

Elements in web design exist in various "states." For example, a button has a "hover" state when the user "hovers" over it with the mouse. The button also has a "focus" state for when the button is focused on with the keyboard. **A pseudo class is a keyword, related to the element's state, that you attach to an element so you can define new styles for that state.**

Notice that pseudo classes are defined by only ONE colon where pseudo elements are defined by TWO colons. Let's start out easy by styling a button's hover state...

You can also think of pseudo classes as defining a "condition" or specific context.

Video



#brxe-gsguwr

CONTENTSTYLE

CSS Filters
[Learn more about CSS filters](#)

Transition
all 0s ease 0s
[Learn more about CSS transitions](#)

Custom CSS

```
1 root:hover {
2   background-color: black;
3 }
```

You can style this with the Bricks controls as well

Use "root" to target the element wrapper: root { background: blue; }

CSS classes

Separated by space. Without class dot.

CSS ID

Elements in web design exist in various "states." For example, a button has a "hover" state when the user "hovers" over it with the mouse. The button also has a "focus" state for when the button is focused on with the keyboard. **A pseudo class is a keyword, related to the element's state, that you attach to an element so you can define new styles for that state.**

```
<style>

.button:hover {}
.button:focus {}

</style>
```

Notice that pseudo classes are defined by only ONE colon where pseudo elements are defined by TWO colons. Let's start out easy by styling a button's hover state...

Style Me


You can also think of pseudo classes as defining a "condition" or specific context.

One of the most powerful and often-used pseudo classes is the **:nth** family of pseudo classes. This pseudo class allows you to style specific elements within a group of elements.

Let's take a look at **:first-child**, **:last-child**, and the most basic iterations of **:nth-child()** by styling the first, last, or second image in the group of images below.

Structure

- Section
 - Container
 - Heading
 - Rich Text
 - Code
 - Rich Text
 - Button
 - Section
 - Section
 - Section
 - Section
 - Section
 - Section
 - Global



◀▶⏮⏭⏯⏸⏹

05:25

🔊

🔍

#brxe-gsguwr

CONTENTSTYLE

CSS Filters

[Learn more about CSS filters](#)

Transition

all 0s ease 0s

[Learn more about CSS transitions](#)

Custom CSS

```
1 root:hover {
2   background-color: black;
3 }
```

Use "root" to target the element wrapper: root { background: blue; }

CSS classes

Separated by space. Without class dot.

CSS ID


Elements in web design exist in various "states." For example, a button has a "hover" state when the user "hovers" over it with the mouse. The button also has a "focus" state for when the button is focused on with the keyboard. **A pseudo class is a keyword, related to the element's state, that you attach to an element so you can define new styles for that state.**

```
<style>

.button:hover {}
.button:focus {}

</style>
```

Notice that pseudo classes are defined by only ONE colon where pseudo elements are defined by TWO colons. Let's start out easy by styling a button's hover state...



Also works in the Builder

Section

You can also think of pseudo classes as defining a "condition" or specific context.

One of the most powerful and often-used pseudo classes is the :nth family of pseudo classes. This pseudo class allows you to style specific elements within a group of elements.

Let's take a look at :first-child, :last-child, and the most basic iterations of :nth-child() by styling the first, last, or second image in the group of images below.

Structure

Section

Container

Heading

Rich Text

Code

Rich Text

Button

Section

Section


Section

Section

Section

Section

Global



05:36

Section

#brxe-ecgnkf

CONTE...

STYLE

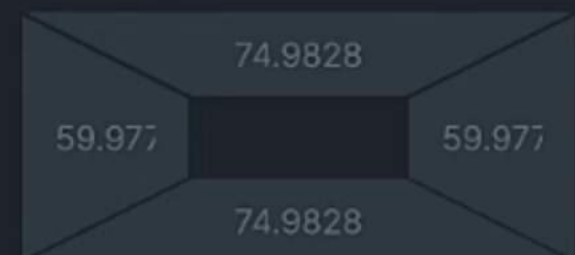
LAYOUT

SPACING

Margin



Padding



SIZING

Width

Min. width

Max. width

Height

Min. height

Max. height

Aspect ratio

POSITIONING

You can also think of pseudo classes as defining a "condition" or specific context.

One of the most powerful and often-used pseudo classes is the **:nth** family of pseudo classes. This pseudo class allows you to style specific elements within a group of elements.

Let's take a look at **:first-child**, **:last-child**, and the most basic iterations of **:nth-child()** by styling the first, last, or second image in the group of images below.



Image



:nth-child() vs :nth-of-type()

Where **:nth-child()** allows you to select ANY element based on its position, **:nth-of-type()** only selects specific types of children based on their position.

Structure

Section

Container

Heading

Rich Text

Code

Rich Text

Button

Section

Section

Section

Section

Section

Section

Global



#brxe-twylgia 1
grid--3

CONTENT

STYLE

TRANSFORM

• CSS

CSS Filters

[Learn more about CSS filters](#)

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

```
1 root > * {  
2   border: 5px solid var(--danger);  
3 }
```

With this we select every direct child of the container

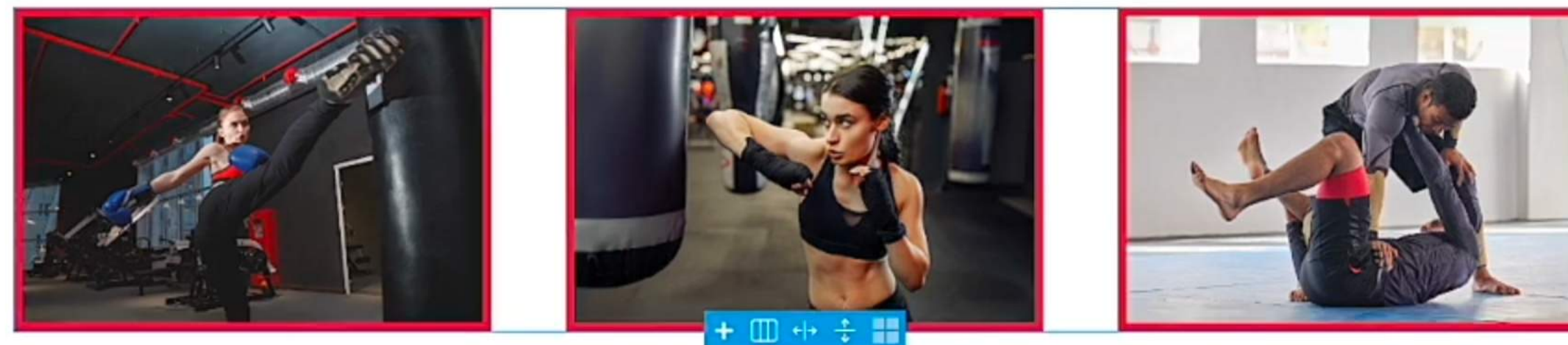
Use "root" to target the element wrapper: root { background: blue; }

CSS classes

You can also think of pseudo classes as defining a "condition" or specific context.

One of the most powerful and often-used pseudo classes is the **:nth** family of pseudo classes. This pseudo class allows you to style specific elements within a group of elements.

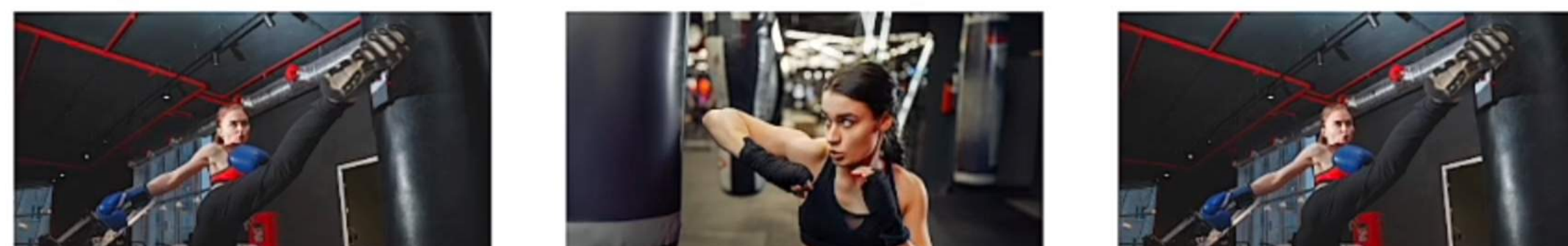
Let's take a look at **:first-child**, **:last-child**, and the most basic iterations of **:nth-child()** by styling the first, last, or second image in the group of images below.



:nth-child() vs :nth-of-type()

Where **:nth-child()** allows you to select ANY element based on its position, **:nth-of-type()** only selects specific types of children based on their position.

In the grid below, three of the images are wrapped in **<figure>** tags and three are not. What happens when we select the 2nd of the figure type? Can we select the first figure or last figure?



Structure

Section

Container

Heading

Rich Text

Code

Rich Text

Button

Section

Container

Container

Image

Image

Image

Container

Container

Container

Container

Container

Container

Container

Container

Container

Container

Container

Section

Section

Section

Section

10:15

[Icons]

#brxe-twylia 1
grid--3

CONTENT

STYLE

TRANSFORM

• CSS

CSS Filters

[Learn more about CSS filters](#)

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

```
1 root > *:first-child {  
2   border: 5px solid var(--danger);  
3 }
```

**Now we have targeted
the first child**

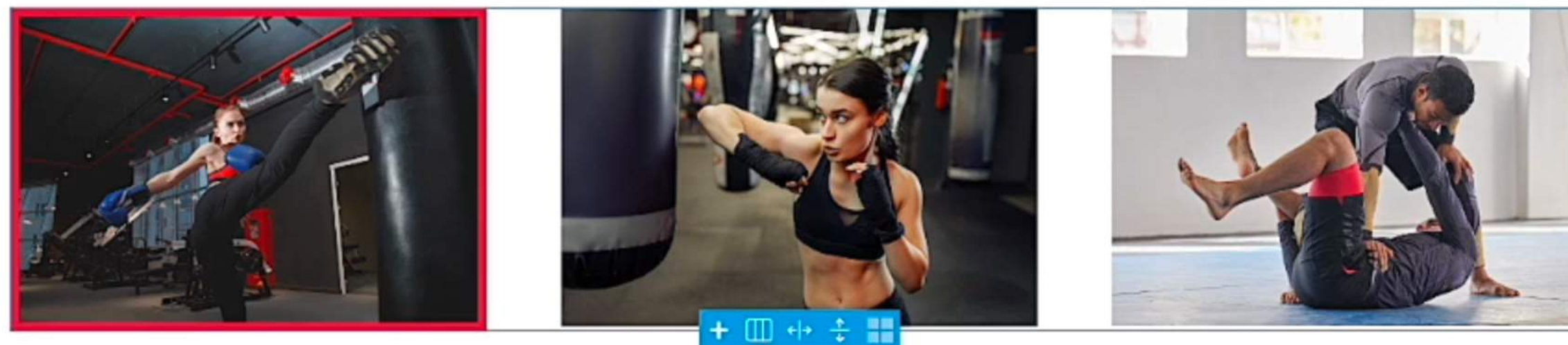
Use "root" to target the element wrapper: root { background: blue }

CSS classes

You can also think of pseudo classes as defining a "condition" or specific context.

One of the most powerful and often-used pseudo classes is the `:nth` family of pseudo classes. This pseudo class allows you to style specific elements within a group of elements.

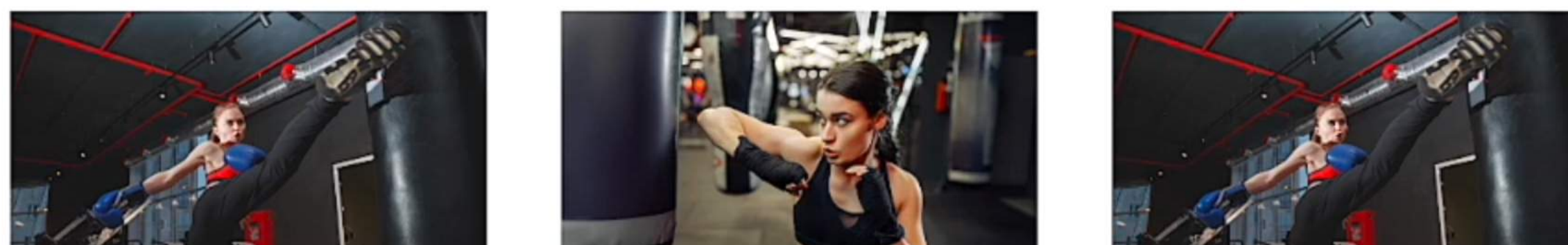
Let's take a look at `:first-child`, `:last-child`, and the most basic iterations of `:nth-child()` by styling the first, last, or second image in the group of images below.



:nth-child() vs :nth-of-type()

Where `:nth-child()` allows you to select ANY element based on its position, `:nth-of-type()` only selects specific types of children based on their position.

In the grid below, three of the images are wrapped in `<figure>` tags and three are not. What happens when we select the 2nd of the figure type? Can we select the first figure or last figure?



Structure

Section

Container

Heading

Rich Text

Code

Rich Text

Button

Section

Container

Container

Image

Image

Image

Container

Container

Container

Container

Container

Container

Container

Container

Container

Container

Section

Section

10:28

#brxe-twylgia 1
.grid--3

CONTENT

STYLE

STANDARD / STYLES

SHAPE DIVIDERS

TRANSFORM

• CSS

CSS Filters

[Learn more about CSS filters](#)

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

```
1 root > *:last-child {  
2   border: 5px solid var(--danger);  
3 }
```

This selects the last child

One of the most powerful and often-used pseudo classes is the `:nth` family of pseudo classes. This pseudo class allows you to style specific elements within a group of elements.

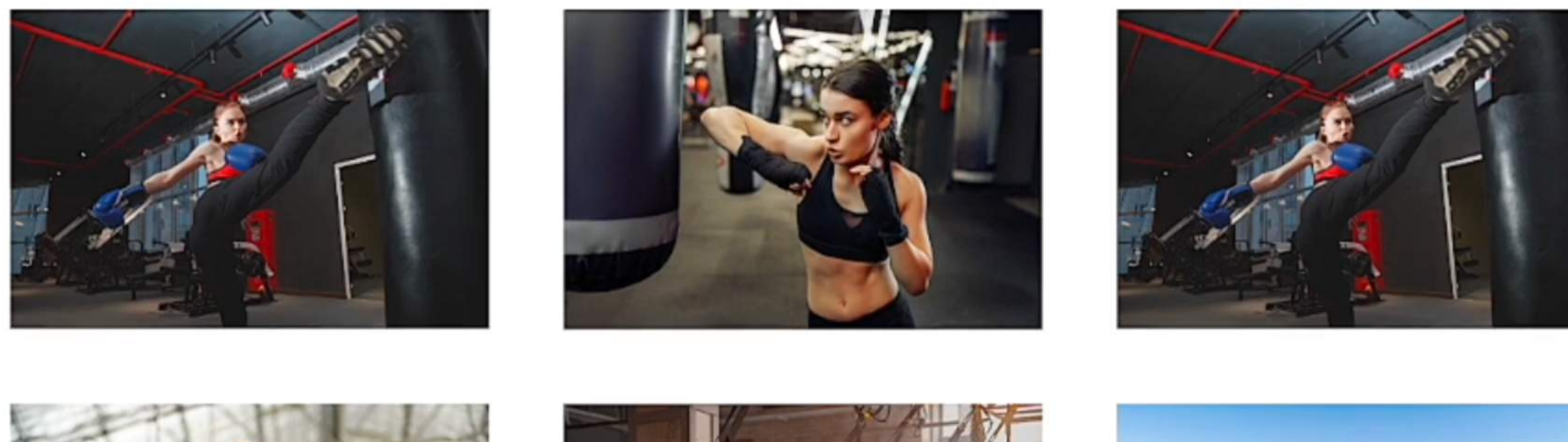
Let's take a look at `:first-child`, `:last-child`, and the most basic iterations of `:nth-child()` by styling the first, last, or second image in the group of images below.



`:nth-child()` vs `:nth-of-type()`

Where `:nth-child()` allows you to select ANY element based on its position, `:nth-of-type()` only selects specific types of children based on their position.

In the grid below, three of the images are wrapped in `<figure>` tags and three are not. What happens when we select the 2nd of the figure type? Can we select the first figure or last figure?



Structure

Section

Container

Heading

Rich Text

Code

Rich Text

Button

Section

Container

Container

Image

Image

Image

Container

Container

Container

Section

Section

Section

Section

Section

Section

Section

Section

Section

Section

Section

Section



W 992H - % 64

Rich Text

#brxe-nurfno

CONTENTSTYLE

LAYOUT

TYPOGRAPHY

BACKGROUND

BORDER / BOX SHADOW

GRADIENT / OVERLAY

TRANSFORM

CSS

CSS Filters

[Learn more about CSS filters](#)

Transition

all 0s ease 0s


[Learn more about CSS transitions](#)

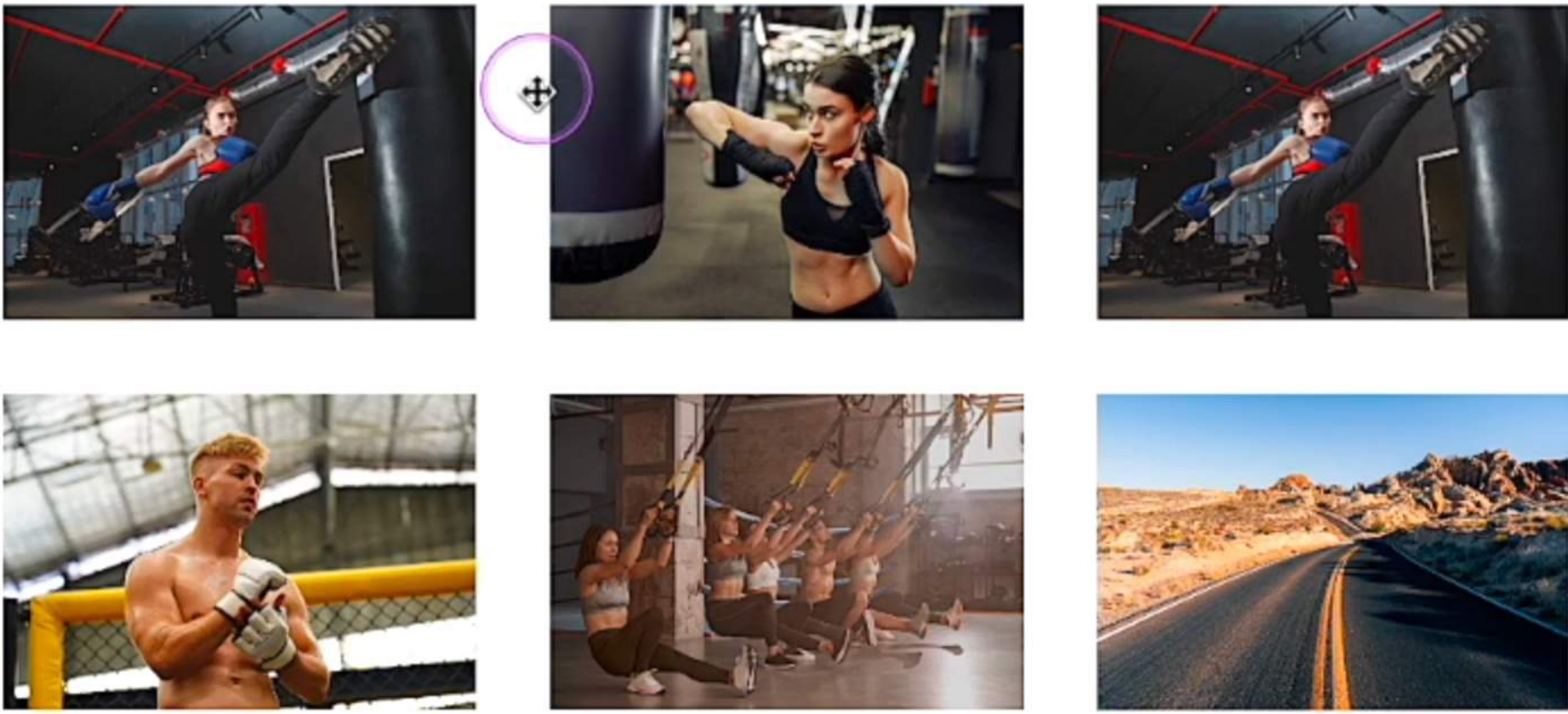
Custom CSS

:nth-child() vs :nth-of-type()

Where :nth-child() allows you to select ANY element based on its position, :nth-of-type() only selects specific types of children based on their position.

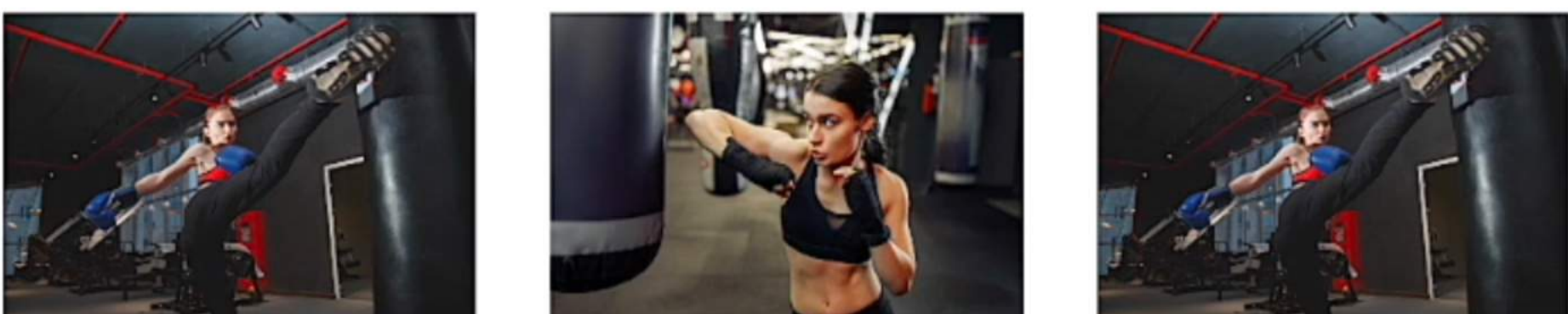
In the grid below, 3 of these images are wrapped in a figure tag. What happens when you select the 3rd or last figure?





Guess what? You can count backwards, too.

It doesn't happen often, but there can be certain situations where you need to be able to select elements from last-to-first. This is possible with :nth-last-child() and :nth-last-of-type().



Structure

Section

Container

Container

Image

Image

Image

Container

Rich Text

Rich Text

Container


Container

Container

Container

Section

Section



13:05

#brxe-uaiazd1

.grid--3

CONTENT

STYLE

Custom CSS

```
1 root figure {
2   border: 5px solid var(--danger);
3 }
```

No we checked which images have a figure tag

Use "root" to target the element wrapper: root { background: blue }

CSS classes

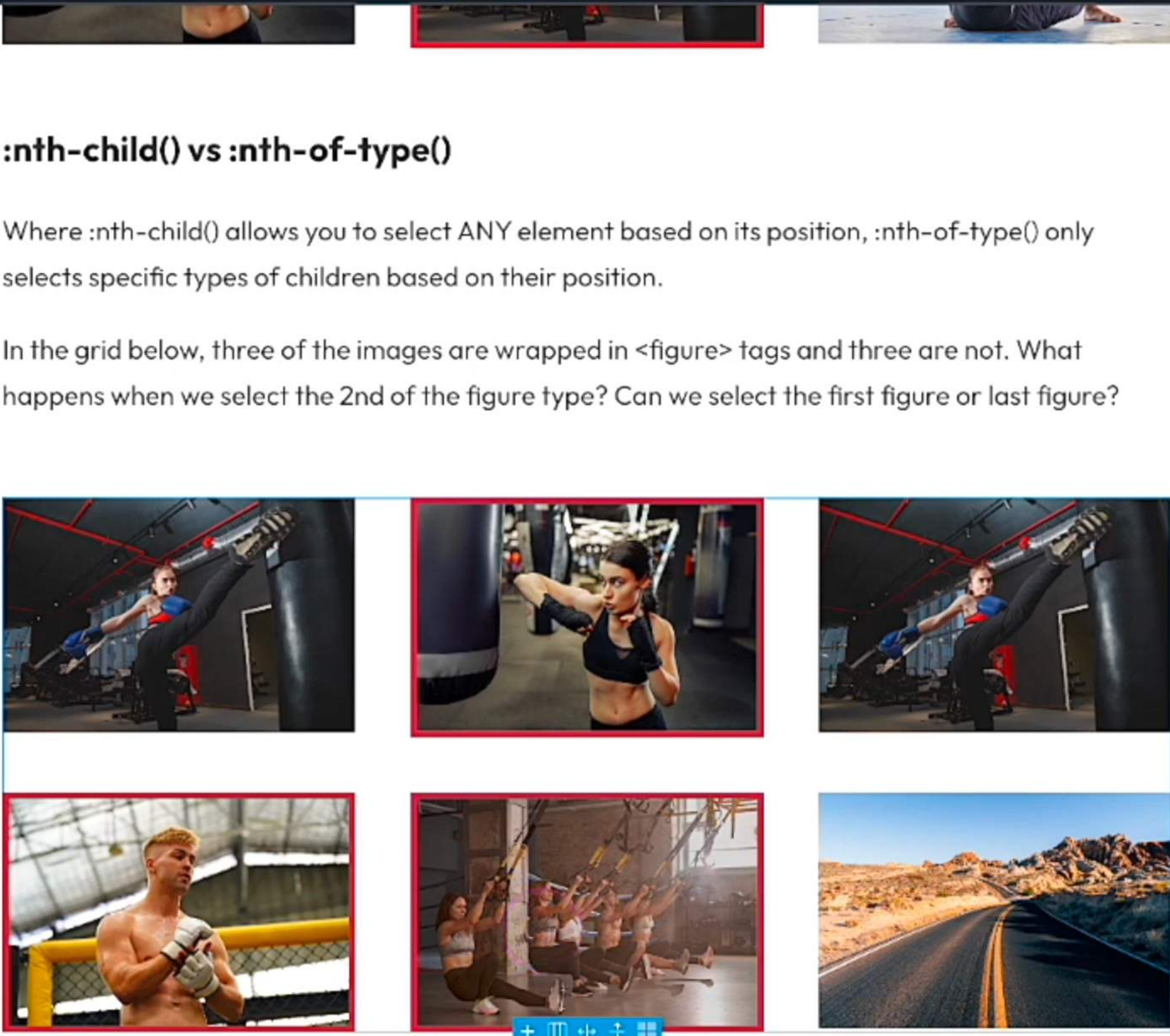
Separated by space. Without class dot.

CSS ID

:nth-child() vs :nth-of-type()


Where :nth-child() allows you to select ANY element based on its position, :nth-of-type() only selects specific types of children based on their position.

In the grid below, three of the images are wrapped in <figure> tags and three are not. What happens when we select the 2nd of the figure type? Can we select the first figure or last figure?



Guess what? You can count backwards, too.

It doesn't happen often, but there can be certain situations where you need to be able to select elements from last-to-first. This is possible with :nth-last-child() and :nth-last-of-type().



Structure

Container

Container

- Image
- Image
- Image

Container

- Rich Text
- Rich Text

Container

Container

Container

Container

Container

Container

Section

Section

14:12

#brxe-uaiazd1

.grid--3

CONTENT

STYLE

Custom CSS

```
1 root figure:nth-of-type(1) {
2   border: 5px solid var(--danger);
3 }
```

Use "root" to target the element wrapper: root { background: blue }

CSS classes

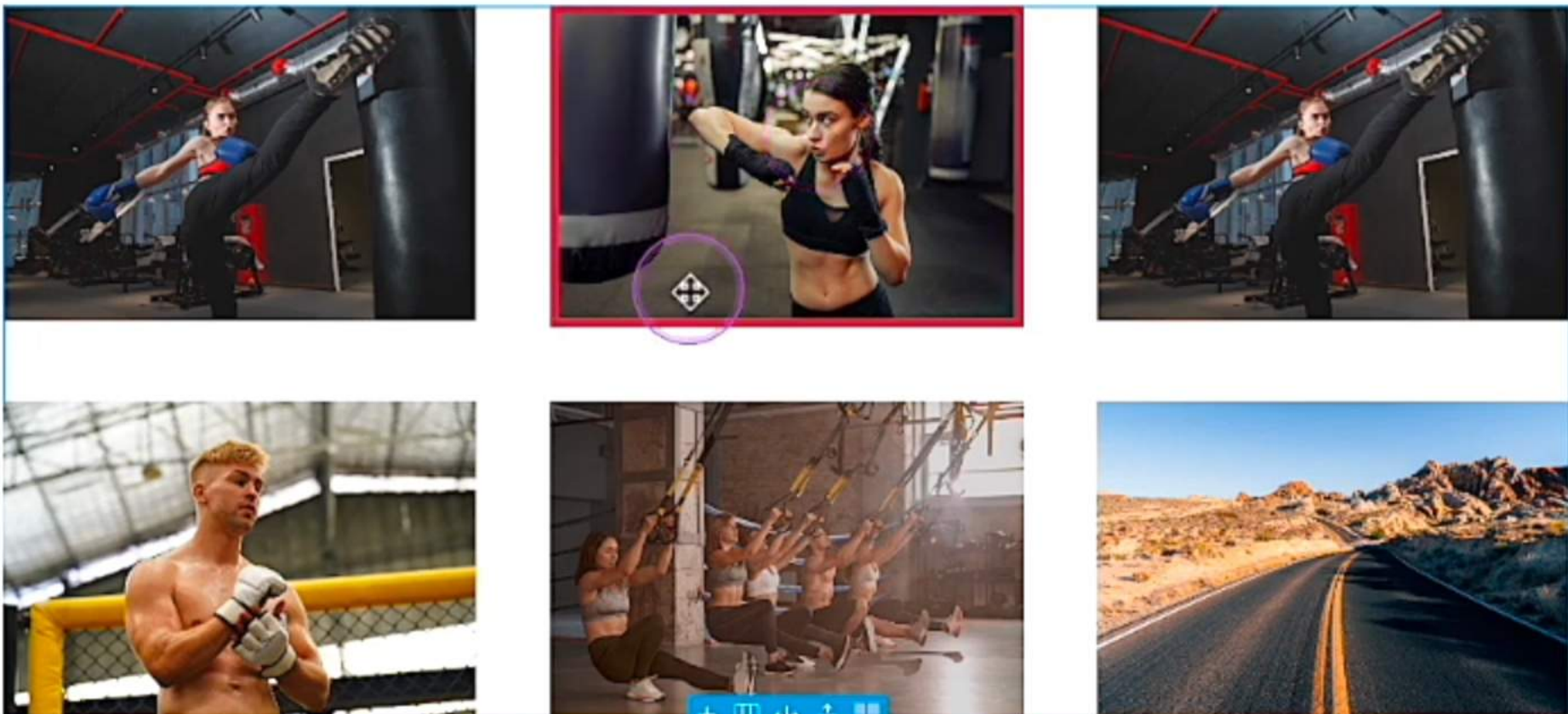
Separated by space. Without class dot.

CSS ID

:nth-child() vs :nth-of-type()

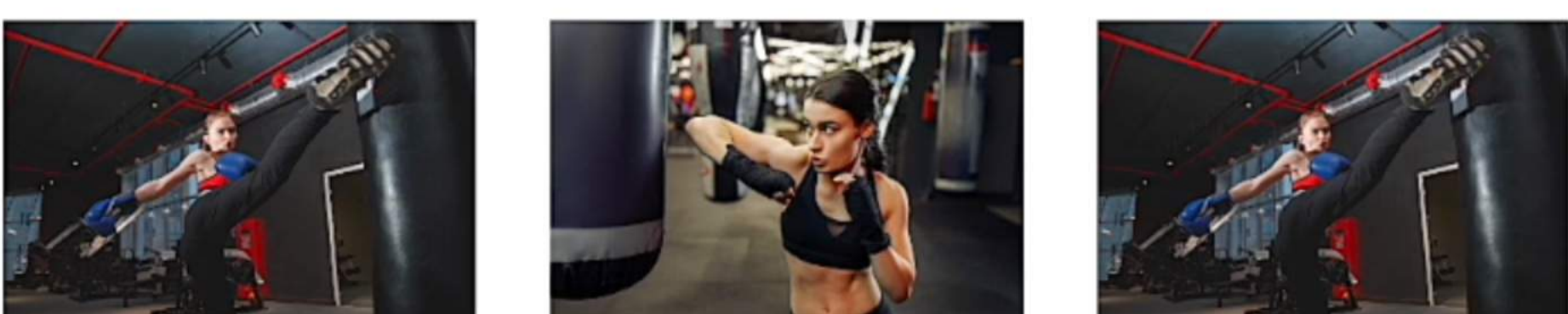
Where :nth-child() allows you to select ANY element based on its position, :nth-of-type() only selects specific types of children based on their position.

In the grid below, three of the images are wrapped in <figure> tags and three are not. What happens when we select the 2nd of the figure type? Can we select the first figure or last figure?



Guess what? You can count backwards, too.

It doesn't happen often, but there can be certain situations where you need to be able to select elements from last-to-first. This is possible with :nth-last-child() and :nth-last-of-type().



Structure

Container

Container

- Image
- Image
- Image

Container

- Rich Text
- Rich Text

Container

Container

Container

Container

Container

Container

Section

Section

14:30

#brxe-uaiazd1

.grid--3

CONTENTSTYLE

Custom CSS

```
1 root figure:last-of-type {
2   border: 5px solid var(--danger);
3 }
```

We selected the last figure element

Use "root" to target the element wrapper: root { background: blue }

CSS classes

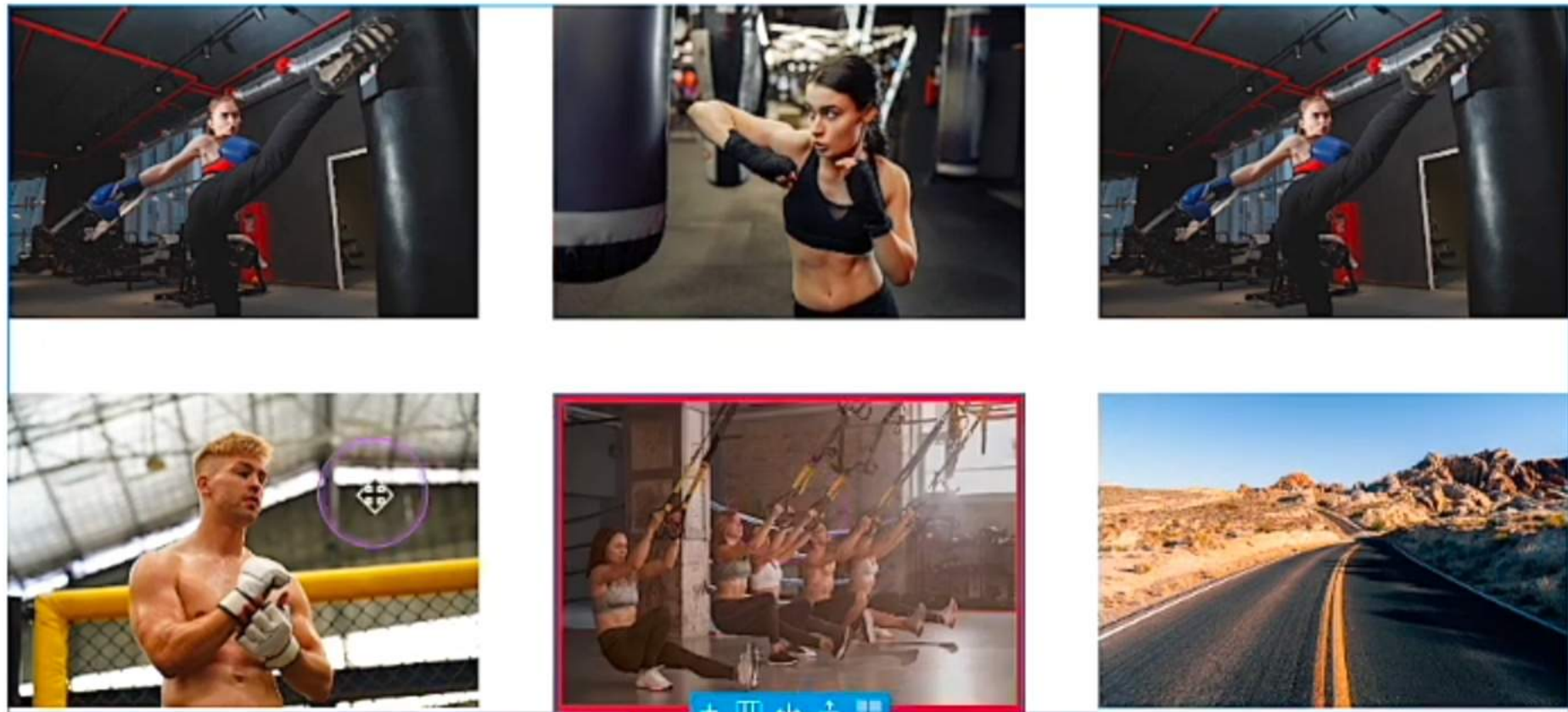
Separated by space. Without class dot.

CSS ID

:nth-child() vs :nth-of-type()

Where :nth-child() allows you to select ANY element based on its position, :nth-of-type() only selects specific types of children based on their position.


In the grid below, three of the images are wrapped in <figure> tags and three are not. What happens when we select the 2nd of the figure type? Can we select the first figure or last figure?



Image

Guess what? You can count backwards, too.

It doesn't happen often, but there can be certain situations where you need to be able to select elements from last-to-first. This is possible with :nth-last-child() and :nth-last-of-type().



Structure

Container

Container

- Image
- Image
- Image

Container

- Rich Text
- Rich Text

Container

Container

Container

Container

Container

Container

Section

Section

15:04

#brxe-uaiazd

1

.grid--3

CONTENT

STYLE

• Custom CSS

```
1 root figure:nth-last-of-type(2) {  
2   border: 5px solid var(--danger);  
3 }
```

No we're selecting backwards

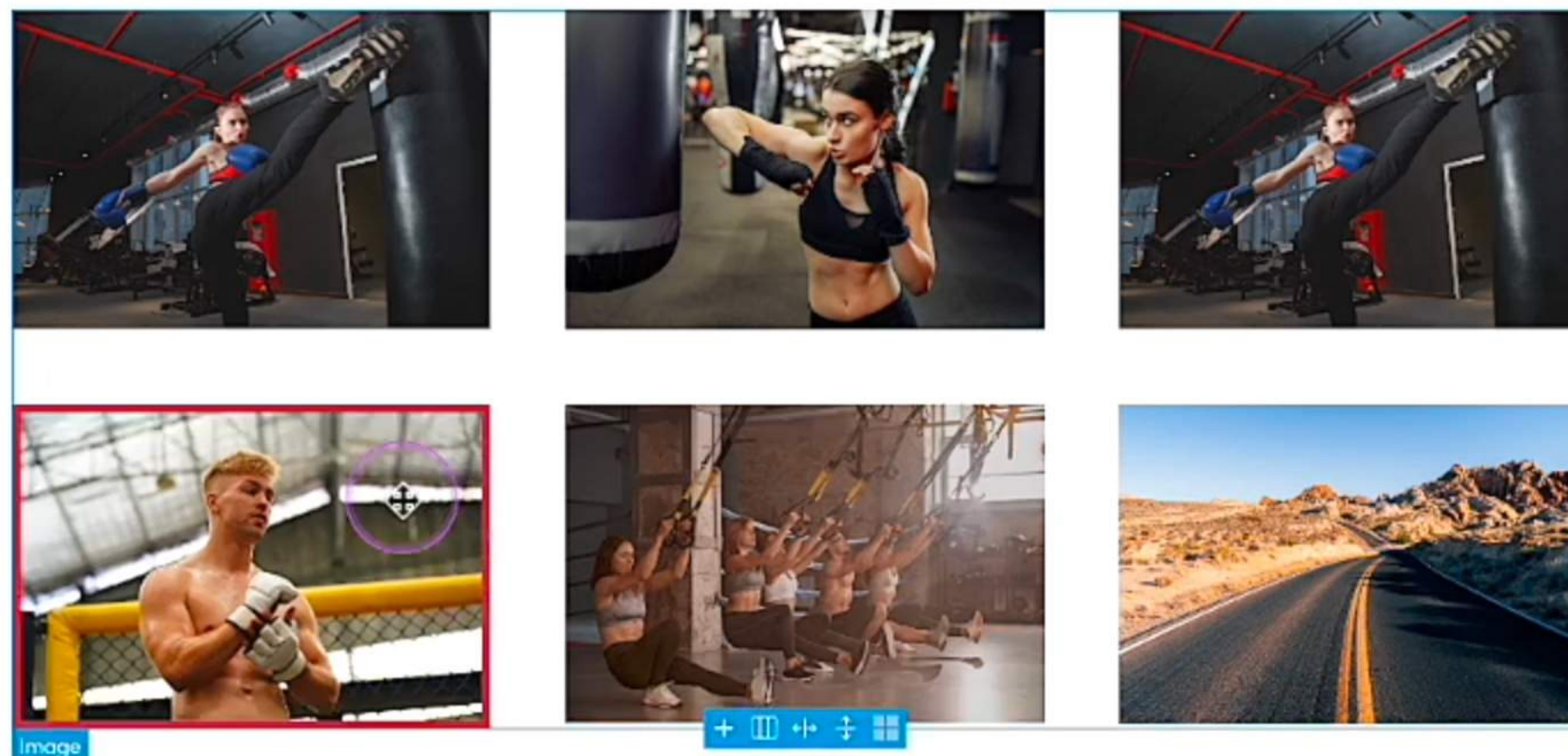
Use "root" to target the element wrapper: root { background: blue }

CSS classes

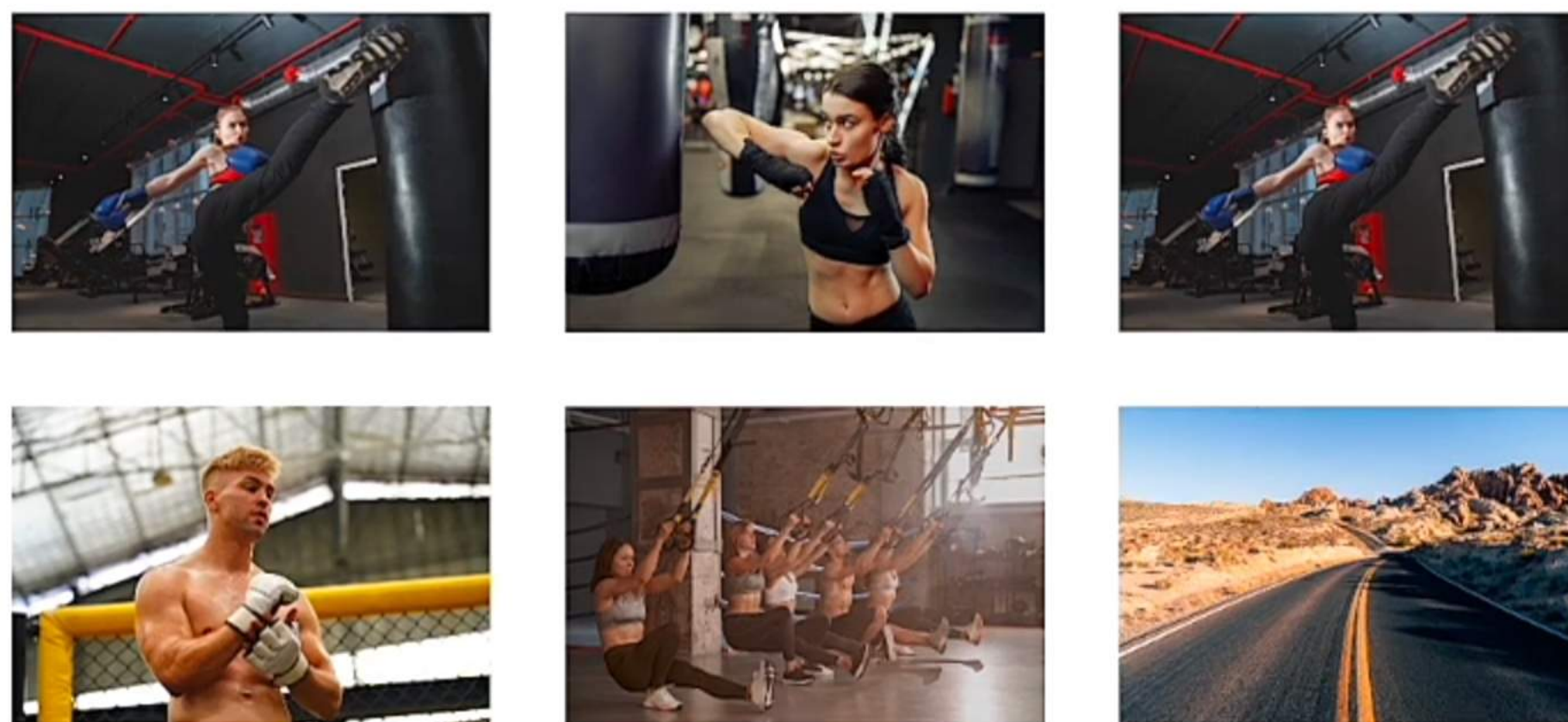
Separated by space. Without class dot.

CSS ID

happens when we select the 2nd of the figure type? Can we select the first figure or last figure?

**Guess what? You can count backwards, too.**

It doesn't happen often, but there can be certain situations where you need to be able to select elements from last-to-first. This is possible with :nth-last-child() and :nth-last-of-type().



Structure



Container

Container

Image

Image

Image

Container

Rich Text

Rich Text

Container

Container

Container

Container

Container

Container

Container

Section

Section

Section

Section



#brxe-yyeivt1

.grid--3

CONTENT

STYLE

[Learn more about CSS filters](#)

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

Custom CSS

```
1 root > *:nth-child(3n) {
2   border: 5px solid var(--danger);
3 }
```

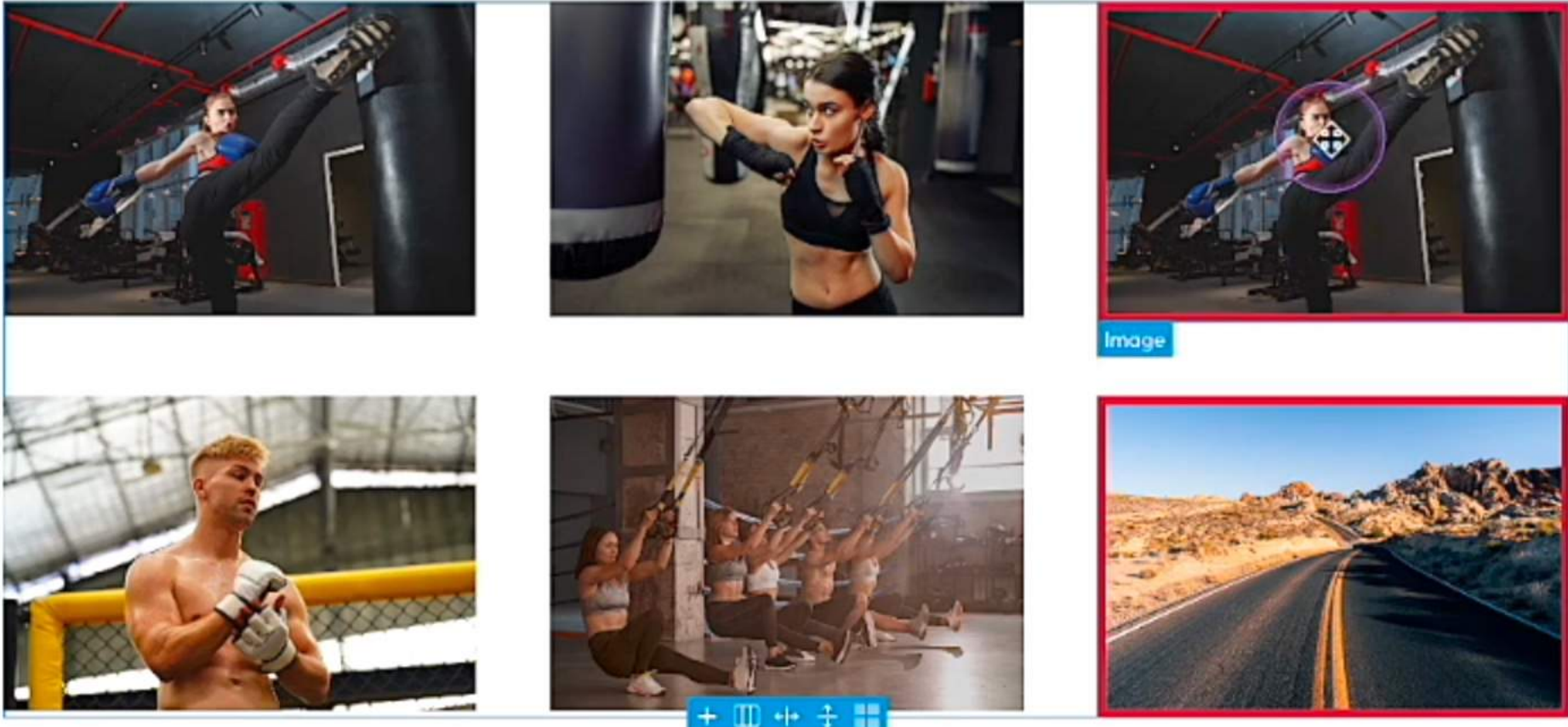
Now we are targeting every 3rd element

Use "root" to target the element wrapper: root { background: blue }

The :nth family of pseudo classes is also programmatic

The placeholder "n" allows you to programmatically select elements using the formula (an + b), where "a" is a multiplier, "n" is the selected child, and "b" is an offset value. Let's try these:

- Select every third element (3n)
- Select every third element, starting from element two (3n+2)
- Select only the first four elements (-n+4)
- Select only the last four elements (-n+4)
- Select all even or odd elements using (even) and (odd) presets



But wait, there's more!

You can combine multiple :nth statements together to create hyper-specific selection patterns. So if one formula doesn't do everything you need, just add another!

- Select the first four elements (-n+4) as well as the last element.

Structure

Image

Image

Container

- Rich Text
- Rich Text

Container

Container

Container

Container

- Rich Text
- Rich Text

Container

Container

Container

Global

⏮ ⏪ ⏩ ⏭

19:30 🔊 🔍

#brxe-yyeivt

1

.grid--3

CONTENT

STYLE

[Learn more about CSS filters](#)

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

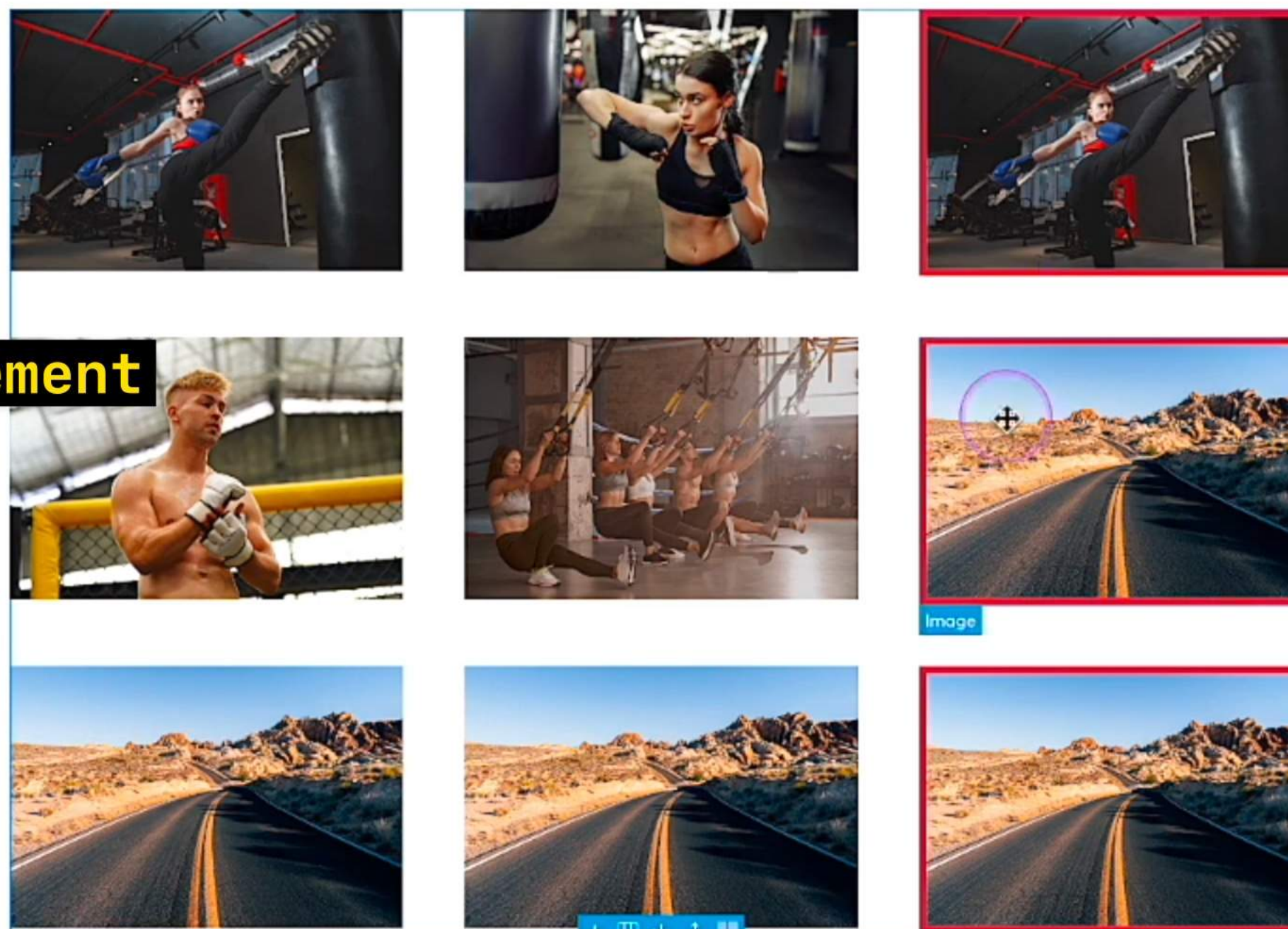
```
1 root > *:nth-child(3n+3) {  
2   border: 5px solid var(--danger);  
3 }
```

Now we're starting with the 2nd element

Use "root" to target the element wrapper: root { background: blue }

The placeholder "n" allows you to programmatically select elements using the formula $(an + b)$, where "a" is a multiplier, "n" is the selected child, and "b" is an offset value. Let's try these:

- Select every third element ($3n$)
- Select every third element, starting from element two ($3n+2$)
- Select only the first four elements ($-n+4$)
- Select only the last four elements ($-n+4$)
- Select all even or odd elements using (even) and (odd) presets

**But wait, there's more!**

You can combine multiple :nth statements together to create hyper-specific selection patterns. So

Structure

Container

Container

Rich Text

Rich Text

Container

Image

Image

Image

Image

Image

Image

Image

Image

Image

Section

Section

Section

Section

Section

Section

#brxe-yyeivt

1

.grid--3

CONTENT

STYLE

[Learn more about CSS filters](#)

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

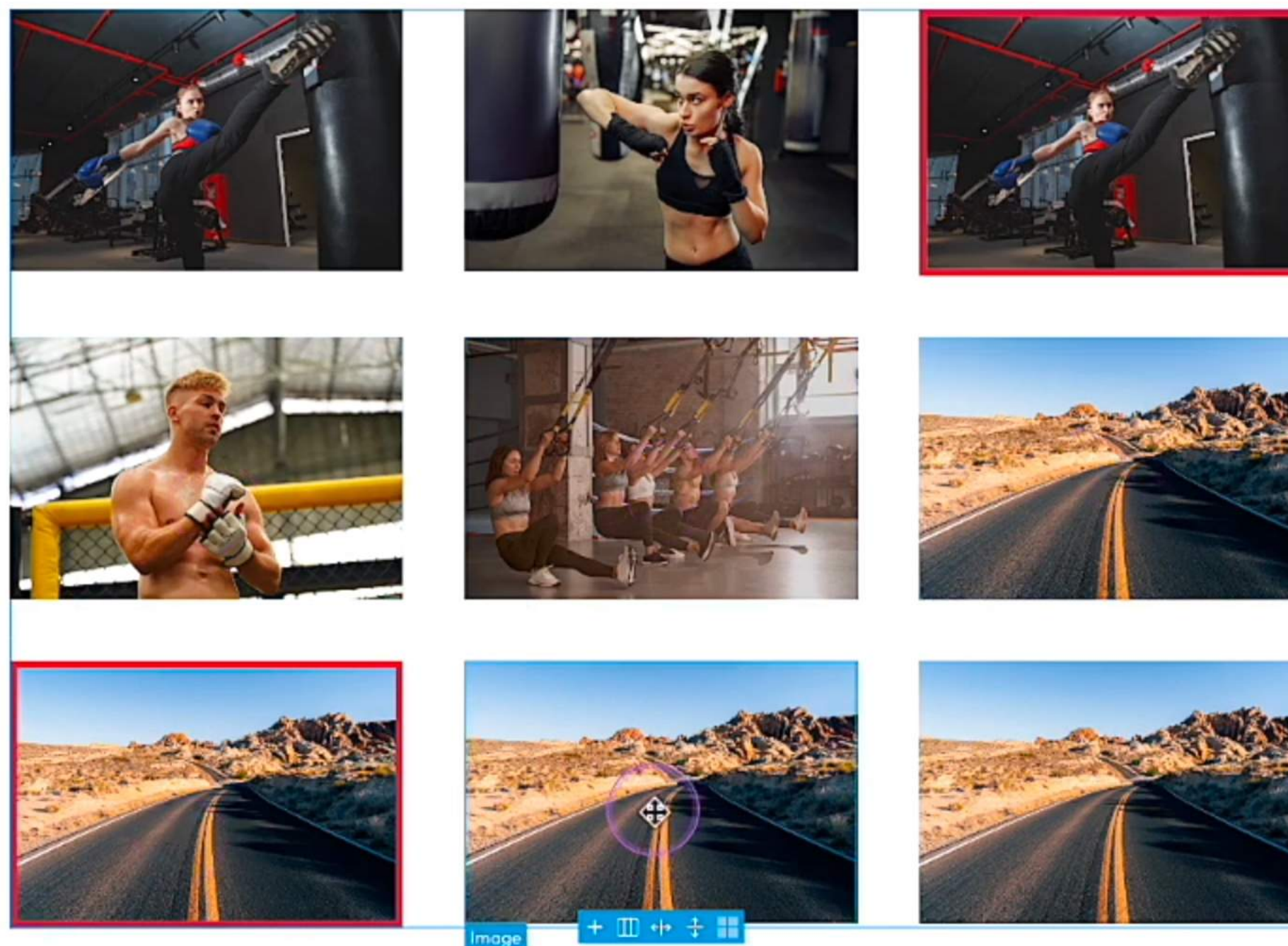
```
1 root > *:nth-child(4n+3) {  
2   border: 5px solid var(--danger);  
3 }
```

Select every 4th element, start counting with the 3rd element

Use "root" to target the element wrapper: root { background: blue }

The placeholder "n" allows you to programmatically select elements using the formula $(an + b)$, where "a" is a multiplier, "n" is the selected child, and "b" is an offset value. Let's try these:

- Select every third element ($3n$)
- Select every third element, starting from element two ($3n+2$)
- Select only the first four elements ($-n+4$)
- Select only the last four elements ($-n+4$)
- Select all even or odd elements using (even) and (odd) presets



But wait, there's more!

You can combine multiple :nth statements together to create hyper-specific selection patterns. So

Structure

> Container

Container

Rich Text

Rich Text

Container

Image

Image

Image

Image

Image

Image

Image

Image

Image

> Container

> Container

> Container

> Container

> Section

> Section

20:44

#brxe-yyeivt

1

.grid--3

CONTENT

STYLE

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

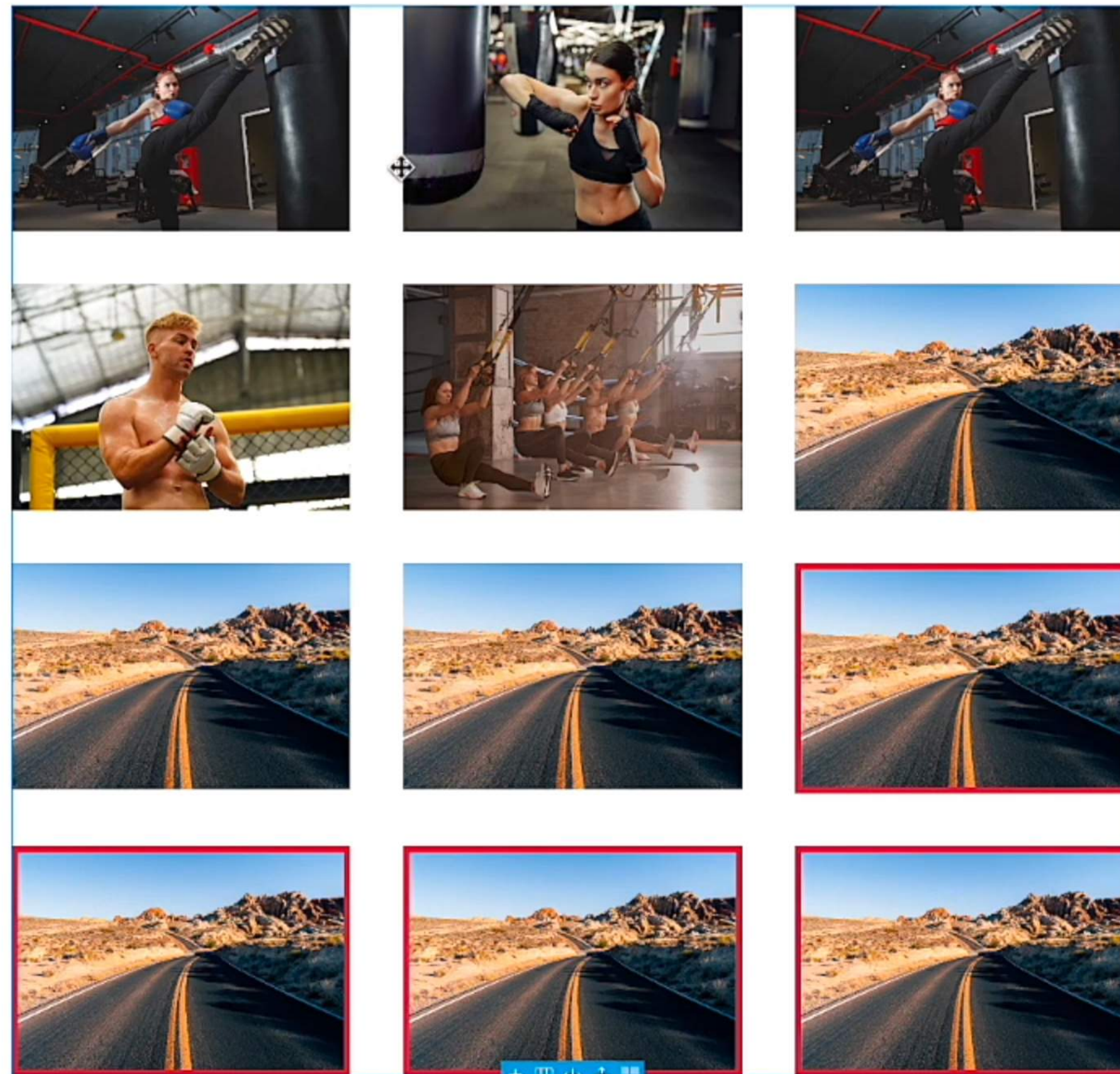
```
1 root > *:nth-last-child(-n+4) {
2   border: 5px solid var(--danger);
3 }
```

No we're selecting the
last 4 elements

Use "root" to target the element wrapper: root { background: blue }

CSS classes

- Select only the first four elements (-n+4)
- Select only the last four elements (-n+4)
- Select all even or odd elements using (even) and (odd) presets



But wait, there's more!

Structure



Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

> Container

> C

> [Icon]

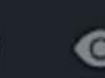
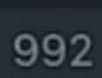
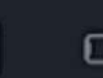
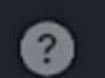
> [Icon]

> [Icon]

> [Icon]

> Global





#brxe-yyeivt

1

.grid--3

CONTENT

STYLE

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

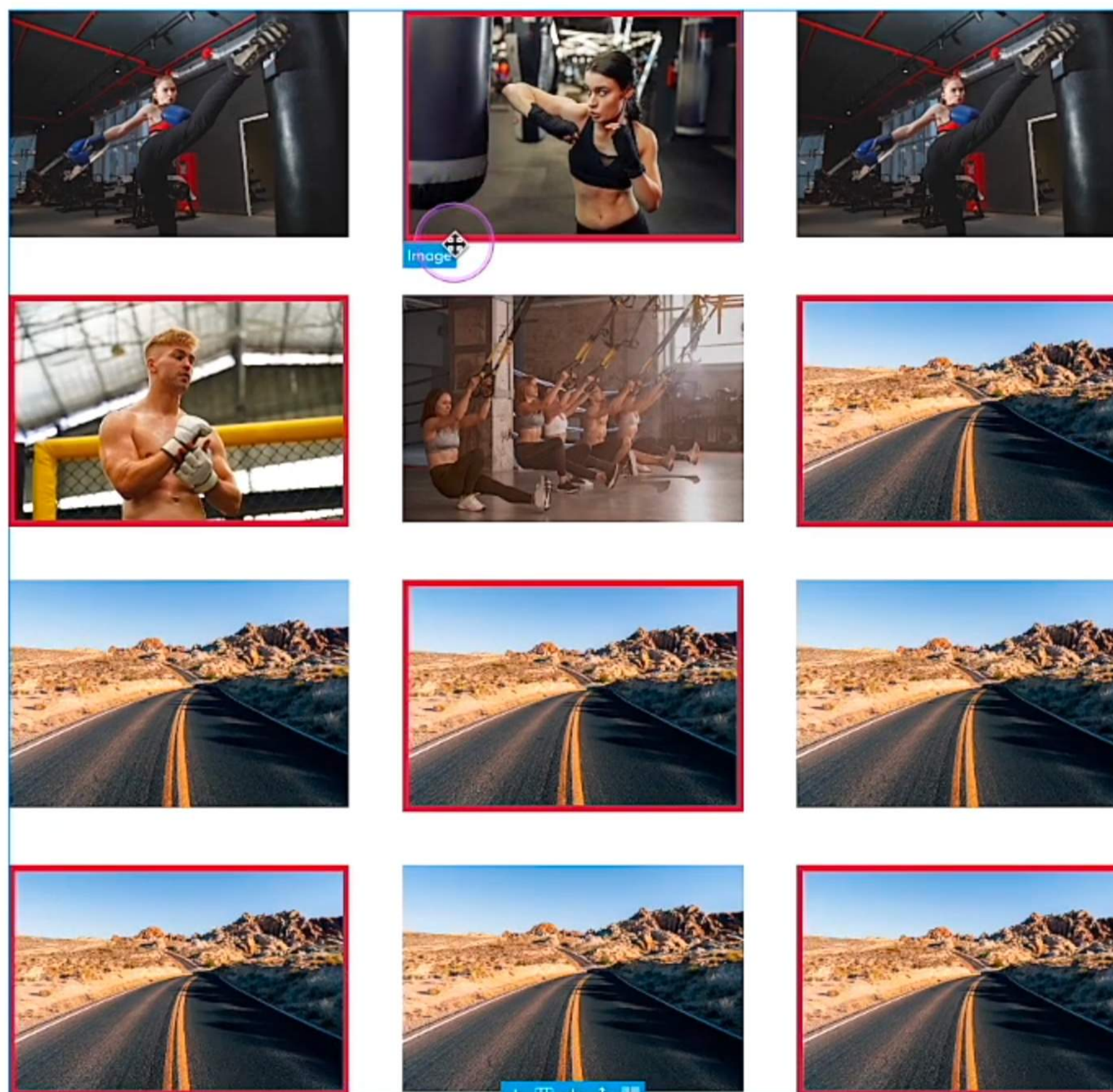
```
1 root > *:nth-child(even) {  
2   border: 5px solid var(--danger);  
3 }
```

We can also select odd and even

Use "root" to target the element wrapper: root { background: blue }

CSS classes

- Select every third element, starting from element two ($3n+2$)
- Select only the first four elements ($-n+4$)
- Select only the last four elements ($-n+4$)
- Select all even or odd elements using (even) and (odd) presets



Structure



Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

>

>

> Container

> C

>

>

>

>

>

>

>

#brxe-yyeivt

1

.grid--3

CONTENT

STYLE

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

```
1 root > *:nth-child(-n+4),  
2 root > *:last-child {  
3   border: 5px solid var(--danger);  
4 }
```

This is how we combine selectors

Use "root" to target the element wrapper: root { background: blue }

CSS classes

You can combine multiple :nth statements together to create hyper-specific selection patterns. So if one formula doesn't do everything you need, just add another!

- Select the first four elements (-n+4) as well as the last element.



Image



+ [] [] [] [] []



Structure

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

> Container

> C

> []

> []

> []

> []

> Global

You can also select elements based on a context where something is :not() happening or in situations where you want to exclude your styling rule

Imagine a scenario where you want to select all list items, but :not() list items that happen to be in a



#brxe-yyeivt

1

.grid--3

CONTENT

STYLE

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

```
1 root > *:nth-child(-n+4),
2 root > *:last-child {
3   border: 5px solid var(--danger);
4 }
```

Use "root" to target the element wrapper: root { background: blue }

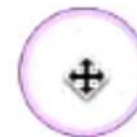
CSS classes

You can also select elements based on a context where something is :not() happening or in situations where you want to exclude your styling rule

Imagine a scenario where you want to select all list items, but :not() list items that happen to be in a navigation. We can do that by selecting all list items and then defining the :not() context we want to exclude. In my head, I always read this as "select this but not that" to help it make more sense.

- List item
- List item
- List item
- List item
- List item

- List item in nav
- List item in nav
- List item in nav
- List item in nav
- List item in nav
- List item in nav



You can also chain multiple contexts together in the same :not() by using comma separation. This reads as "select this, but not in this situation or this other situation."

- List item
- List item
- List item
- List item
- List item

- List item
- List item
- List item
- List item
- List item

- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude

Structure

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

> Container

> C

>

>

>

>

> Global



#brxe-cxuqyw
.grid--2

CONTENTSTYLE

SHAPE DIVIDERS

TRANSFORM

CSS

CSS Filters
[Learn more about CSS filters](#)

Transition
width 0s ease 0s
[Learn more about CSS transitions](#)

Custom CSS

Your input contains unbalanced brackets.

```
1 not li:not(nav li) {  
2   color: var(--danger);  
3 }
```

Tip: Think of the selection like: Select this but not that

You can also select elements based on a context where something is :not() happening or in situations where you want to exclude your styling rule

Imagine a scenario where you want to select all list items, but :not() list items that happen to be in a navigation. We can do that by selecting all list items and then defining the :not() context we want to exclude. In my head, I always read this as "select this but not that" to help it make more sense.

- List item
- List item
- List item
- List item
- List item

- List item in nav
- List item in nav
- List item in nav
- List item in nav
- List item in nav

You can also chain multiple contexts together in the same :not() by using comma separation. This reads as "select this, but not in this situation or this other situation."

- List item
- List item
- List item
- List item
- List item

- List item
- List item
- List item
- List item
- List item

- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude

Structure

Image

Image

Container

Container

Section

Container

Heading

Rich Text

Container

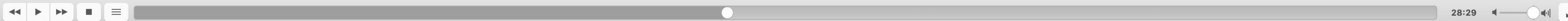
Rich Text

Rich Text

Rich Text

Container

Global



#brxe-hpxmpn

.grid--3

CONTENT

STYLE

CSS

CSS Filters

[Learn more about CSS filters](#)

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

Custom CSS

```
1 root li:not(nav li) {
2   color: var(--danger);
3 }
```

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

You can also chain multiple contexts together in the same :not() by using comma separation. This reads as "select this, but not in this situation or this other situation."

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

You can also use :not() as the starting selector:

• List item in nav

• List item in nav

• List item in nav

• List item in nav

• List item in nav

• List item

• List item

• List item

• List item

• List item

Structure

Container

Section

Container

Heading

Rich Text

Container

Rich Text

Rich Text

Rich Text

Rich Text

Container

Rich Text

Rich Text

Rich Text

Global

We also want to exclude the list items with the class 'exclude'

Alternatively, you can select items that match a specific

29:42

Pseudo Classes (Builder)Pseudo Classes – Bricks CleanPseudo Classes – Bricks Clean

Not Secure | bricks-clean.local/psuedo-2/?bricks=run

FacebookHey!BusinessPersonalHostingResourcesStreamYardBasecampAutomatic.cssFramesHubspotHelpScoutGeni.usInnerCircleACSSUserMavenJSSoftballPseudo Classes (...)

b

?

📄

🕒

⚙️

+

🔄

⋮

🖥️

📱

📺

📺

W 992

H -

% 64

🔍

📁

🔗

👁️

💾

Rich Text

#brxe-bcnfsa

CONTENTSTYLE

LAYOUT

TYPOGRAPHY

BACKGROUND

BORDER / BOX SHADOW

GRADIENT / OVERLAY

TRANSFORM

CSS

CSS Filters

[Learn more about CSS filters](#)

Transition

all 0s ease 0s

[Learn more about CSS transitions](#)

Custom CSS

• List item in nav

• List item in nav

• List item

• List item

Alternatively, you can select items that match a specific selector or context using :is().

Heading

What if we want to style list items that are only in two specific contexts, like list items that are either in a nav or in a parent with the class ".include."

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

• List item in container with .exclude

You can do the exact same thing with :where(), but the :where() pseudo class resets the specificity of the styles to zero...

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item

• List item in container with .exclude

• List item in container with .exclude

Structure

Rich Text

Rich Text

Container

Section

Container

Heading

Rich Text

Container

Rich Text

Rich Text

Rich Text

Heading

Container

Section

31:18

🔊

🔍



- List item in nav
- List item in nav
- List item in nav

- List item
- List item
- List item

Alternatively, you can select items that match a specific selector or context using :is().

What if we want to style list items that are only in two specific contexts, like list items that are either in a nav or in a parent with the class ".include."

- List item
- List item
- List item
- List item
- List item

- List item
- List item
- List item
- List item
- List item

- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude

You can do the exact same thing with :where(), but the :where() pseudo class resets the specificity of the styles to zero...

- List item
- List item
- List item

- List item
- List item
- List item

- List item in container with .exclude
- List item in container with .exclude

Structure



Rich Text

Rich Text

Container

Section

Container

Heading

Rich Text

Container

Rich Text

Rich Text

Rich Text

Heading

Container

T

>

>

>

>

Section

Section



#brxe-axwnwa

1

.grid--3

CONTENT

STYLE

Custom CSS

```
1 root :is(nav li, .include li) {  
2   color: var(--danger);  
3 }
```

Same principle as before

Use "root" to target the element wrapper: root { background: blue }

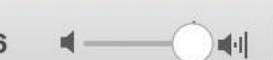
CSS classes

Separated by space. Without class dot.

CSS ID



33:16



#brxe-axwnwa

.grid--3

CONTENT

STYLE

TRANSFORM

CSS

CSS Filters

Learn more about CSS filters

Transition

width 0s ease 0s

Learn more about CSS transitions

Custom CSS

```
1 root :is(nav li, .include li) {
2   color: var(--danger);
3 }
```

List item

List item

List item

List item

List item

List item

List item

List item

List item

List item

List item

List item

List item in container with .exclude

List item in container with .exclude

List item in container with .exclude

List item in container with .exclude




List item in container with .exclude

List item in container with .exclude

You can do the exact same thing with :where(), but the :where() pseudo class resets the specificity of the styles to zero...

And then there's :has(), one of the most powerful pseudo classes ever created...

:has() allows you to style a parent based the existence of a child context, something that's never been possible until very recently. For example, I want to select every .brxe-container that has a figure tag somewhere inside it.



Structure

Rich Text

Rich Text

Container

Section

Container

Heading

Rich Text

Container

Rich Text

Rich Text

Rich Text

Heading

Container

Section

Section

33:56

Page settings

GENERAL

ONE PAGE NAVIGATION

• CUSTOM CODE

• Custom CSS

```
1 li {  
2   color: green;  
3 }  
4  
5 :where(nav li, .include li) {  
6   color: var(--danger);  
7 }  
8  
9 section:not(:last-child) {  
10  border-bottom: 1px solid #ddd;  
11 }  
12  
13 h2 {  
14  width: 38ch;  
15 }  
16  
17 .brxe-text ul, .brxe-text ol {  
18  margin: 0;  
19  padding: 0;  
20  padding-inline-start: 1em;  
21 }  
22  
23 .brxe-text ul li:not(:last-child), ol li:not(:last-child) {  
24  margin-bottom: 10px;  
25 }
```

Adds inline CSS to <head> tag.

Header scripts

- List item
- List item
- List item
- List item
- List item
- List item
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude

Here we are in the Page CSS and the first rule is overriding the second rule due to it's higher specificity

- List item
- List item
- List item
- List item
- List item
- List item
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude
- List item in container with .exclude

And then there's :has(), one of the most powerful pseudo classes ever created...

:has() allows you to style a parent based the existence of a child context, something that's never been possible until very recently. For example, I want to select every .brxe-container that has a

Structure

Container

Section

Container

Heading

Rich Text

Container

Rich Text

Rich Text

Rich Text

Heading

Container

Heading

Rich Text

Container

Section

Container

Section

Container

Global





Page settings

GENERAL

ONE PAGE NAVIGATION

• CUSTOM CODE

• Custom CSS

```
1 :where(nav li, .include li) {  
2   color: var(--danger);  
3 }  
4  
5 section:not(:last-child) {  
6   border-bottom: 1px solid #ddd;  
7 }  
8  
9 h2 {  
10  width: 38ch;  
11 }  
12  
13 .brxe-text ul, .brxe-text ol {  
14   margin: 0;  
15   padding: 0;  
16   padding-inline-start: 1em;  
17 }  
18  
19 .brxe-text ul li:not(:last-child), ol li:not(:last-child) {  
20   margin-bottom: 10px;  
21 }  
22  
23 p, li {  
24   max-width: 70ch !important;  
25 }
```

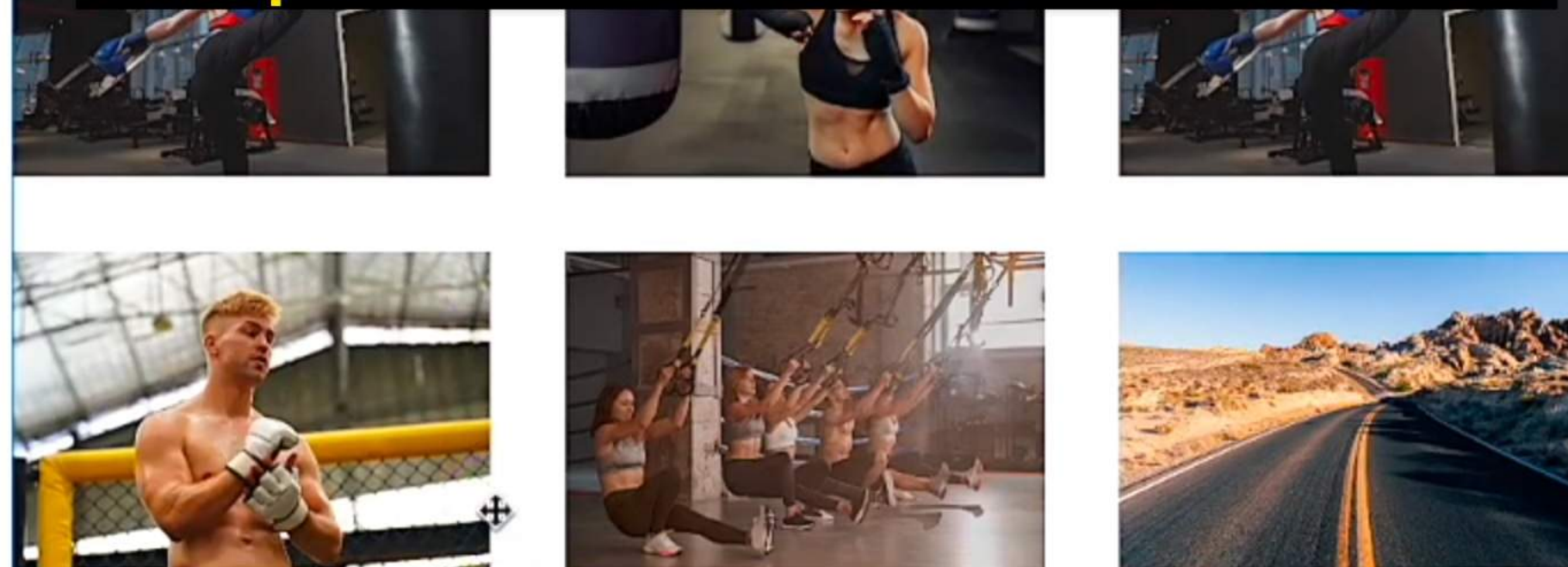
Adds inline CSS to <head> tag.

Header scripts

And then there's :has(), one of the most powerful pseudo classes ever created...

:has() allows you to style a parent based the existence of a child context, something that's never been possible until very recently. For example, I want to select every .brxe-container that has a figure tag somewhere inside it.

With this selector we can style the parent because of it's content



You can even combine :has() and :nth(). For example, you can select any container that has a direct child figure element in the 2nd position :has(>figure:nth-child(2))

Just remember: :has() is still waiting on full browser support (but it's very close!).

Don't forget accessibility with :focus and :focus-within

Structure

Rich Text

Rich Text

Container

Section

Container

Heading

Rich Text

Container

Rich Text

Rich Text

Rich Text

Heading

Container

T

Section

Section



Page settings

GENERAL

ONE PAGE NAVIGATION

CUSTOM CODE

Custom CSS

```
1 .brxe-container:has(> figure) {  
2   border: 5px solid var(--danger);  
3 }
```

```
5 :where(nav li, .include li) {  
6   color: var(--danger);  
7 }
```

```
9 section:not(:last-child) {  
10  border-bottom: 1px solid #ddd;  
11 }
```

```
13 h2 {  
14  width: 38ch;  
15 }
```

```
17 .brxe-text ul, .brxe-text ol {  
18  margin: 0;  
19  padding: 0;  
20  padding-inline-start: 1em;  
21 }
```

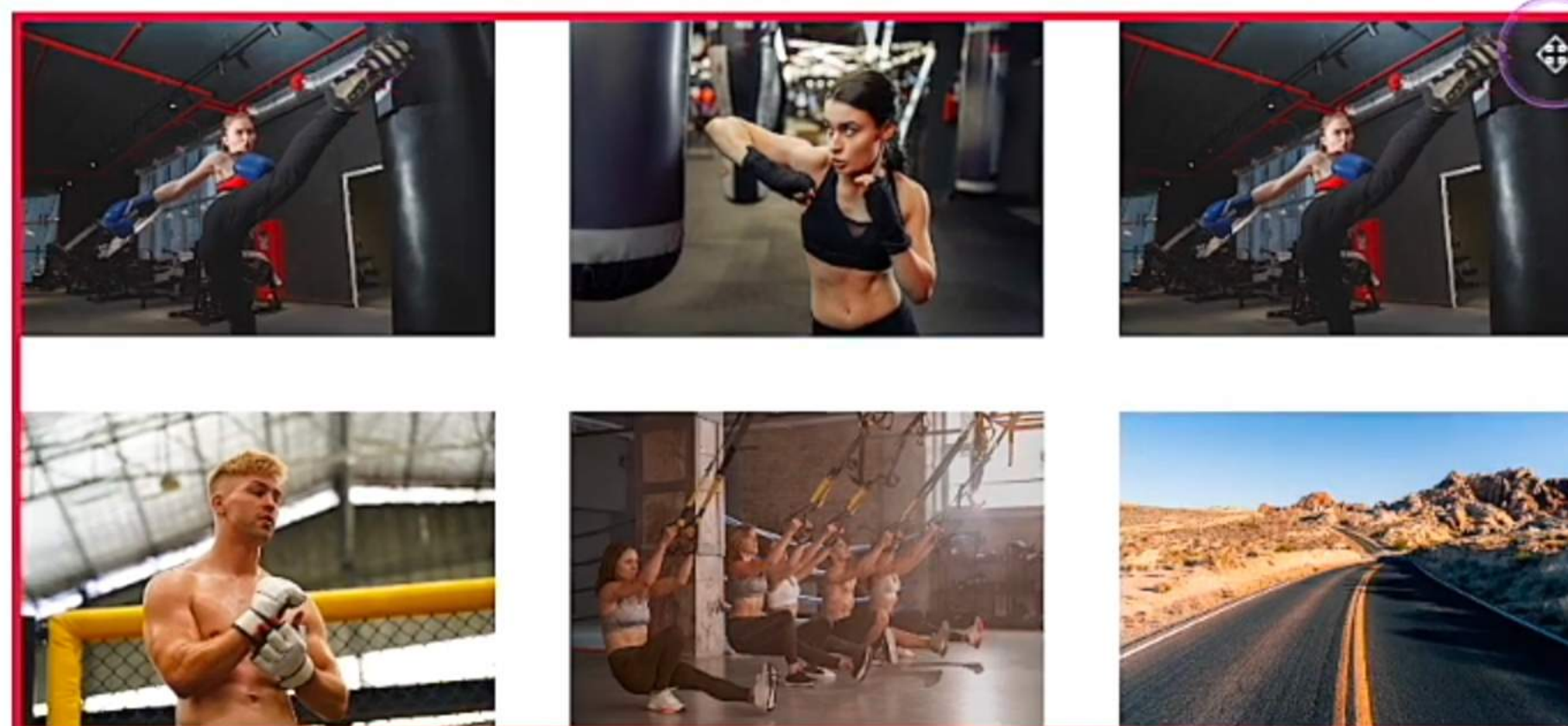
```
23 .brxe-text ul li:not(:last-child), ol li:not(:last-child) {  
24  margin-bottom: 10px;  
25 }
```

Adds inline CSS to <head> tag.

Header scripts

And then there's :has(), one of the most powerful pseudo classes ever created...

:has() allows you to style a parent based the existence of a child context, something that's never been possible until very recently. For example, I want to select every .brxe-container that has a figure tag somewhere inside it.



You can even combine :has() and :nth(). For example, you can select any container that has a direct child figure element in the 2nd position :has(>figure:nth-child(2))

Just remember: :has() is still waiting on full browser support (but it's very close!).

Structure

Rich Text

Rich Text

Container

Section

Container

Heading

Rich Text

Rich Text

Rich Text

Rich Text

Rich Text

Heading

Container

T

Section

Section

Pseudo Classes (Builder) x Pseudo Classes - Bricks Clean x Pseudo Classes - Bricks Clean x +

Not Secure | bricks-clean.local/psuedo-2/?bricks=run

Facebook Hey! Business Personal Hosting Resources StreamYard Basecamp Automatic.css Frames Hubspot HelpScout Geni.us InnerCircle ACSS UserMaven JS Softball Pseudo Classes (...)

b ? [Icons] W 992 H - % 64 [Icons]

Page settings

GENERAL

ONE PAGE NAVIGATION




CUSTOM CODE

Custom CSS

```
1 .brxe-container:has(> figure) {
2   border: 5px solid var(--danger);
3 }
4
5 :where(nav li, .include li) {
6   color: var(--danger);
7 }
8
9 section:not(:last-child) {
10  border-bottom: 1px solid #ddd;
11 }
12
13 h2 {
14   width: 38ch;
15 }
16
17 .brxe-text ul, .brxe-text ol {
18   margin: 0;
19 }
```

Adds inline CSS to <head> tag.

Header scripts



You can even combine :has() and :nth(). For example, you can select any container that has a direct child figure element in the 2nd position :has(>figure:nth-child(2))

Just remember: :has() is still waiting on full browser support (but it's very close!).

Don't forget accessibility with :focus and :focus-within

Keyboard users are important, too. :focus sets the style of elements that are currently in focus and :focus-within sets the style of elements that contain a focusable element.

This is just a placeholder

This is just placeholder text. Don't be alarmed, this is just here to fill up space since your finalized copy isn't ready yet.

Call to action

In the card above, we need to practice fixing the focus style on the button. Then we need to see if we can get the entire card to be clickable and focusable.

Structure

Container

Rich Text

Rich Text

Container

Container

Container

Container

Rich Text


Rich Text

Container

Image

Image

Image



Image

42:26 [Icons]

b

Page settings

GENERAL

ONE PAGE NAVIGATION

• CUSTOM CODE

• Custom CSS

```
1 .brxe-container:has(> figure:nth-child(2)) {  
2   border: 5px solid var(--danger);  
3 }  
4
```

```
5 :where(nav li, .include li) {  
6   color: var(--danger);  
7 }  
8  
9 section:not(:last-child) {  
10  border-bottom: 1px solid #ddd;  
11 }
```

```
12  
13 h2 {  
14   width: 38ch;  
15 }
```

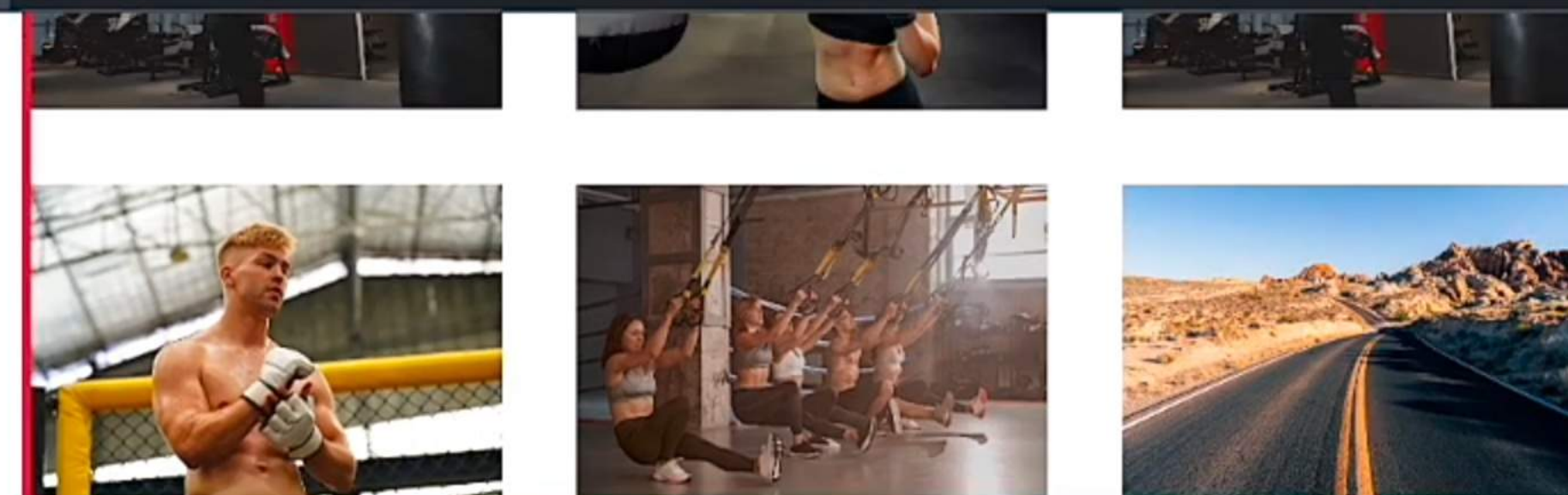
```
16  
17 .brxe-text ul, .brxe-text ol {  
18   margin: 0;  
19   padding: 0;  
20   padding-inline-start: 1em;  
21 }
```

```
22  
23 .brxe-text ul li:not(:last-child), ol li:not(:last-child) {  
24   margin-bottom: 10px;  
25 }
```

Adds inline CSS to <head> tag.

Header scripts

Now we're selecting parents who have a figure tag in position 2



You can even combine `:has()` and `:nth()`. For example, you can select any container that has a direct child figure element in the 2nd position `:has(>figure:nth-child(2))`

Just remember: `:has()` is still waiting on full browser support (but it's very close!).

Rich Text

Structure

Container

Rich Text

Rich Text

Container

Container

Container

Container

Rich Text

Rich Text

Container

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Keyboard users are important, too. `:focus` sets the style of elements that are currently in focus and `:focus-within` sets the style of elements that contain a focusable element.

This is just a placeholder

This is just placeholder text. Don't be alarmed, this is just here to fill up space since your finalized copy isn't ready yet.

Call to action

In the card above, we need to practice fixing the focus style on the button. Then we need to see if we can get the entire card to be clickable and focusable.



43:05

PB101- L18 - Programmatic Styling With Pseudo Classes (Critical).mp4

Pseudo Classes (Builder)

Pseudo Classes – Bricks Clean

Pseudo Classes – Bricks Clean

Not Secure | bricks-clean.local/psuedo-2?bricks_preview=1694793266

Facebook

Hey!

Business

Personal

Hosting

Resources

StreamYard

Basecamp

Automatic.css

Frames

Hubspot

HelpScout

Genius

InnerCircle

ACSS

UserMaven

JS

Softball

Pseudo Classes (...)

Bricks Clean

Customize

0

New

Edit Page

Edit with Bricks

Render with Bricks

Automatic CSS

Howdy, admin

figure element in the 2nd position :has(>figure:nth-child(2))

Just remember: :has() is still waiting on full browser support (but it's very close!).

Don't forget accessibility with :focus and :focus-within

Keyboard users are important, too. :focus sets the style of elements that are currently in focus and :focus-within sets the style of elements that contain a focusable element.

This is just a placeholder

This is just placeholder text. Don't be alarmed, this is just here to fill up space since your finalized copy isn't ready yet.

Call to action

This button is navigated with the keyboard but the focus color is not visible

In the card above, we need to practice fixing the focus style on the button. Then we need to see if we can get the entire card to be clickable and focusable.

The final one I'm going to cover in this lesson is :empty()

bricks-clean.local/psuedo-2?bricks_preview=1694793266#

45:47

Pseudo Classes (Builder) x Pseudo Classes - Bricks Clean x Pseudo Classes - Bricks Clean x +

Not Secure | bricks-clean.local/psuedo-2/?bricks=run

Facebook Hey! Business Personal Hosting Resources StreamYard Basecamp Automatic.css Frames Hubspot HelpScout Geni.us InnerCircle ACSS UserMaven JS Softball Pseudo Classes (...)

#brxe-28f939 2

.fr-cta-card-alpha__button .btn--action

CONTENT STYLE

CSS Filters

[Learn more about CSS filters](#)

Transition

all 0.3s ease 0s

[Learn more about CSS transitions](#)

Custom CSS

```
1 root:focus {
2   --focus-color: black;
3 }
```

Use "root" to target the element wrapper: root { background: blue }

CSS classes

Separated by space. Without class dot.

CSS ID

Other pseudo classes we didn't cover:

- :active
- :visited
- :enabled
- :disabled
- :root
- a bunch related to forms
- and more...

Structure

- Container
 - Heading
 - Rich Text
- Container
 - Rich Text
 - Rich Text
 - Rich Text
- Container
 - Heading
 - Rich Text
- Container
 - Image
 - Image
 - Image
- CTA Card Alpha

Because this site is using ACSS we can set a new value for the variable 'focus-color'

46:57

Not Secure | bricks-clean.local/psuedo-2/?bricks_preview=1694793266

FacebookHey!BusinessPersonalHostingResourcesStreamYardBasecampAutomatic.cssFramesHubspotHelpScoutGeniusInnerCircleACSSUserMavenJSSoftballPseudo Classes (...)

Elements>>⚙️⋮✕

card-alpha__content-wrapper">

⋮</div>flex

▼<div class="brxe-block f a-card-alpha__action-wrapper">

⋮flex

⋮Call to actionflex == \$0

⋮</div>

⋮</div>

⋮</div>

▶<div class="brxe-container">⋮

◀a#brxe-28f939.brxe-button fr-cta-card-alpha▶

StylesComputedLayout>>

Filter: :hov .cls +🔍📄🔧

Force element state

☐ :active

☐ :focus

☐ :focus-within

☐ :target

☐ :hover

☐ :visited

☐ :focus-visible

element.style {

}

.btn--action { automatic-b...4785612:359

--btn-background: var(--action);

--btn-background-hover: var(--action-hover);

--btn-text-color: var(--action-ultra-light);

--btn-text-color-hover: var(--action-ultra-light);

⋮ConsoleWhat's New ✕Issues✕

Bricks CleanCustomize0+NewEdit PageEdit with BricksRender with BricksAutomatic CSS

Howdy, admin🔍

This is just a placeholder

This is just placeholder text. Don't be alarmed, this is just here to fill up space since your finalized copy isn't ready yet.

Call to action

In the card above, we need to practice fixing the focus style on the button. Then we need to see if we can get the entire card to be clickable and focusable.

The final one I'm going to cover in this lesson is :empty()

Sometimes, when conditions, dynamic data, etc. are used, a container/wrapper may end up empty. In this case, we may want to style the empty container a certain way, or hide it altogether. The below grid has an empty cell. Let's style it, and then practice hiding it.

Not empty

Not empty

Not empty

47:43

card-alpha__content-wrapper">

flex

<div class="brxe-block f

card-alpha__action-wrapper">

flex

<a id="brxe-28f939" class="br

xe-button fr-cta-card-alpha__

button btn--action bricks-but

ton" href="#">Call to action

flex == \$0

</div>

</div>

</div>

<div class="brxe-container">

a#brxe-28f939.brxe-button.fr-cta-card-alpha

Styles

Computed

Layout

>>

Filter

:hov

.cls

+

🔍

🔧

Force element state

☐ :active

☐ :hover

☒ :focus

☐ :visited

☐ :focus-within

☐ :focus-visible

☐ :target

element.style {

}

#brxe-28f939: focus

--focus-color: black;

body.bricks-is-frontend

:focus:not(:focus-visible) {

⋮

Console

What's New

Issues

PB101- L18 - Programmatic Styling With Pseudo Classes (Critical).mp4

Pseudo Classes (Builder)

Pseudo Classes - Bricks Clean

Pseudo Classes - Bricks Clean

Not Secure | bricks-clean.local/psuedo-2?bricks_preview=1694793266

Facebook

Hey!

Business

Personal

Hosting

Resources

StreamYard

Basecamp

Automatic.css

Frames

Hubspot

HelpScout

Genius

InnerCircle

ACSS

UserMaven

JS

Softball

Pseudo Classes (...)

Bricks Clean

Customize

0

New

Edit Page

Edit with Bricks

Render with Bricks

Automatic CSS

Howdy, admin

TIME SENSITIVE

15

Busy Block

Today at 12:30 PM

23m ago

This is just a placeholder

This is just placeholder text. Don't be alarmed, this is just here to fill up space since your finalized copy isn't ready yet.

Call to action

In the card above, we need to practice fixing the focus style on the button. Then we need to see if we can get the entire card to be clickable and focusable.

The final one I'm going to cover in this lesson is :empty()

Sometimes, when conditions, dynamic data, etc. are used, a container/wrapper may end up empty. In this case, we may want to style the empty container a certain way, or hide it altogether. The below grid has an empty cell. Let's style it, and then practice hiding it.

Not empty

Not empty

Not empty

47:50

🔊

🔍

Don't forget accessibility with :focus and :focus-within

Keyboard users are important, too. :focus sets the style of elements that are currently in focus and :focus-within sets the style of elements that contain a focusable element.

This is just a placeholder

This is just placeholder text. Don't be alarmed, this is just here to fill up space since your finalized copy isn't ready yet.

Call to action

The parent has a box shadow now

In the card above, we need to practice fixing the focus style on the button. Then we need to see if we can get the entire card to be clickable and focusable.

The final one I'm going to cover in this lesson is :empty()



Not empty Not empty Not empty

Note that in this situation, even though we've hidden the empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use `:has()` to select a grid that has an empty child and remove a cell.

Other pseudo classes we didn't cover:

- `:active`
- `:visited`
- `:enabled`
- `:disabled`
- `:valid`
- `:invalid`
- a bunch related to forms
- and more...

Get rid of the box shadow for the button

```
1 root:focus-within {
2   box-shadow: 0 0 0 5px black;
3 }
4
5 root:focus-within :focus {
6   box-shadow: none;
7 }
```


The final one I'm going to cover in this lesson is :empty()

Sometimes, when conditions, dynamic data, etc. are used, a container/wrapper may end up empty. In this case, we may want to style the empty container a certain way, or hide it altogether. The below grid has an empty cell. Let's style it, and then practice hiding it.

Not empty

Not empty

Not empty

Note that in this situation, even though we've hidden the empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use :has() to select a grid that has an empty child and remove a cell.



Container

#brxe-oicvvi

.grid--4

CONTENT

STYLE

LAYOUT

TYPOGRAPHY

BACKGROUND

BORDER / BOX SHADOW

GRADIENT / OVERLAY

SHAPE DIVIDERS

TRANSFORM

CSS

CSS Filters

Learn more about CSS filters

Transition

width 0s ease 0s

Learn more about CSS transitions

Custom CSS

1

The final one I'm going to cover in this lesson is :empty()

Sometimes, when conditions, dynamic data, etc. are used, a container/wrapper may end up empty. In this case, we may want to style the empty container a certain way, or hide it altogether. The below grid has an empty cell. Let's style it, and then practice hiding it.

Not empty	Not empty	Not empty	
-----------	-----------	-----------	--

Note that in this situation, even though we've hidden the empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use :has() to select a grid that has an empty child and remove a cell.

Other pseudo classes we didn't cover:

- :active
- :visited
- :enabled
- :disabled

Structure

Container

Rich Text

Rich Text

Rich Text

Container

Heading

Rich Text

Container

Image

Image

Image

Image

Image

Image

Image

Rich Text


Rich Text

Section

Container

Container

The 4th col is empty



53:28

The final one I'm going to cover in this lesson is `:empty()`

Sometimes, when conditions, dynamic data, etc. are used, a container/wrapper may end up empty. In this case, we may want to style the empty container a certain way, or hide it altogether. The below grid has an empty cell. Let's style it, and then practice hiding it.

Not empty	Not empty	Not empty
-----------	-----------	-----------

Note that in this situation, even though we've hidden the empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use `:has()` to select a grid that has an empty child and remove a cell.

STYLE

[Learn more about CSS transitions](#)

- Custom CSS

```
1 root :empty {
2   background-color: var(--danger) ;
3 }
```

CSS classes

Separated by space. Without class dot.

CSS ID

No spaces. No pound (#) sign.

☐ Button

The final one I'm going to cover in this lesson is `:empty()`

It's not shown anymore
but still in the DOM

Not empty Not empty Not empty

Note that in this situation, even though we've hidden the empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use `:has()` to select a grid that has an empty child and remove a cell.

Other pseudo classes we didn't cover:

☐ Button

Not Secure | bricks-clean.local/psuedo-2/?bricks_preview=1694793266

FacebookHey!BusinessPersonalHostingResourcesStreamYardBasecampAutomatic.cssFramesHubspotHelpScoutGeniusInnerCircleACSSUserMavenJSSoftballPseudo Classes (...)

Elements

>>

<section class="brxe-section">

flex

<div class="brxe-container">

flex

<div id="brxe-oicvvi" class="brxe-container grid--4">

grid

<div class="brxe-block">

flex

<div class="brxe-block">

flex

<div class="brxe-block">

flex

<div class="brxe-block bricks-lazy-hidden"></div> == \$0

</div>

r.grid--4

div.brxe-block.bricks-lazy-hidden

Styles

Computed

Layout

>>

Filter

:hov .cls

+

🔍

🔧

element.style {

}

#brxe-oicvvi

post-1131.m...694797212:1

:empty {

display: none;

}

.brxe-block.bricks-lazy-hidden {

frontend-li...694784691:1

background-image: none !important;

}

[class*=brxe-] {

frontend-li...694784691:1

max-width: 100%;

}

.brxe-block {

frontend-li...694784691:1

Console

What's New

Issues

Bricks Clean

Customize

0

New

Edit Page

Edit with Bricks

Render with Bricks

Automatic CSS

Howdy, admin

TIME SENSITIVE

15

Busy Block

Today at 12:30 PM

30m ago

The final one I'm going to cover in this lesson is :empty()

Sometimes, when conditions are met, this case, we may want to use :empty() to hide a cell that has an empty cell. Let's style it, and then practice hiding it.

It's still here as a DOM element but no longer selectable and it's no longer taking space (in opposite to 'visibility: hidden')

Not empty

Not empty

Not empty

Note that in this situation, even though we've hidden the empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use :has() to select a grid that has an empty child and remove a cell.

Keep in mind that we defined a 4 col grid! And col 4 is still there!

Sometimes, when conditions, dynamic data, etc. are used, a container/wrapper may end up empty. In this case, we may want to style the empty container a certain way, or hide it altogether. The below grid has an empty cell. Let's style it, and then practice hiding it.

Not empty Not empty Not empty

Other pseudo classes we didn't cover:

No spaces. No pound (#) sign.

Image

Button

☐ Button

The final one I'm going to cover in this lesson is `:empty()`

Not empty	Not empty	Not empty
-----------	-----------	-----------

4 col grid to a 3 col grid! empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use `:has()` to select a grid that has an empty child and remove a cell.

Other pseudo classes we didn't cover:

```
.grid-4
```

CONTENT

STYLE

Transition

[Learn more about CSS transitions](#)

- Custom CSS

```
1 .grid--4 :empty {
2     display: none;
3 }
4
5 .grid--4:has(> :empty) {
6     grid-template-columns: var(--grid-3);
7 }
```

Use "root" to target the element wrapper: `root { background: blue }`

CSS classes

Separated by space. Without class dot.

CSS ID

No spaces. No pound (#) sign.

☐ Button

◀ r.grid--4 div.brx-block.bricks-lazy-hidden

Styles Computed Layout >>

Filter :hov .cls +

```

element.style {
}

.grid--4 :empty {      post-1131.m...694797507:1
  display: none;
}

.brx-frontend-li...694784691:1
block.bricks-
lazy-hidden {
  background-image: none !important;
}

[class*=brxe-] {      frontend-li...694784691:1
  max-width: 100%;
}

.brx-block {          frontend-li...694784691:1
  align-items: flex-start; ⓘ

```

⋮ Console What's New × Issues ×

Sometimes, when conditions, dynamic data, etc. are used, a container/wrapper may end up empty. In this case, we may want to style the empty container a certain way, or hide it altogether. The below grid has an empty cell. Let's style it, and then practice hiding it.

Not empty

Note that in this situation, even though we've hidden the empty div, the grid is still set to be a 4 column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use `:has()` to select a grid that has an empty child and remove a cell.



#brxe-oicvvi 1
grid--4

CONTENT

STYLE

Transition

width 0s ease 0s

[Learn more about CSS transitions](#)

• Custom CSS

```
1 .grid--4 :empty {  
2   display: none;  
3 }  
4  
5 .grid--4:has(> :empty) {  
6   grid-template-columns: var(--grid-3);  
7 }
```

Use "root" to target the element wrapper: root { background: blue }

CSS classes

Separated by space. Without class dot.

CSS ID

No spaces. No pound (#) sign.

column grid. The grid doesn't know the last element is empty, so it can't change to adapt. This is a situation where we might want to use `:has()` to select a grid that has an empty child and remove a cell.

Other pseudo classes we didn't cover:

- `:active`
- `:visited`
- `:enabled`
- `:disabled`
- `:valid`
- `:invalid`
- `:root`
- a bunch related to forms
- and more...

Structure

Container

Rich Text

Rich Text

Rich Text

Heading

Container

Heading

Rich Text

Container

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image

Image



Button

Pseudo Classes (Builder) x Pseudo Classes - Bricks Clean x Pseudo Classes - Bricks Clean x +

Not Secure | bricks-clean.local/psuedo-2/?bricks=run

Facebook Hey! Business Personal Hosting Resources StreamYard Basecamp Automatic.css Frames Hubspot HelpScout Genius InnerCircle ACSS UserMaven JS Softball Pseudo Classes (...)

Container #brxe-uoxlzn

CONTE... STYLE

LAYOUT >

TYPOGRAPHY >

BACKGROUND >

BORDER / BOX SHADOW >

GRADIENT / OVERLAY >

SHAPE DIVIDERS >

TRANSFORM >

CSS >

ATTRIBUTES >

Other pseudo classes we didn't cover:

- :active
- :visited
- :enabled
- :disabled
- :valid
- :invalid
- :root
- a bunch related to forms
- and more...

Here you create your global variables

Structure

- Container
 - Image
 - Image
 - Image
 - Image
 - Image
 - Image
 - Image
- Rich Text
- Rich Text
- Section
 - Container
 - Container
 - CTA Card Alpha
 - Content Wrapper
 - Action Wrapper

Section

1:00:44