BRO Academy

Decorative Styling

Lesson 9

Lesson Plan

1
What is included in decorative styling and why we need it.

2
Pseudo-elements vs Pseudo-classes.

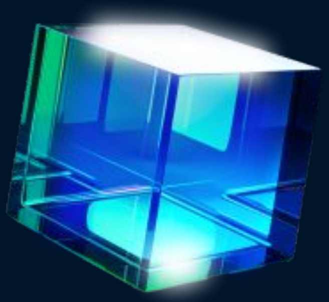
3
Shadows and Filters.

4
How to define shape – mask, clip-path, shape-outside.

Pseudo-classes

Help style specific attributes or states that are not reflected in the DOM.

- **User-action** pseudo-classes
- **The lang** pseudo-class
- **The negation** pseudo-class
- **Structural** pseudo-classes
- **User interface** pseudo-class selectors



Link Pseudo-classes for `<a>`

`:link` – styles unvisited links

`:visited` – styles visited links

```
a:link { color: blue; }

a:visited { color: purple; }
```

User-action pseudo-classes

`:active` – element is being clicked

`:focus` – element is in focus

`:hover` – mouse is hovering over the element

`a, button, input`

The `lang` pseudo-class

Applied to elements with the `lang` attribute

```
p:lang(fr) { font-style: italic; }


<p lang="fr">
   Adieu
</p>
<p lang="jw">
   Sugeng rawuh
</p>
```

The negation pseudo-class

Styles are applied to all elements except those matching the selector

`:not(p) { }` – all elements except paragraphs **tags**

`:not(.intro) { }` – all elements except those **with class** .intro

`:not(#news) { }` – all elements except those **with id** #news

`:not(:lang(fr)) { }` – all elements **except** those with the French **language**

`:not([disabled]) { }` – all elements except those **without** the disabled **attribute**

`p:not(.intro) { }` – all paragraphs **except** those with the class `.intro`

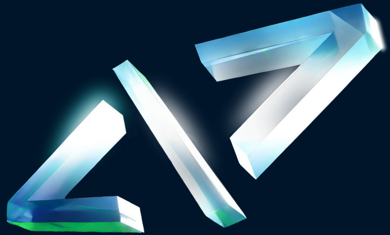
Structural pseudo-classes

Allow you to select elements based on their position in the document structure.

! If the document structure changes, the structural pseudo-class might apply to a different element or potentially to no element at all.

It can sometimes be difficult to determine exactly which element the styles will be applied to.

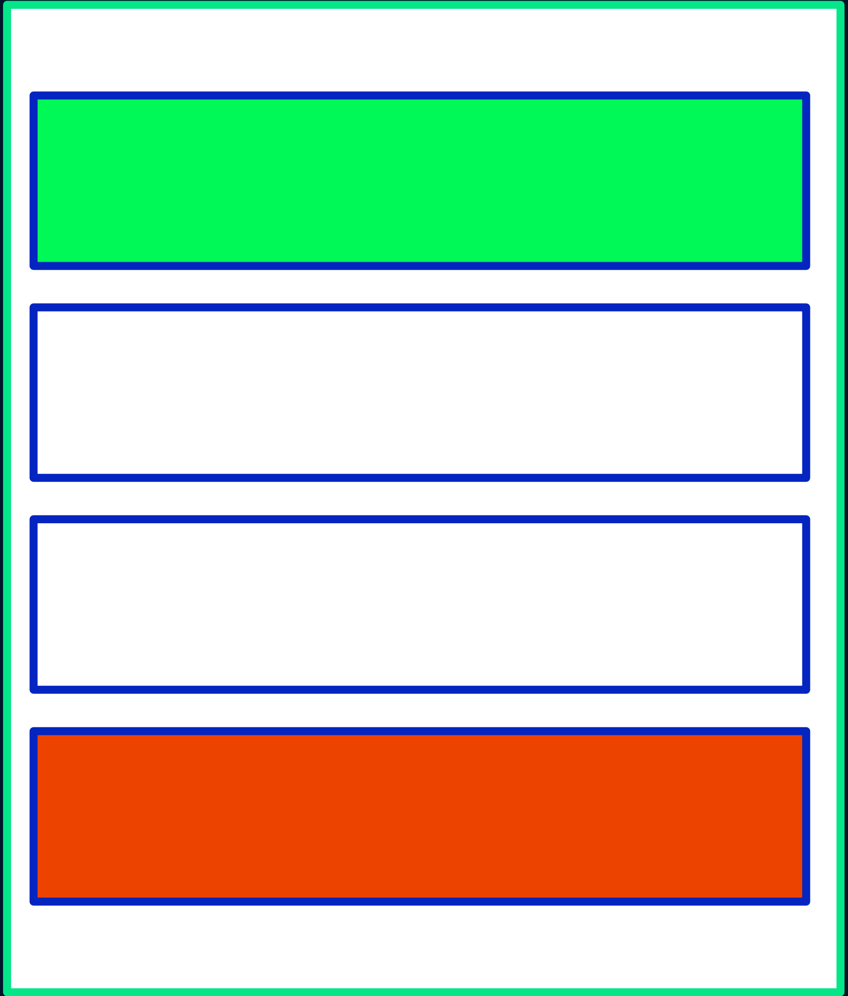
```
:first-child { }
:only-child { }
:nth-child(3n) { }
```

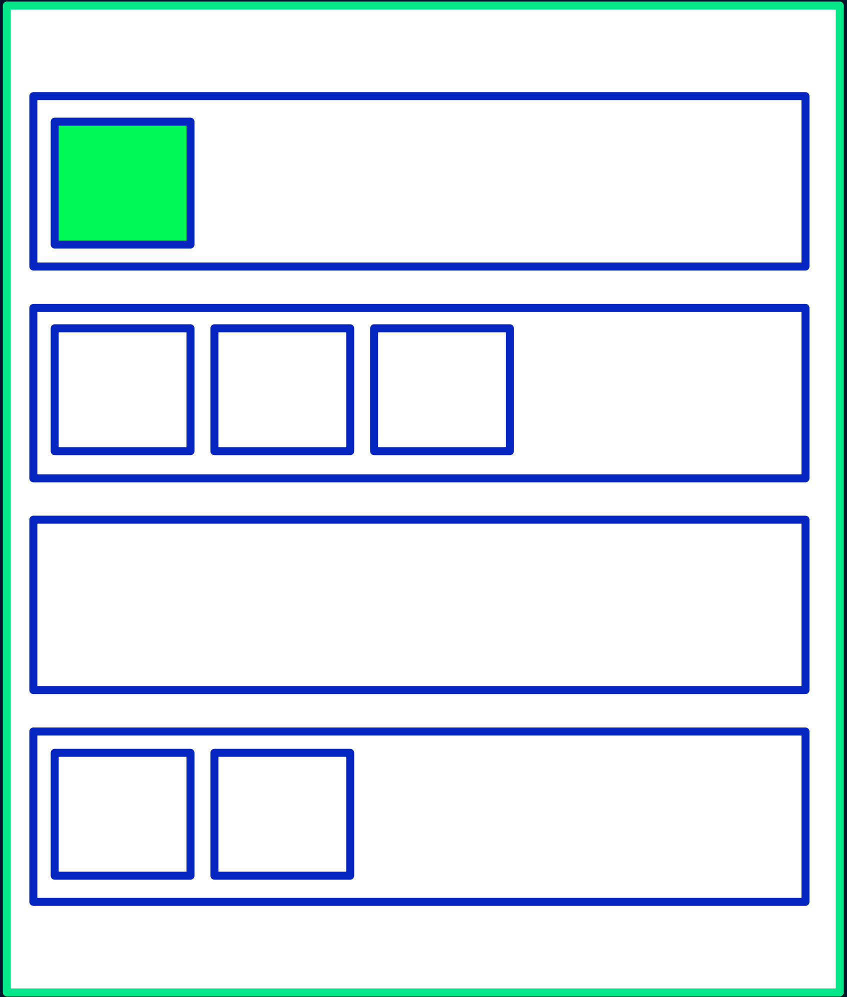


:first-child
:last-child

– Selects the element that is the first/last child of another element.

```
article:first-child { }
article:last-child { }

<section>
   <article> 1 </article>
   <article> 2 </article>
   <article> 3 </article>
   <article> 4 </article>
</section>
```



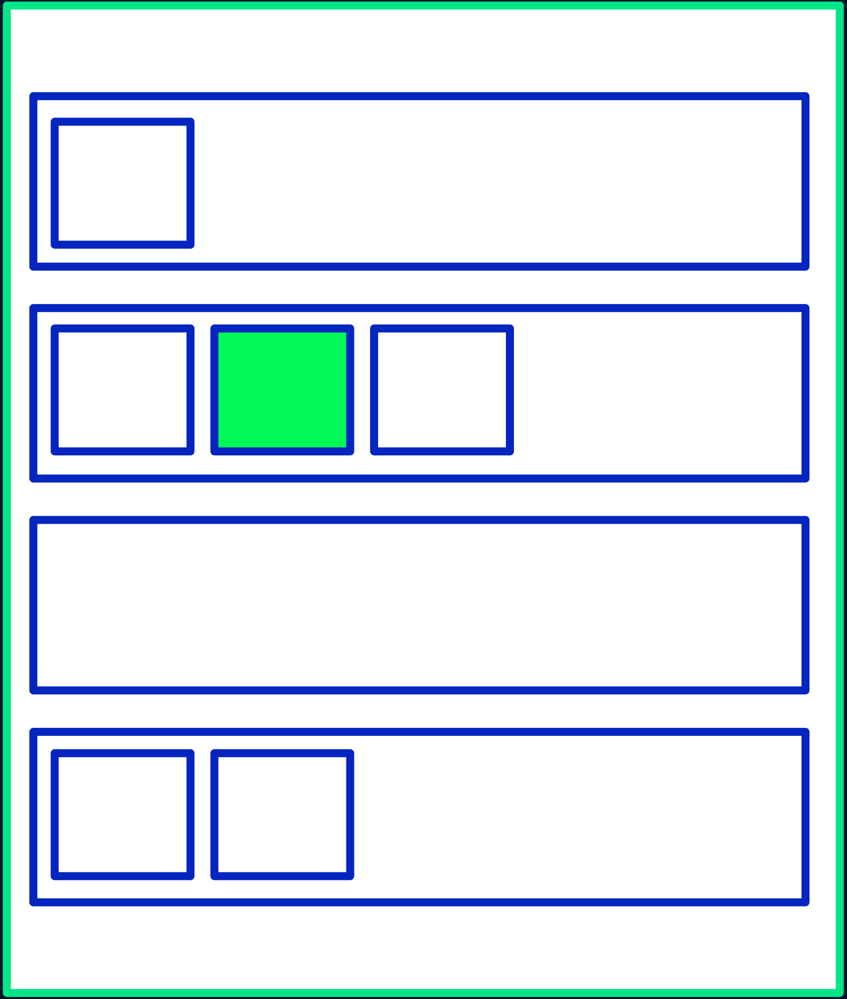


pseudo-class

:only-child

– Selects an element that is the only child of another element.

```
div:only-child { }

<article>
   <div> 1 </div>
</article>
```



pseudo-class

:only-of-type

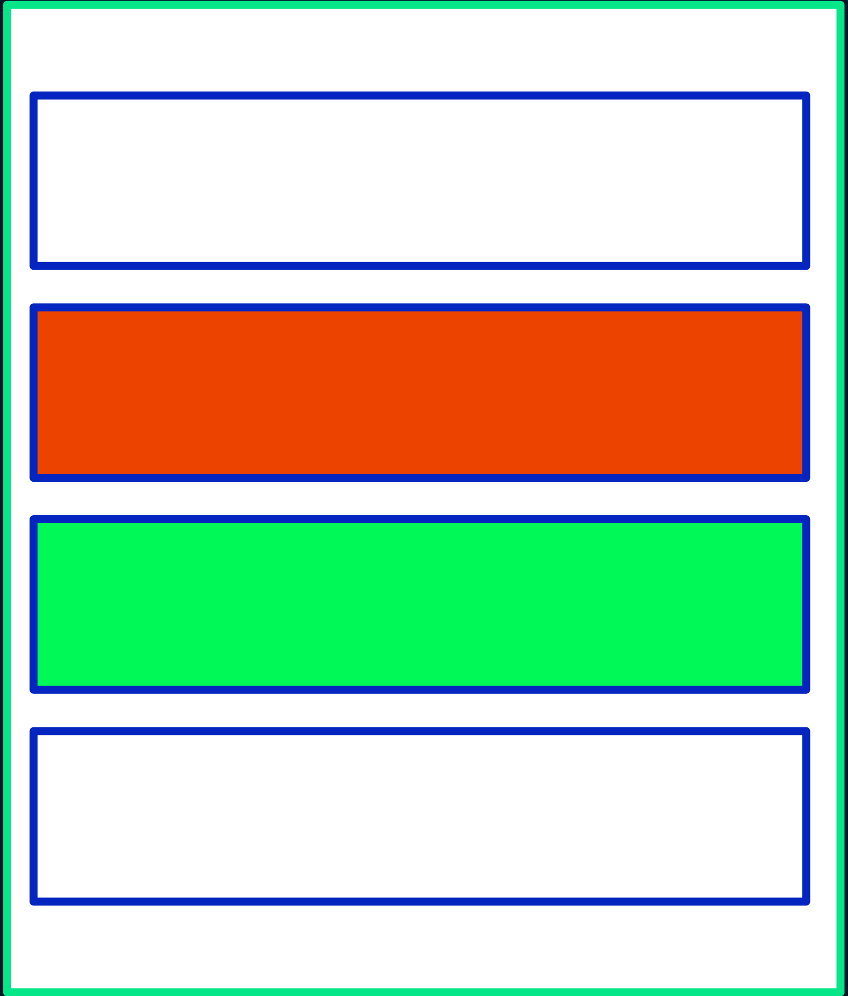
– Selects an element that is the **only** element of its **type** within its parent.

```
p:only-of-type { }

<article>
   <div> 1 </div>
   <p> 1 </p>
   <div> 1 </div>
</article>
```

:first-of-type
:last-of-type

– Selects an element that is the first/last of its type within its parent element.

```
p:first-of-type { }
p:last-of-type { }

<section>
   <article> 1 </article>
   <p> 2 </p>
   <p> 3 </p>
   <article> 4 </article>
</section>
```

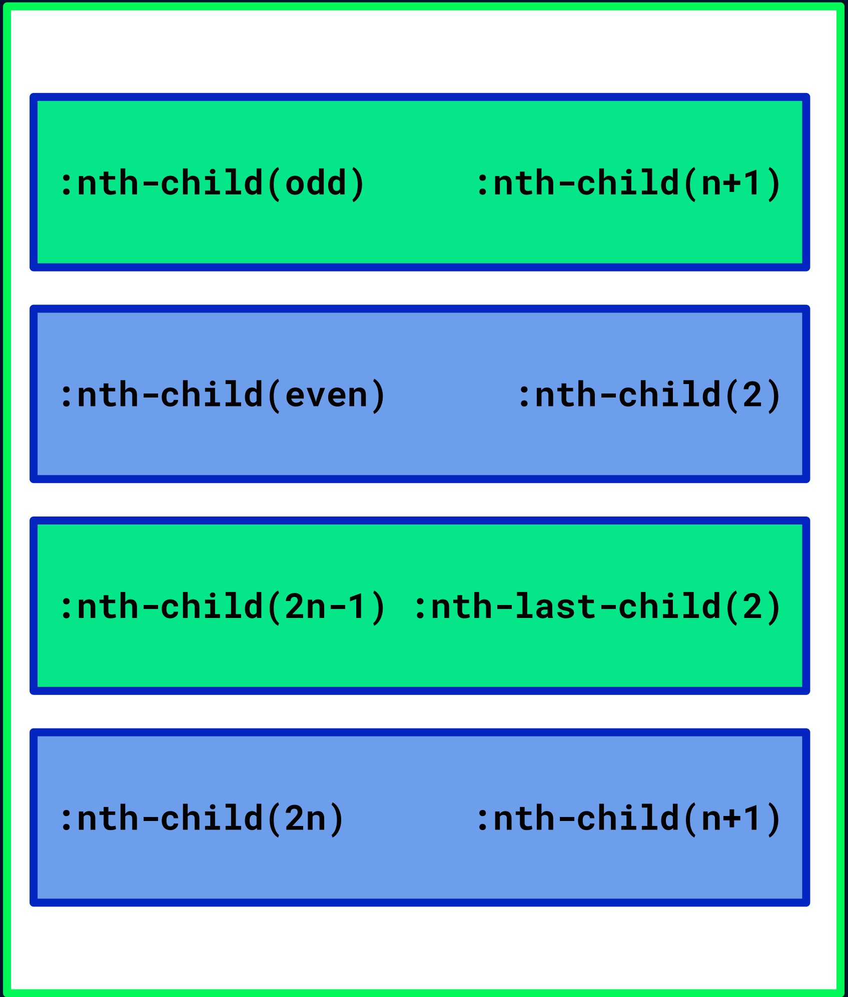


:nth-child(n)
:nth-last-child(n)

– Selects specific child elements in a parent element starting from the beginning or the end.

n:

- **number**
- **number + n** (selects every **n-th** element)
- expression with **+/-** (allows starting from an element other than the first)
- **even** (all even elements)
- **odd** (all odd elements)

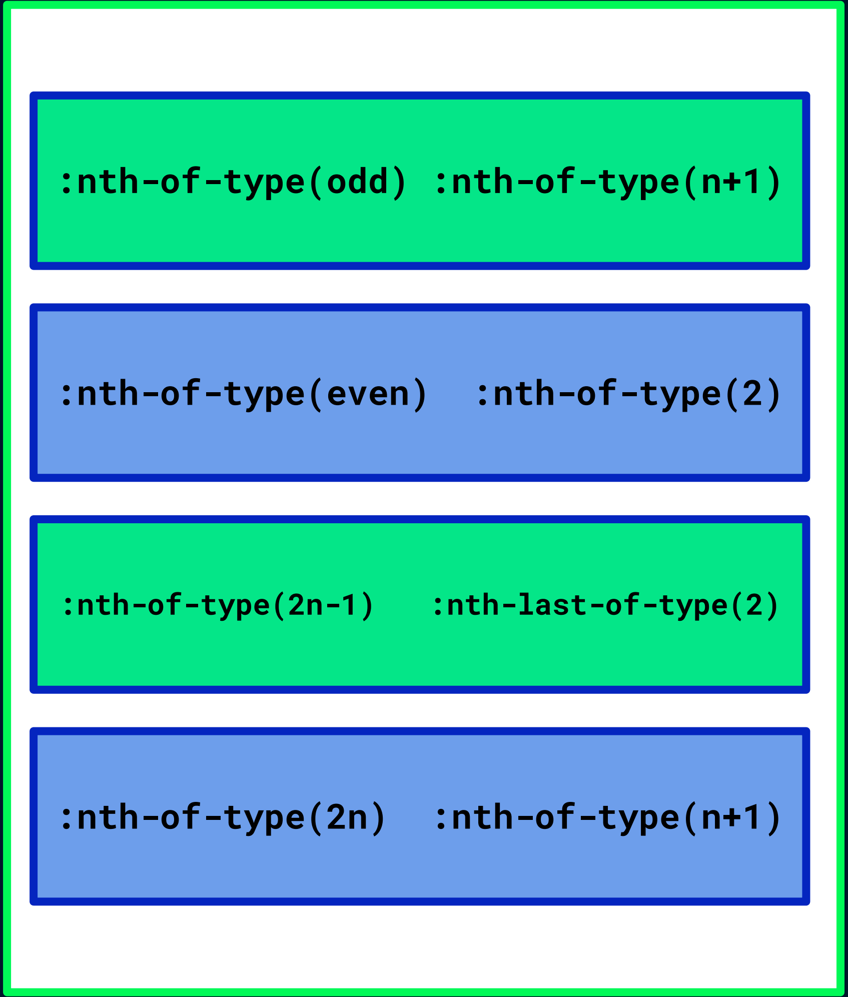


:nth-of-type(n)
:nth-last-of-type(n)

– Selects elements of a specific type in the parent element starting from the beginning or the end.

n:

- number
- number + n (selects every n-th element)
- expression with +/- (allows starting from an element other than the first)
- even (all even elements)
- odd (all odd elements)


:nth-of-type(odd) :nth-of-type(n+1)
:nth-of-type(even) :nth-of-type(2)
:nth-of-type(2n-1) :nth-last-of-type(2)
:nth-of-type(2n) :nth-of-type(n+1)

:root

– Selects the root element of the document (tag `<html>`).

:empty

– Selects an element that has no content or child elements (an empty element).

A space is already a character, so the tag is no longer considered empty.
It also applies to input elements where no value has been entered.

```
:root { }

<html>
   <head> 1 </head>
   <body> 1 </body>
</html>


p:empty { }

<article>
   <p> 1 </p>
   <p> </p>
   <p></p>
   <p><span></span></p>
</article>
```

Pseudo-elements

– **(fake elements)** Allow styling elements that are not in the document tree.

`::-webkit-scrollbar` – styles the scrollbar

+ Other pseudo-elements of the form `::-webkit-scrollbar-*`, are used only with prefixes and only in `webkit` browsers

```css
.invisible-scrollbar::-webkit-scrollbar {
  display: none;
}
```

Pseudo-elements for text

`::first-line` – styles the first line of text

`::first-letter` – styles the first letter of text

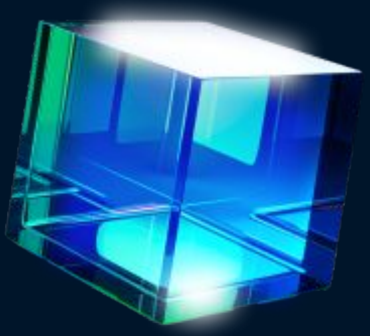
```
p::first-line { }
p:first-letter { }
<p>
   This is the first line
   of a paragraph of text
</p>
```

Pseudo-elements for Lists

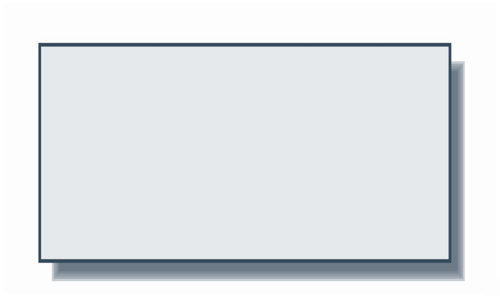
- Usage of counters in lists
- Styling list markers

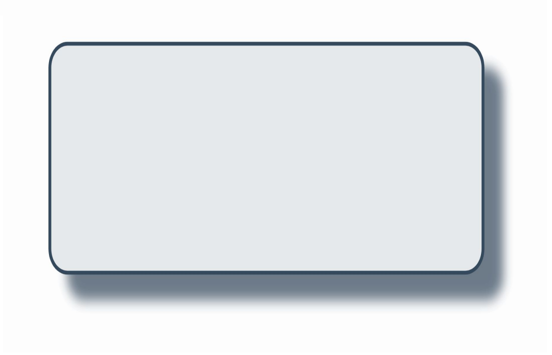
`::marker` – Styling list markers.

```
ol {
  counter-reset: section;
}
li::before {
  counter-increment: section;
  content: counter(section);
}
li::marker { }
```



BOX-SHADOW





```
p { box-shadow: red 5px 8px 15px 18px inset; }
```

External shadows

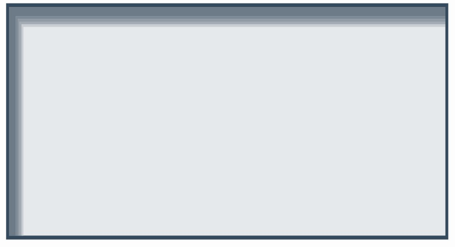
– Placed outside the `border`

– Mimics the shape of the block, including rounded corners, etc.

Internal shadows (inset)

– Placed above `background-images`, `background-color`

– but below content



Syntax box-shadow

```
p { box-shadow: inset 5px 8px 15px 18px red; }
```

`offset-x, offset-y, blur-radius, spread-radius` – order matters

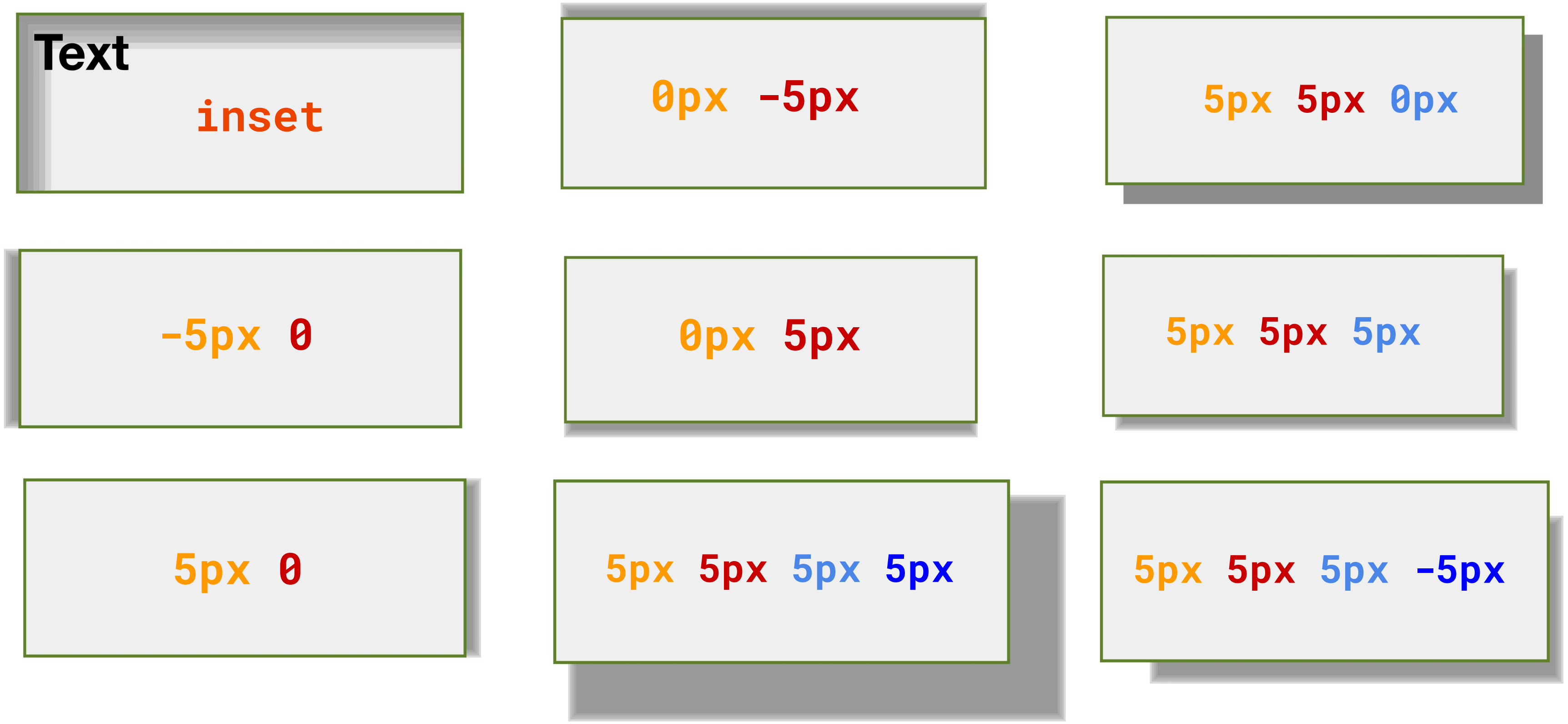
`inset`
internal shadow
 	
optional
 	
order does not matter
	`offset-x`
	horizontal offset
	
	positive = right,
	negative = left,
	0 = no offset
		`offset-y`
		vertical offset
		
		positive = down,
		negative = up,
		0 = no offset

`blur-radius` optional, default is 0 which means sharp edges, values only greater than 0	**`spread-radius`** optional, default is 0, increases or decreases the size of the shadow	**`color`** optional, if not specified, it inherits the element's color order does not matter

```
p { box-shadow: inset offset-x, offset-y, blur-radius, spread-radius color; }
```

examples

box-shadow



Multiple box-shadow

can be applied to a single element by separating each shadow with a comma:

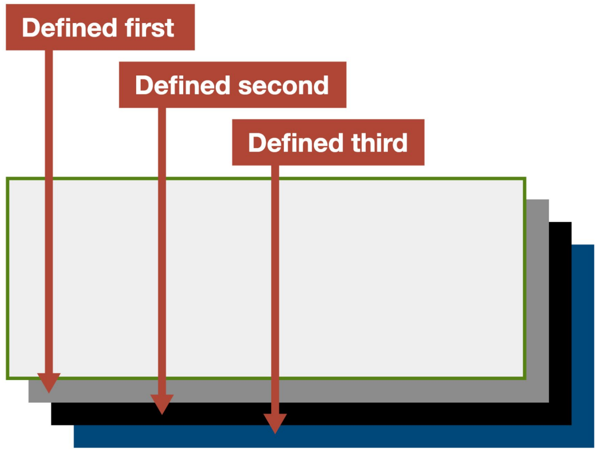
```
p { box-shadow: inset 10px 10px 10px 10px red,     /* all possible values specified */
                red 10px 10px 10px 10px inset,     /* different order */
                10px 10px 10px 10px red,           /* no inset specified */
                10px 10px 10px 10px,               /* no color specified */
                10px 10px 10px,                    /* no spread-radius specified */
                10px 10px; }                       /* no blur-radius specified */
```

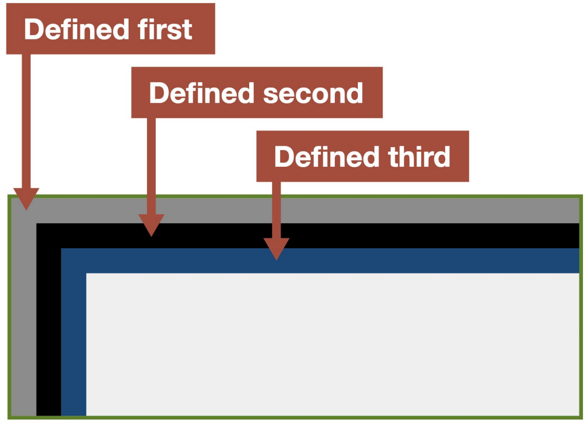
order in `box-shadow`

The order of the shadows matters

```
p { box-shadow: first, second, third; }
```

the first shadow in the list will be on top, and subsequent shadows will be layered below it.





TEXT-SHADOW

- Only external shadows.
- Follows the shape of the text.
- Multiple shadows can be applied.
- Syntax is similar to **box-shadow** :

```
offset-x
offset-y
blur-radius
color
```

```
p {
  text-shadow: 2px 2px 8px #FF0000;
}
```

Text
HELLO
FIELDING
ELEGANT
SHADOW
Funky
Fresh
SHADOWS

filter

– Applies visual effects to elements

(like in instagram)

```
filter: blur(5px);
filter: contrast(175%) brightness(3%);
```

Values:

- **Keyword** – blur
- **url** – url("filters.svg#filter-id")

filter

`blur(px)`

`brightness(0-1)`

`contrast(%)`

`drop-shadow(x y blur color)` – inner shadow

`grayscale(%)` – black and white

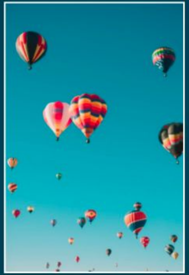
`hue-rotate(deg)` – shifts the color palette around the color wheel

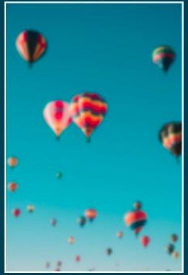
`invert(%)`

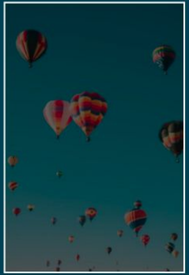
`opacity(%)`

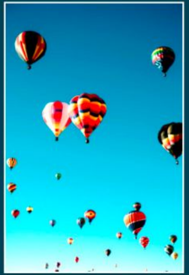
`saturate(%)`

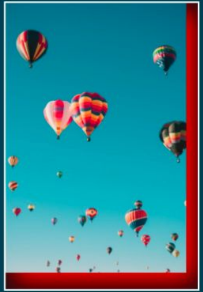
`sepia(%)` – like vintage photo

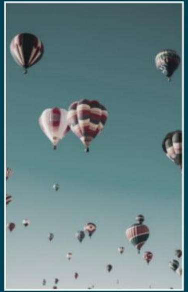

No Filter Applied

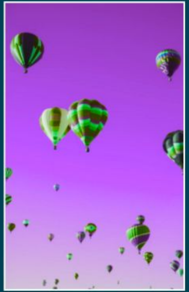

filter: blur(2px);

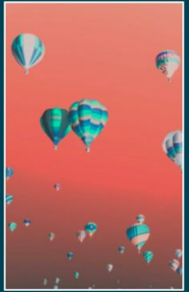

filter: brightness(0.4);

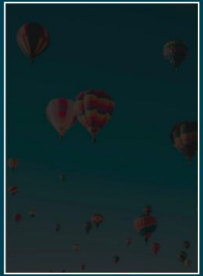

filter: contrast(200%);

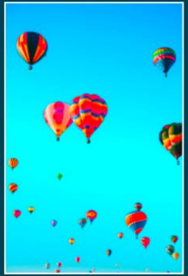

filter: drop-shadow(16px red);

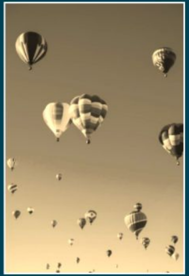

filter: grayscale(80%);


filter: hue-rotate(90deg);

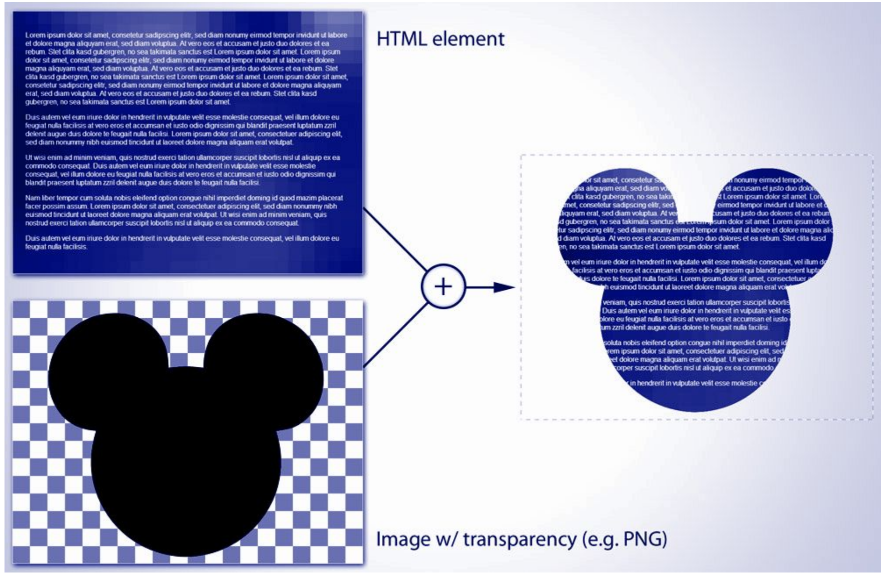

filter: invert(85%);


filter: opacity(15%);


filter: saturate(400%);


filter: sepia(560%);

CSS mask



Imagine cutting out a circle in a sheet of paper and placing it over a picture; you've applied a mask.

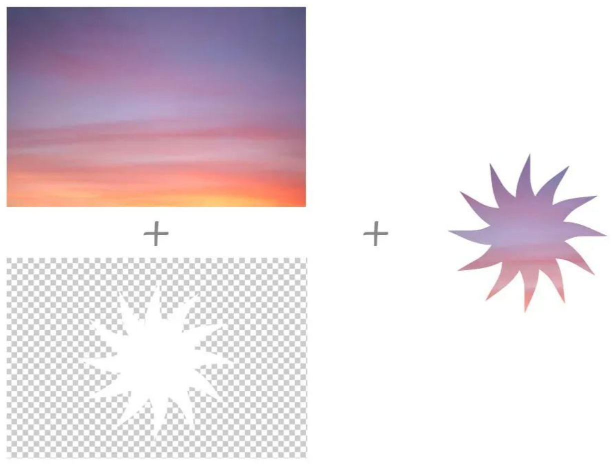
Used to create complex shapes for elements.

Masks operate based on the alpha channel:
- ➔ **Black –** full invisibility
- ➔ **White –** full visibility
- ➔ **Gray –** partial transparency

CSS mask

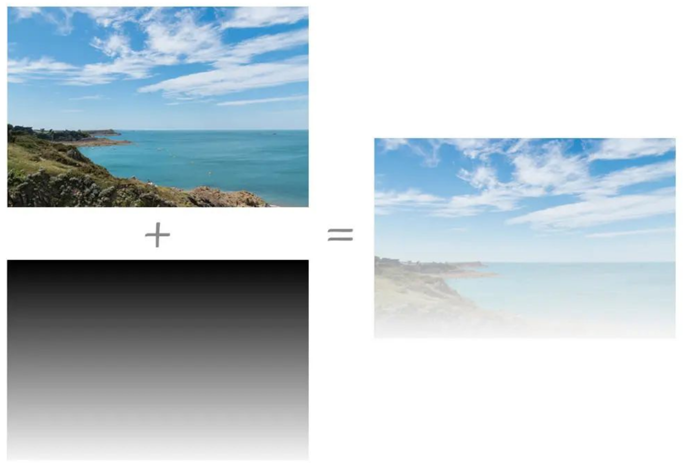
Image:

```
mask: url(mask.png);
```



Gradient:

```
mask: linear-gradient(from, to);
```



CSS mask properties

`mask-image` – the image used as the mask

`mask-mode` – chooses the mask based on transparent or opaque areas

`mask-position` – mask position relative to the element

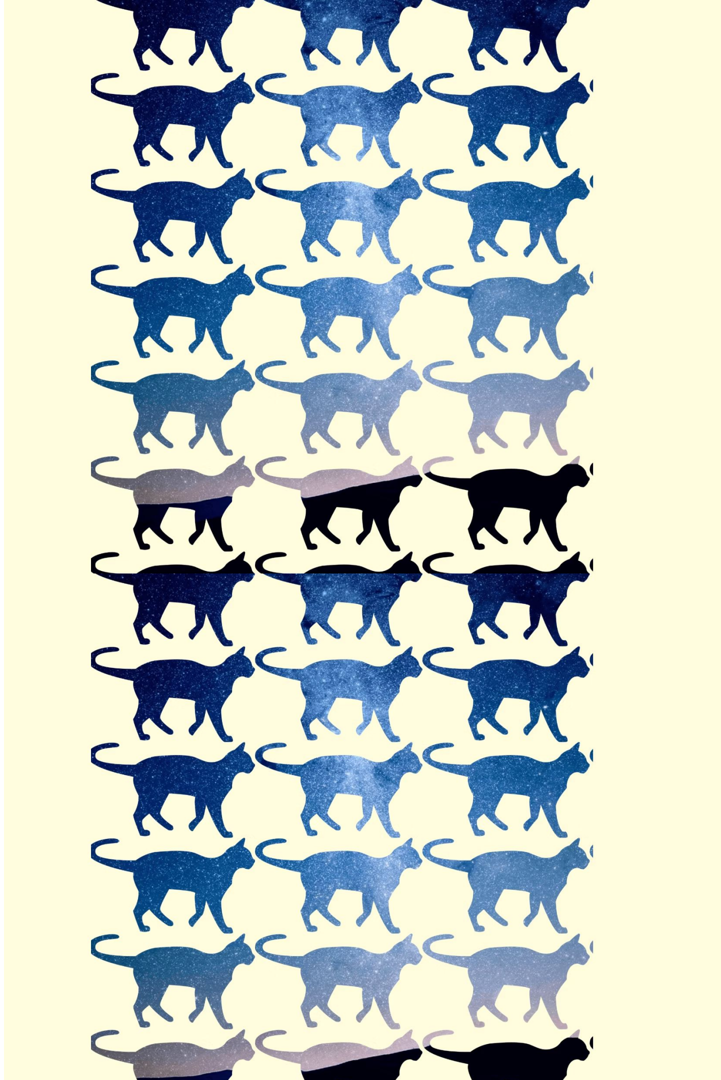
`mask-size`

`mask-repeat` – whether the mask repeats

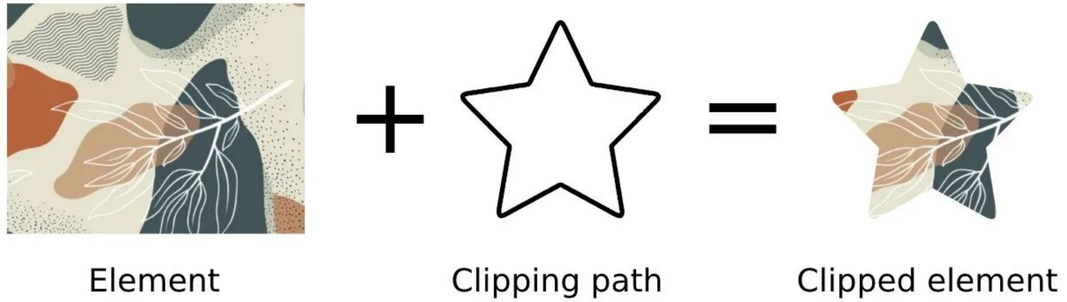
`mask-origin` – defines the starting point of the mask – `border`, `padding`, `content`

`mask-clip` – the area to which the mask is applied

`mask-composite` – allows combining mask layers



CSS clip-path

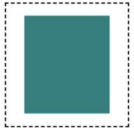


– Defines the area to show or hide

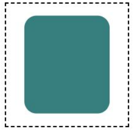
– Consists of shapes or coordinates

Generator clip-path – Clippy

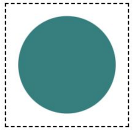
CSS clip-path shapes



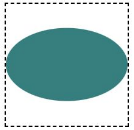
```
clip-path: inset(width height);
```



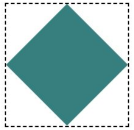
```
clip-path: inset(width height round border-radius);
```



```
clip-path: circle(radius at x, y);
```



```
clip-path: ellipse(radius-x, radius-y at x, y);
```



```
clip-path: polygon(vertex, vertex ...);
```

`inset` – rectangle

`circle`

`ellipse`

`polygon` – any shape with any number of corners

`path` – SVG path with coordinates

```
clip-path: path("M0.5,1 C0.5 ... “)
```

MASK VS CLIP-PATH

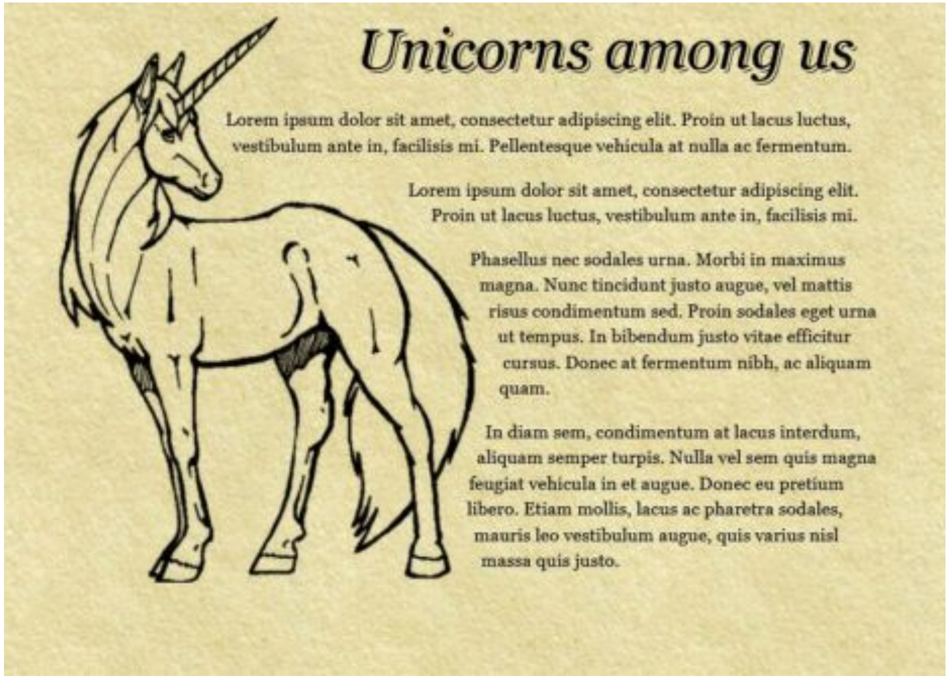
Raster	Vector
Partial Transparency	Opacity Only
Pre-drawn Images	Custom Shapes
More Complex Settings (mask-* properties)	Limits Element Shape (no additional properties)
Static Shape	Animatable Shape Changes
Text Wraps Shape Perimeter	Text Wraps Around Original Rectangle

shape-outside

Text Wrapping Around a Shape.

Shapes:

- **circle()** – Creates a circular shape for the text to wrap around.
- **ellipse()**
- **inset()** – Defines a rectangular area.
- **polygon()** – Creates any shape with three or more corners.
- **url()** – Uses an image as the shape for text wrapping.



Unicorns among us

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin ut lacus luctus, vestibulum ante in, facilisis mi. Pellentesque vehicula at nulla ac fermentum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin ut lacus luctus, vestibulum ante in, facilisis mi.

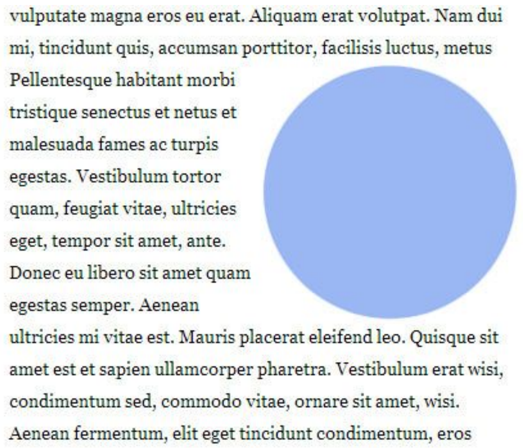
Phasellus nec sodales urna. Morbi in maximus magna. Nunc tincidunt justo augue, vel mattis risus condimentum sed. Proin sodales eget urna ut tempus. In bibendum justo vitae efficitur cursus. Donec at fermentum nibh, ac aliquam quam.

In diam sem, condimentum at lacus interdum, aliquam semper turpis. Nulla vel sem quis magna feugiat vehicula in et augue. Donec eu pretium libero. Etiam mollis, lacus ac pharetra sodales, mauris leo vestibulum augue, quis varius nisl massa quis justo.

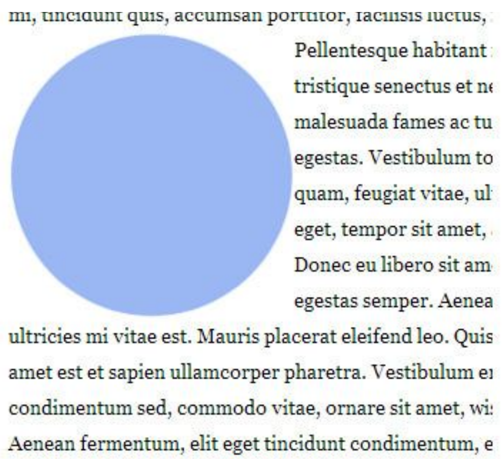
Applied to an element that the text should wrap around.

shape-outside

```
clip-path: circle(70% at 0% 50%)
 ↓
shape-outside: circle(70% at 0% 50%)
+
float: left/right
+
margin-left/right
```



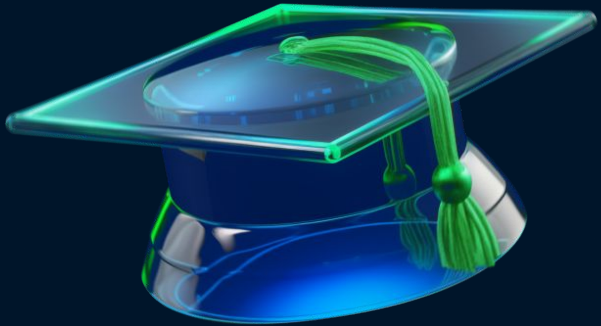
vulputate magna eros eu erat. Aliquam erat volutpat. Nam dui
mi, tincidunt quis, accumsan porttitor, facilisis luctus, metus
Pellentesque habitant morbi
tristique senectus et netus et
malesuada fames ac turpis
egestas. Vestibulum tortor
quam, feugiat vitae, ultricies
eget, tempor sit amet, ante.
Donec eu libero sit amet quam
egestas semper. Aenean
ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit
amet est et sapien ullamcorper pharetra. Vestibulum erat wisi,
condimentum sed, commodo vitae, ornare sit amet, wisi.
Aenean fermentum, elit eget tincidunt condimentum, eros



mi, tincidunt quis, accumsan porttitor, facilisis luctus,
Pellentesque habitant
tristique senectus et ne
malesuada fames ac tu
egestas. Vestibulum to
quam, feugiat vitae, ul
eget, tempor sit amet,
Donec eu libero sit am
egestas semper. Aenea
ultricies mi vitae est. Mauris placerat eleifend leo. Quis
amet est et sapien ullamcorper pharetra. Vestibulum e
condimentum sed, commodo vitae, ornare sit amet, wi
Aenean fermentum, elit eget tincidunt condimentum, e

Summary

1. Pseudo-elements
2. Pseudo-classes
3. Shadows
4. Filters
5. Masks



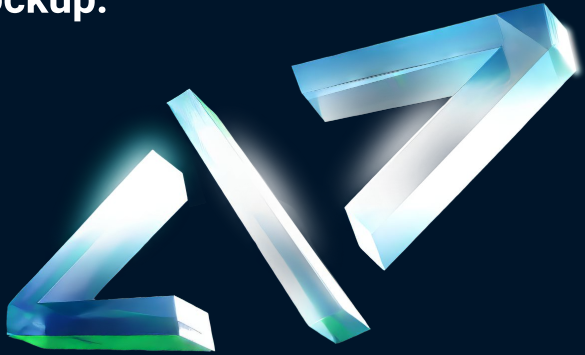
Homework

1. Complete one of the following courses to reinforce your understanding of the theory
2. Achieve the highest level of accuracy with the design mockup:
   - Apply all states for links and buttons:

     hover, active, focus, according to the UI kit
   - Apply visual effects such as

     shadows, shapes, filters and etc
   - Use pseudo-elements where necessary
   - Set all internal and external margins and padding

Your website should look exactly like the design mockup.
This is the final stage of work on the website. Next, we will only be adding animations.



BRO Academy

QUESTIONS?

Please fill out the feedback form

It’s very important for us

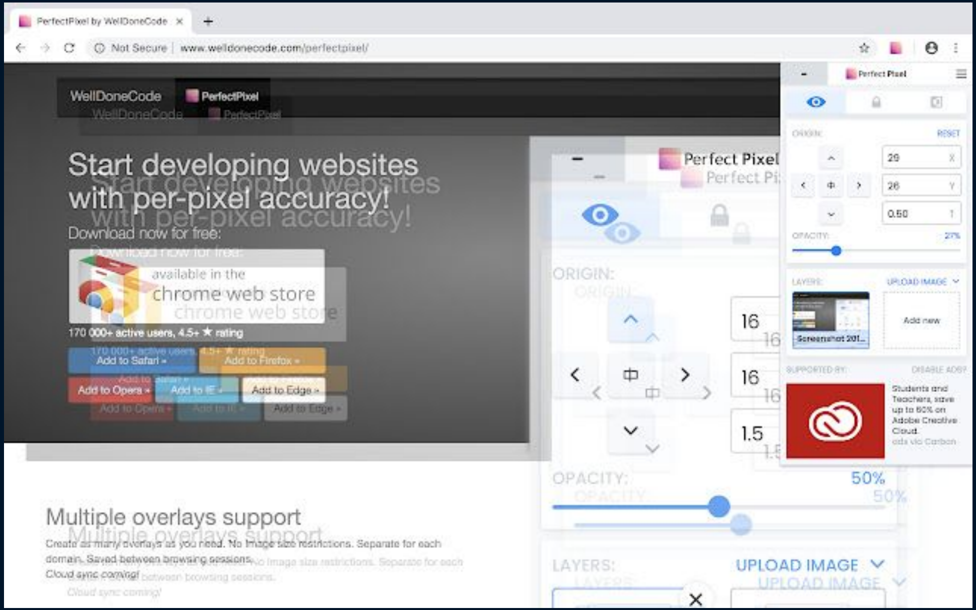
THANK YOU!

Have a good evening!

Perfect Pixel

– соответствие верстки макету пиксель в пиксель

- ➔ Стандарт индустрии
- ➔ Иногда допускаются отклонения до 5px
- ➔ Проверяется с помощью специального плагина в браузере

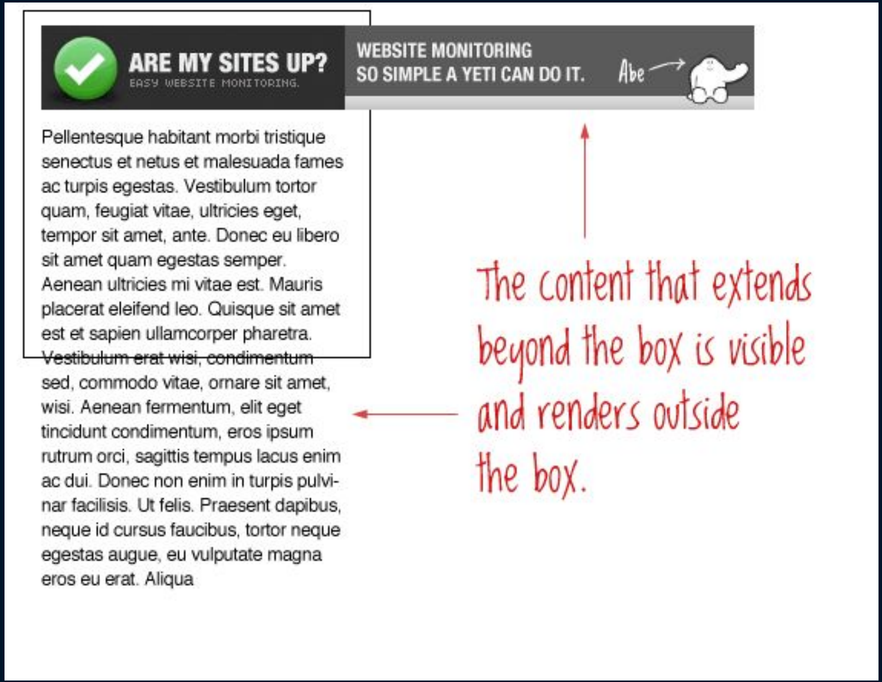


PerfectPixel by WellDoneCode
Not Secure | www.welldonecode.com/perfectpixel/
WellDoneCode
PerfectPixel
Start developing websites with per-pixel accuracy!
Download now for free:
available in the chrome web store
170 000+ active users, 4.5+ ★ rating
Add to Safari »
Add to Firefox »
Add to Opera »
Add to IE »
Add to Edge »
Perfect Pixel
ORIGIN:
RESET
16
16
1.5
OPACITY:
50%
LAYERS:
UPLOAD IMAGE
Add new
SUPPORTED BY:
DISABLE ADS?
Students and Teachers, save up to 60% on Adobe Creative Cloud.
ads via Carbon
Multiple overlays support
Create as many overlays as you need. No image size restrictions. Separate for each domain. Saved between browsing sessions.
Cloud sync coming!

Переполнение контентом

– сайт после верстки может меняться

- ➔ Больше/меньше текста
- ➔ изменение порядка элементов
- ➔ удаление/добавление новых элементов
- ➔ картинки могут поменяться
- ➔ ввод данных в форму



Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvi-nar facilisis. Ut felis. Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna eros eu erat. Aliqua

The content that extends beyond the box is visible and renders outside the box.