex
flex-wrap /*Miguel Colque Montoya*/
>


Dungeon 16
Try reaching these chests with a combination of `flex-wrap` and `justify-items`.
<div id="dungeon" class="flex
flex-wrap-reverse justify-end /*Miguel Colque Montoya*/
>
Next Dungeon


Dungeon 17
When flex items are wrapped around multiple rows, you can use `align-content` classes to control how you want to place the items. The available Tailwind CSS classes are:
Use `content-start` to pack rows in a container against the start of the cross axis.
Use `content-center` to pack rows in a container in the center of the cross axis.
Use `content-end` to pack rows in a container against the end of the cross axis.
Use `content-between` to distribute rows in a container such that there is an equal amount of space between each line.
Use `content-around` to distribute rows in a container such that there is an equal amount of space around each line.
Use `content-evenly` to distribute rows in a container such that there is an equal amount of space around each item, but also accounting for the doubling of space you would normally see between each item when using content-around.
<div id="dungeon" class="flex
content-between /*Miguel Colque Montoya*/
>


Dungeon 18
Use a combination of the Tailwind CSS classes you learnt, to retrieve the chests from the dungeon.
<div id="dungeon" class="flex
flex-wrap-reverse content-center justify-center /*Miguel Colque Montoya*/
>
Next Dungeon


You won!
You mastered all dungeons! You are a true Knight of the Flexbox table
I, Sir Fereic Flexbox, do hereby dub you "Knight of the Flexbox table". May your courage and devotion become a shining example to the people of the Tailwind community.