`::before` & `::after` allow to insert content with CSS

`::before` & `::after` allow to insert content with CSS

They have a specific required property `content`

If you don't set it, the extra element will not be displayed

`::before` & `::after` allow to insert content with CSS

They have a specific required property `content`

If you don't set it, the extra element will not be displayed

```
.new::before {
  content: "NEW!";
  color: red;
}
```

In this example we use fixed value `"NEW!"` which will always be the same.

HTML

```html
<ul>
  <li class="new">Item 1</li>
  <li class="new">Item 2</li>
  <li>Item 3</li>
</ul>
```

CSS

```css
.new::before {
  content: "NEW! ";
  color: red;
}
```

- NEW! Item 1
- NEW! Item 2
- Item 3

HTML

```html
<ul>
  <li class="new">Item 1</li>
  <li class="new">Item 2</li>
  <li>Item 3</li>
</ul>
```

CSS

```css
.new::before {
  content: "NEW! ";
  color: red;
}
```

- NEW! Item 1
- NEW! Item 2
- Item 3

HTML

```
<ul>
  <li class="new">Item 1</li>
  <li class="new">Item 2</li>
  <li>Item 3</li>
</ul>
```

CSS

```
.new::before {
  content: "NEW! ";
  color: red;
}
```

- NEW! Item 1
- NEW! Item 2
- Item 3

It is also possible to extract an attribute of the element itself using `attr()` to make the output variable.

An example useage of this, is to show reveal the url of a link in a printed object.

```
selector::after {
  content: attr(<name-of-the-attribute>);
}
```

```
a::after {
  content: attr(href);
}
```

```
HTML:
<a href="https://hr.nl">
    Hogeschool Rotterdam
</a>

CSS:
a::after {
  content: " (" attr(href) ")";
}
```

HTML:

```
<a href="https://hr.nl">
    Hogeschool Rotterdam
</a>
```

CSS:

```
a::after {
  content: " (" attr(href) ")";
}
```

Output:

```
Hogeschool Rotterdam (https://hr.nl)
```

HTML:

```html
<a href="https://hr.nl">
    Hogeschool Rotterdam
</a>
```

CSS:

```css
a::after {
  content: " (" attr(href) ")";
}
```

Output:

```
Hogeschool Rotterdam (https://hr.nl)
```

HTML:

```
<a href="https://hr.nl">
    Hogeschool Rotterdam
</a>
```

CSS:

```
a::after {
  content: " (" attr(href) ")";
}
```

Output:

```
Hogeschool Rotterdam (https://hr.nl)
```

```
attr(href)
attr(data-type)
```

```
<a href="https://hr.nl"
   data-type="External link">
    Hogeschool Rotterdam
</a>
```

With data- attributes it is possible to set custom attributes on elements.

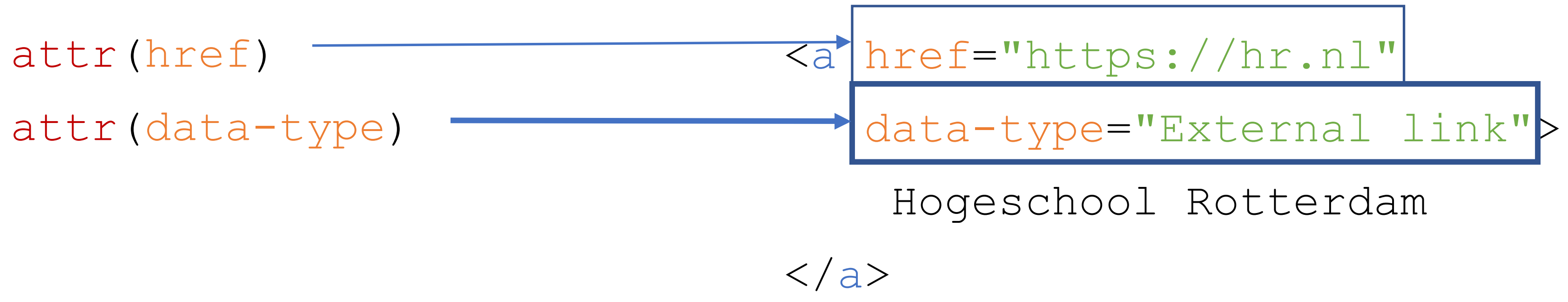
It is then possible to extract these attributes using CSS.

```
attr(href)
attr(data-type)
```

```
<a href="https://hr.nl"
   data-type="External link">
    Hogeschool Rotterdam
</a>
```

With data- attributes it is possible to set custom attributes on elements.

It is then possible to extract these attributes using CSS.



```
attr(href)
attr(data-type)
```

```
<a href="https://hr.nl"
   data-type="External link">
    Hogeschool Rotterdam
</a>
```

With data- attributes it is possible to set custom attributes on elements.

It is then possible to extract these attributes using CSS.

CSS:

```css
a::after {
  content: " (" attr(href) ")";
}

a::before {
  content: attr(data-type) ": ";
}
```

HTML:

```html
<a href="https://hr.nl"
   data-type="External link">
    Hogeschool Rotterdam
</a>
```

CSS:

```css
a::after {
  content: " (" attr(href) ")";
}

a::before {
  content: attr(data-type) ": ";
}
```

HTML:

```html
<a href="https://hr.nl"
   data-type="External link">
    Hogeschool Rotterdam
</a>
```

Result:

External link: Hogeschool Rotterdam (https://hr.nl)

CSS:

```
a::after {
  content: " (" attr(href) ")";
}

a::before {
  content: attr(data-type) ": ";
}
```

HTML:

```
<a href="https://hr.nl"
   data-type="External link">
    Hogeschool Rotterdam
</a>
```

Result:

External link: Hogeschool Rotterdam (https://hr.nl)

CSS:

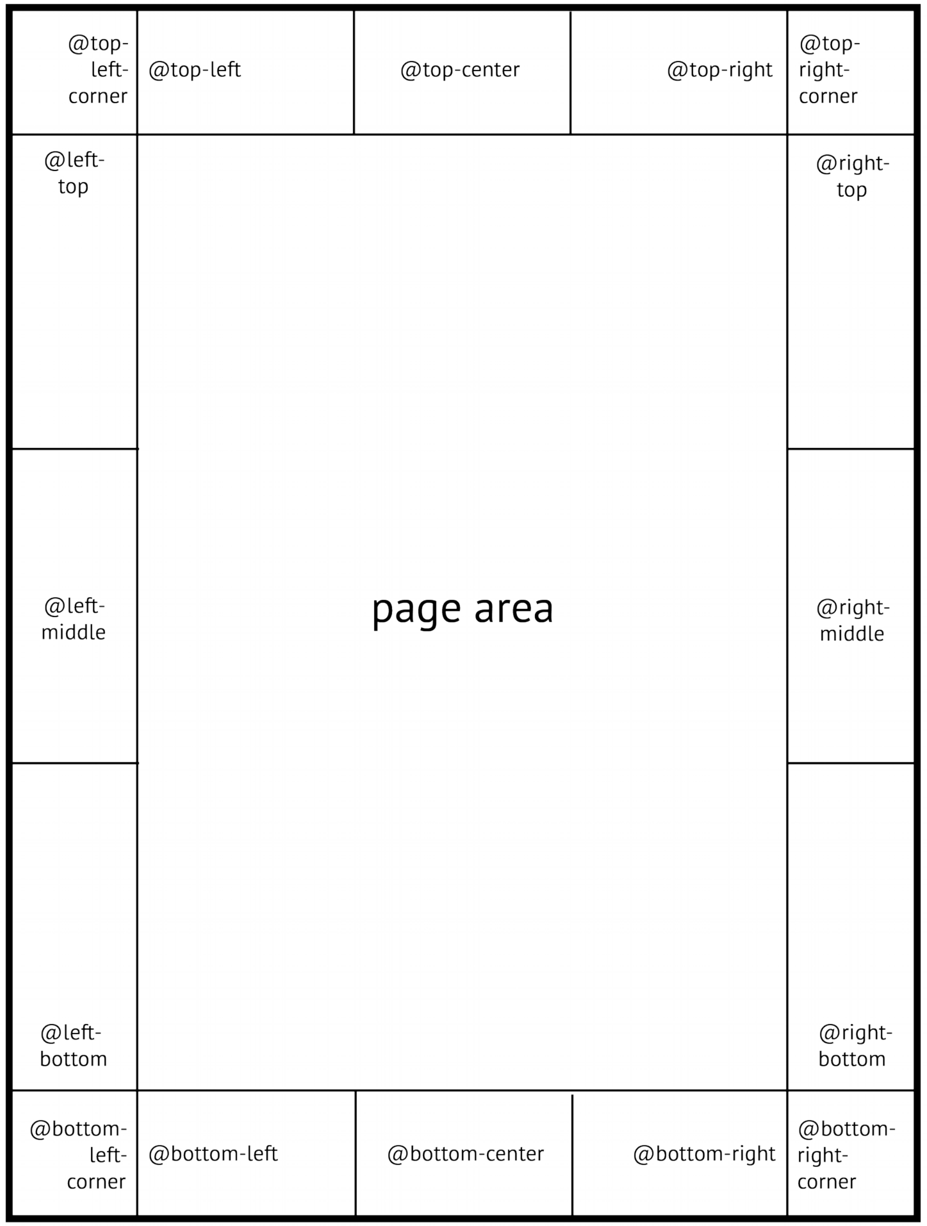
```css
a::after {
  content: " (" attr(href) ")";
}

a::before {
  content: attr(data-type) ": ";
}
```

HTML:

```html
<a href="https://hr.nl"
data-type="External link">
  Hogeschool Rotterdam
</a>
```

Result:

External link: Hogeschool Rotterdam (https://hr.nl)


@top-left-corner
@top-left
@top-center
@top-right
@top-right-corner
@left-top
@right-top
@left-middle
page area
@right-middle
@left-bottom
@right-bottom
@bottom-left-corner
@bottom-left
@bottom-center
@bottom-right
@bottom-right-corner

```css
@page {
  @top-center {
    content: "My book";
  }


  @bottom-left {
    content: counter(page);
  }
}
```

Variable content in the margin boxes.

Is depending on the content of the page.

For example, with a running header it would be the text which is in the latest h2.

```
HTML:
<h2>
    Hogeschool Rotterdam
</h2>
```

CSS:

```css
h2 {
  string-set: running-header content(text);
}

@page {
  @top-left {
    content: string(running-header);
  }
}
```

HTML:

```html
<h2>
    Hogeschool Rotterdam
</h2>
```

CSS:

```
h2 {
  string-set: running-header content(text);
}
```

HTML:

```
<h2>
  Hogeschool Rotterdam
</h2>
```

CSS:

```css
h2 {
  string-set: running-header content(text);
}
```

HTML:

```html
<h2>
   Hogeschool Rotterdam
</h2>
```

```css
@page {
  @top-left {
    content: string(running-header);
  }
}
```

CSS:

```
h2 {
  string-set: running-header content(text);
}
```

HTML:

```
<h2>
Hogeschool Rotterdam
</h2>
```

Whenever an `h2` is encountered the string `running-header` is set to the content of that element.

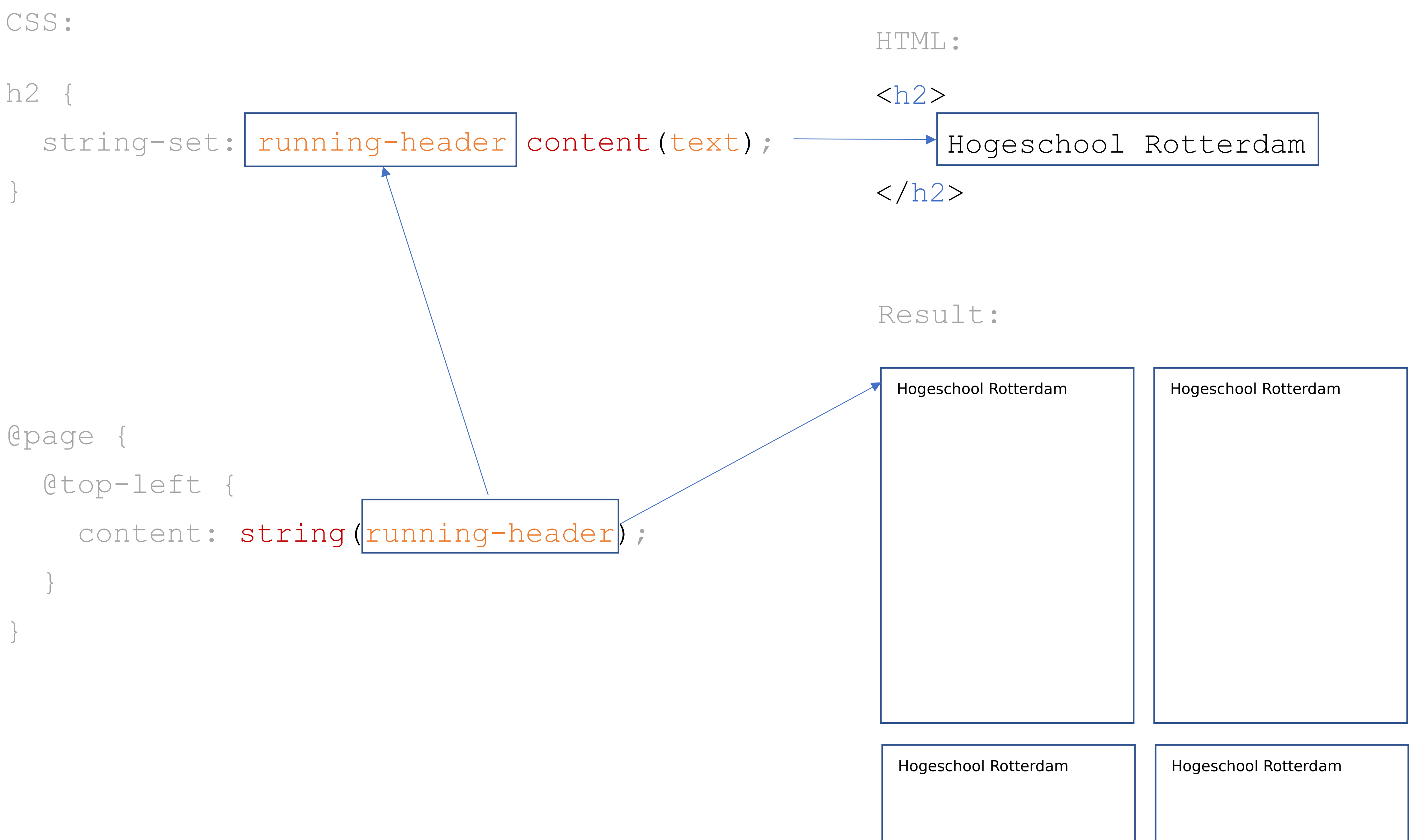
Like a variable. But confusingly css has a very different syntax for variables.

```
CSS:

h2 {
  string-set: running-header content(text);
}
```

```
HTML:

<h2>
  Hogeschool Rotterdam
</h2>
```

```
@page {
  @top-left {
    content: string(running-header);
  }
}
```

Takes the value of the string `running-header` and inserts it as text content in the top left margin box.



CSS:

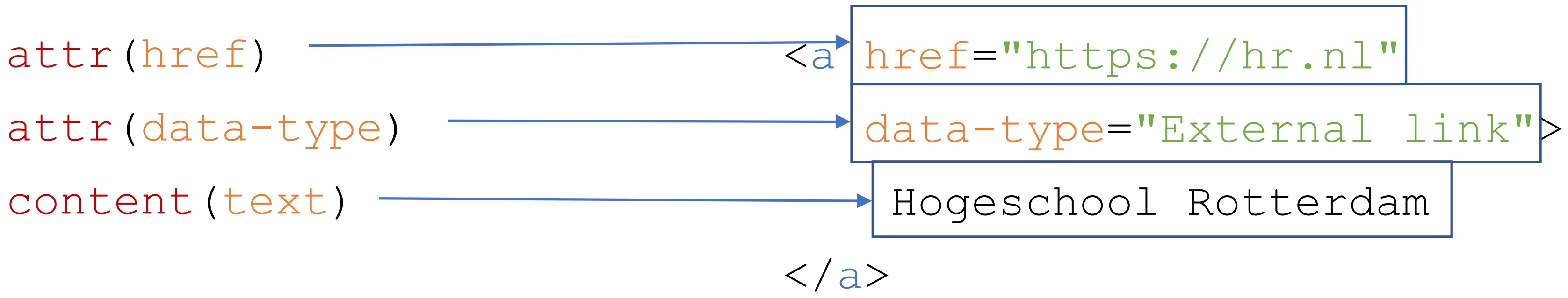
```
h2 {
  string-set: running-header content(text);
}
```

```
@page {
  @top-left {
    content: string(running-header);
  }
}
```

HTML:

```
<h2>
  Hogeschool Rotterdam
</h2>
```

Result:



```
attr(href)
attr(data-type)
content(text)
```

```
<a href="https://hr.nl"
   data-type="External link">
   Hogeschool Rotterdam
</a>
```

CSS:

```css
a.toc::after {
   content: target-counter(attr(href), page);
}
```

https://www.pagedjs.org/posts/2020-02-19-toc/

HTML :

```html
<a class="toc"
   href="#header-in-text">
  Hogeschool Rotterdam
</a>

...many lines in the document...

<h2 id="header-in-text">
    Hogeschool Rotterdam
</h2>
```

```
CSS:
a.toc::after {
   content: target-counter(attr(href), page);
}

HTML :
<a class="toc"
   href="#header-in-text">
  Hogeschool Rotterdam
</a>

...many lines in the document...

<h2 id="header-in-text">
    Hogeschool Rotterdam
</h2>
```

CSS:

```
a.toc::after {
   content: target-counter(attr(href), page);
}
```

HTML:

```
<a class="toc"
   href="#header-in-text">
  Hogeschool Rotterdam
</a>

...many lines in the document...

<h2 id="header-in-text">
    Hogeschool Rotterdam
</h2>
```

CSS:

```
a.toc::after {
   content: target-counter(attr(href), page);
}
```

HTML :

```
<a class="toc"
   href="#header-in-text">
  Hogeschool Rotterdam
</a>


...many lines in the document...


<h2 id="header-in-text">
    Hogeschool Rotterdam
</h2>
```

```
CSS:
a.toc::after {
   content: target-counter(attr(href), page);
}

HTML :
<a class="toc"
  href="#header-in-text">
  Hogeschool Rotterdam
</a>

...many lines in the document...

<h2 id="header-in-text">
    Hogeschool Rotterdam
</h2>
```

Extract the href attribute from the matched element

```
CSS:
a.toc::after {
   content: target-counter(attr(href), page);
}

HTML :
<a class="toc"
   href="#header-in-text">
  Hogeschool Rotterdam
</a>

...many lines in the document...

<h2 id="header-in-text">
    Hogeschool Rotterdam
</h2>
```

Find the element through the id.

```
CSS:
a.toc::after {
    content: target-counter(attr(href), page);
}

HTML :
<a class="toc"
   href="#header-in-text">
  Hogeschool Rotterdam
</a>

...many lines in the document...

<h2 id="header-in-text">
    Hogeschool Rotterdam
</h2>
```

What's the value of the page counter at that element?

Tutorial on pagedjs website

https://www.pagedjs.org/posts/2020-02-19-toc/

A recipe for an Index

Mark words you want to index with a special class 'book-index'.

Use a script to find all occurences of those elements and insert references to those elements in an index.

Insert the generated index in the document

```
#toc a::after {
  content: " - " target-counter(attr(href), page);
}
```

https://www.pagedjs.org/posts/2020-02-16-buildanindexwithpagedjs/

Recipe, for loading pagedjs in an existing document.

Rather than loading the javascript and css for pagedjs when tho document is loaded.

Insert the javascript & css in the document after a specific event.

For example, when a button with the id 'print' is clicked.

See:

pagedjs.snippet.html

pagedjs.html