

CSS

CSS FUNDAMENTALS

SELECTORS

Pseudo-class



IN A ROCKET

Learn front-end development at rocket speed

PSEUDO-CLASSES

A **pseudo-class** selects an element with a **special state specified by a keyword**.

Syntax `selector:pseudo-class {style properties}`

`a:link {color: green}`

With this code all a elements that have not yet been visited are shown in green.

DYNAMIC

**TARGET &
LANG**

**UI ELEMENT
STATES**

STRUCTURAL

NEGATION

DYNAMIC

PSEUDO-CLASSES

**TARGET &
LANG**

**UI ELEMENT
STATES**

STRUCTURAL

NEGATION

Dynamic pseudo-classes

:link

:visited

:hover

:active

:focus

Contact

a:link

Represents links that have not yet been visited.

Contact

a:visited

Styles for links that have been visited (exists in the browser's history).

Contact

a:hover

Generally triggered when the user hovers over an element with the cursor (mouse pointer).

Contact

a:active

Triggered when the user clicks the link or selects it with the keyboard's tab key.

Dynamic pseudo-classes

:link

:visited

:hover

:active

:focus

How to remember them?



LOVE



HATE

Selects all links that have **not yet been visited**.

Syntax `selector:link {style properties}`

`a:link {color: green}`

With this code all not visited links are shown in green.

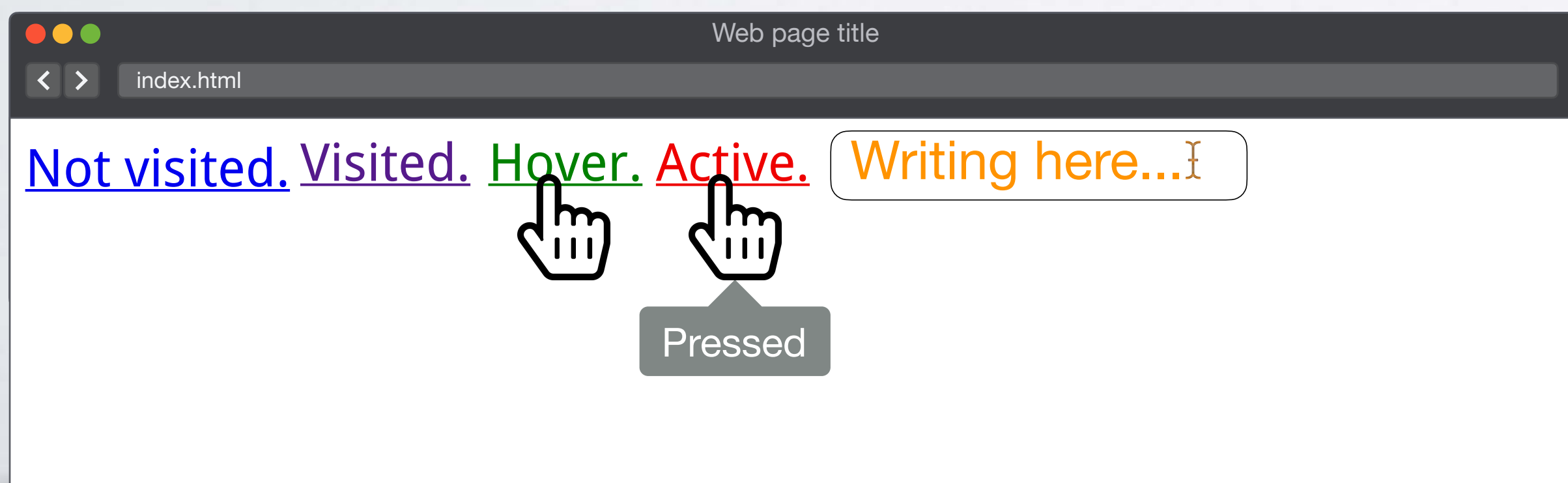
HTML

```
<body>
<a href="#">Not visited.</a>
<a href="#">Visited.</a>
<a href="#">Hover.</a>
<a href="#">Active.</a>
<input type="text" name="zip" id="zip">
</body>
```

CSS

```
a:link { color: blue; }
a:visited { color: purple; }
a:hover { color: green; }
a:active { color: red; }
input:focus { color: orange; }
```

Browser



DYNAMIC

**TARGET &
LANG**

PSEUDO-CLASSES

**UI ELEMENT
STATES**

STRUCTURAL

NEGATION

Selects a **fragment identifier** that has a location within a resource.

Syntax `selector:target {style properties}`

`h2:target {color: green}`

With this code the h2 used as a target is shown in green.

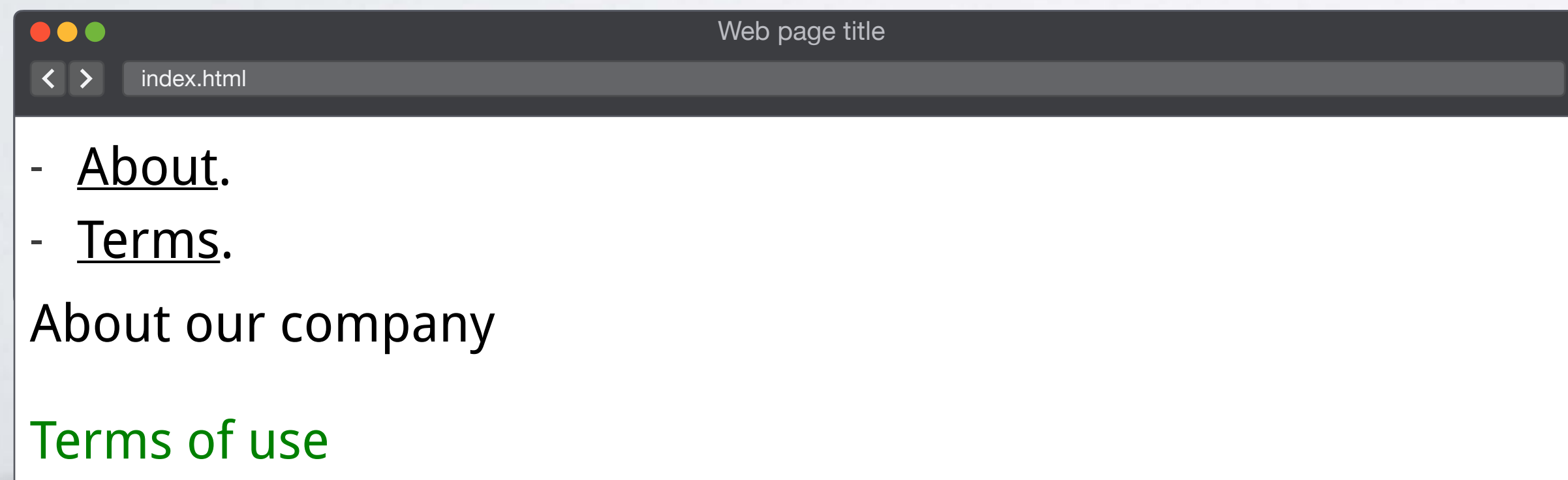
HTML

```
<body>
<a href="#about">About.</a>
<a href="#terms">Terms.</a>
<h2 id="about">About our company</h2>
<h2 id="terms">Terms of use</h2>
</body>
```

CSS

```
h2:target { color: green; }
```

Browser



PSEUDO-CLASSES / TARGET & LANG

Selects elements based on the **language** they are determined to be in.

Syntax `selector:lang(lg) {style properties}`

`p:lang(en) {color: green}`

With this code the paragraphs in English (en) are shown in green.

HTML

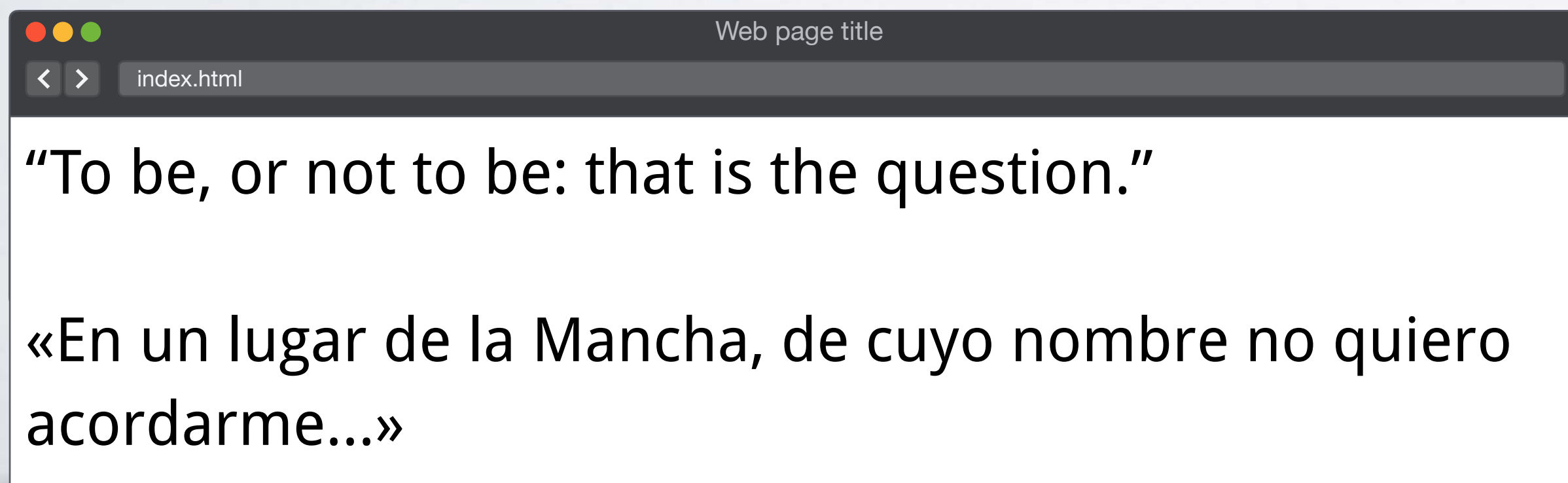
```
<body>
<p lang="en"><q>To be, or not to be:
that is the question.</q></p>
<p lang="es"><q>En un lugar de la
Mancha, de cuyo nombre no quiero
acordarme...</q></p>
</body>
```

CSS

```
p:lang(en) > q { quotes: '\201C' '\201D'; }

p:lang(es) > q { quotes: '«' '»'; }
```

Browser



DYNAMIC

**TARGET &
LANG**

**UI ELEMENT
STATES**

PSEUDO-CLASSES

STRUCTURAL

NEGATION

PSEUDO-CLASSES / UI ELEMENT STATES

Selects any enabled element (it can be selected, clicked on, typed into, etc.).

Syntax `selector:enabled {style properties}`

`input:enabled {color: green}`

With this code only the enabled inputs are shown in green.

PSEUDO-CLASSES / UI ELEMENT STATES

Selects any disabled element.

Syntax `selector:enabled {style properties}`

`input:disabled {color: green}`

With this code only the disabled inputs are shown in green.

HTML

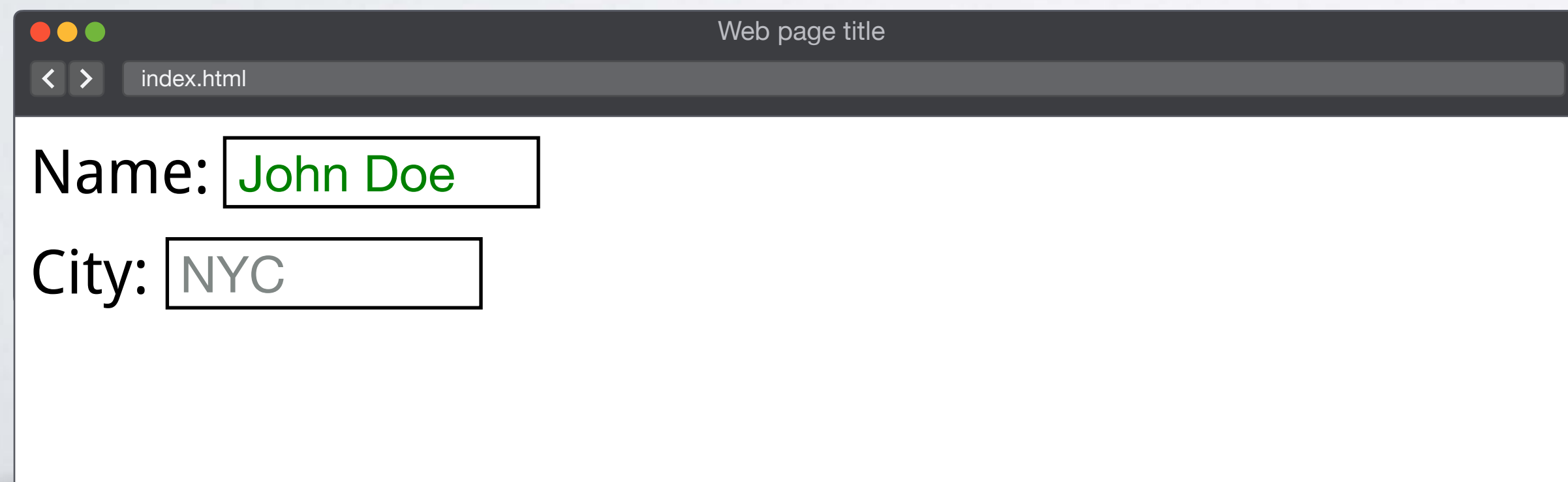
```
<body>
<form>
<label for="name">Name:</label>
<input type="text" id="name">
<label for="city">City:</label>
<input type="text" id="city" value="NYC"
disabled="disabled">
<input type="button" value="Submit">
</form>
</body>
```

CSS

```
input:enabled { color: green; }
```

```
input:disabled { color: gray; }
```

Browser



PSEUDO-CLASSES / UI ELEMENT STATES

Selects any radio, checkbox or option that is checked or toggled to an on state.

Syntax `selector: checked {style properties}`

`input: checked {color: green}`

With this code only the checked inputs are shown in green.

HTML

```
<body>
<form>
<input type="radio" name="pay" id="cash">
<label for="cash">Cash</label>
<input type="radio" name="pay" id="card">
<label for="card">Card</label>

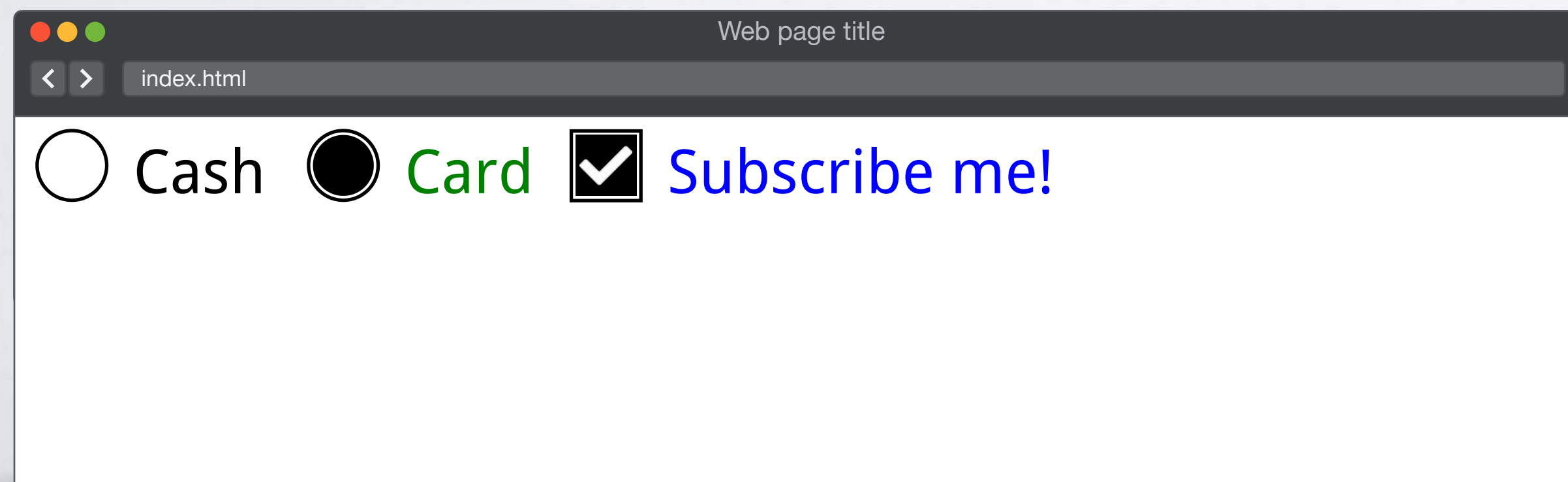
<input type="checkbox" name="nwslt" id="nwslt">
<label for="nwslt">Subscribe me!</label>
</form>
</body>
```

CSS

```
input[type=radio]:checked + label { color:
green; }
```

```
input[type=checkbox]:checked + label { color:
blue; }
```

Browser



DYNAMIC

**TARGET &
LANG**

**UI ELEMENT
STATES**

STRUCTURAL

PSEUDO-CLASSES

NEGATION

Structural pseudo-classes

:root

:empty

:first-child

:last-child

:nth-child()

:only-child

:first-of-type

:last-of-type

:nth-of-type()

:only-of-type

PSEUDO-CLASSES / STRUCTURAL

Selects an element that is the root of the document (in HTML, this is always the HTML element).

Syntax `:root {style properties}`

`:root {color: green}`

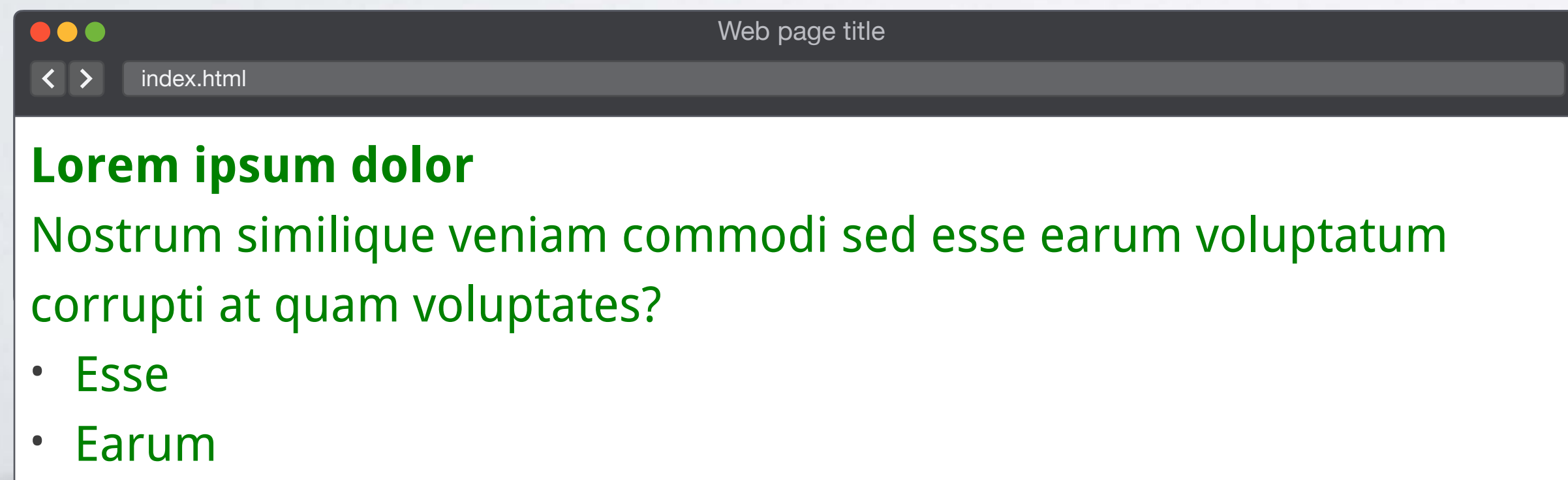
HTML

```
<body>
<h2>Lorem ipsum dolor</h2>
<p>Nostrum similique veniam commodi sed esse earum
voluptatum corrupti at quam voluptates?</p>
<ul>
  <li>Esse</li>
  <li>Earum</li>
</ul>
</body>
```

CSS

```
:root { color: green; }
```

Browser



HTML

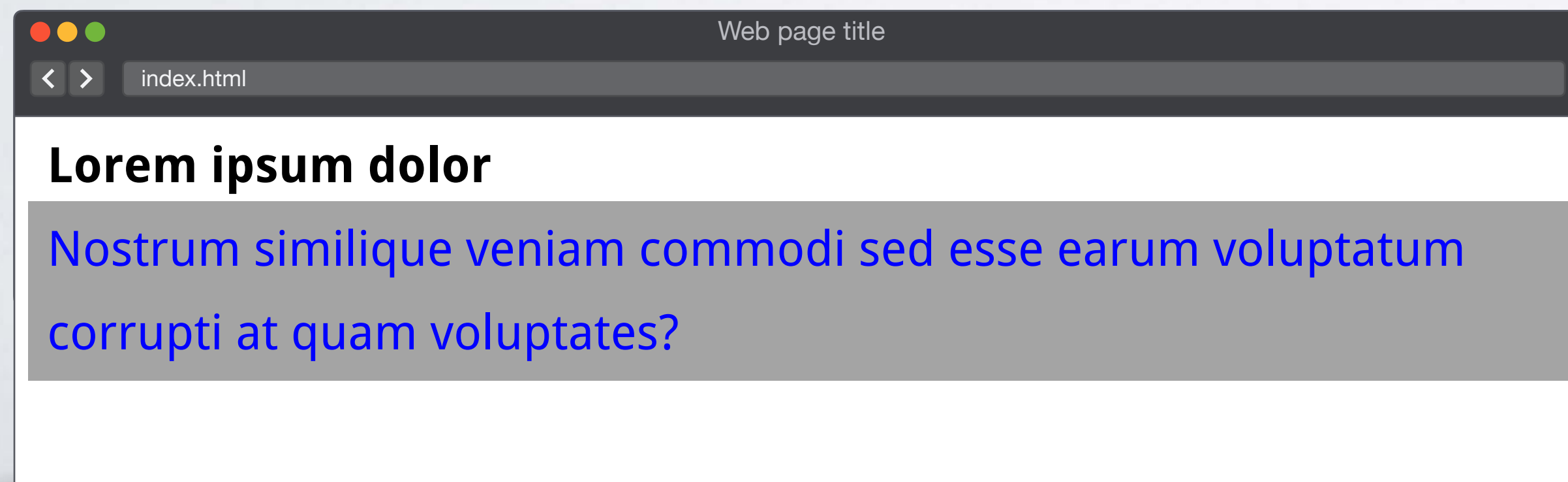
```
<body>
<h2>Lorem ipsum dolor</h2>
<p>Nostrum similique veniam commodi sed esse
earum voluptatum corrupti at quam voluptates?
</p>
</body>
```

CSS

```
:root {
  --color-primary: blue;
  --color-secondary: gray;
}

p {
  background: var(--color-secondary);
  color: var(--color-primary);
}
```

Browser



Structural pseudo-classes

:root

:empty

:first-child

:last-child

:nth-child()

:only-child

:first-of-type

:last-of-type

:nth-of-type()

:only-of-type

Selects an element that has no children at all.

Syntax `element:empty {style properties}`

`div:empty {background: gray}`

With this code only the empty divs are shown in gray.

HTML

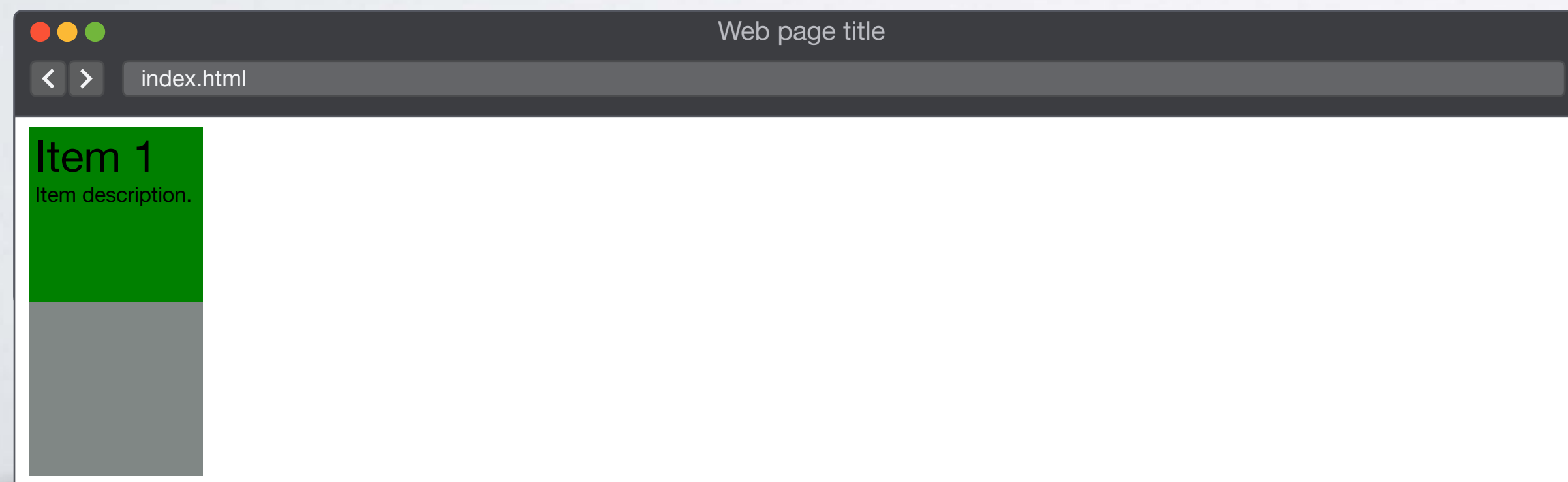
```
<body>
<article>
  <h2>Item 1</h2>
  <p>Item description.</p>
</article>
<article><!-- No item here --></article>
</body>
```

CSS

```
article {
  background: green;
  width: 5rem;
  height: 5rem;
}

article:empty {
  background: gray;
}
```

Browser



Structural pseudo-classes

:root
:empty

:first-child

:last-child
:nth-child()
:only-child

:first-of-type
:last-of-type
:nth-of-type()
:only-of-type

Selects an element that is first in a list of siblings.

Syntax `element:first-child {style properties}`

`li:first-child {color: green}`

With this code only the first `li` of a list is shown in green.

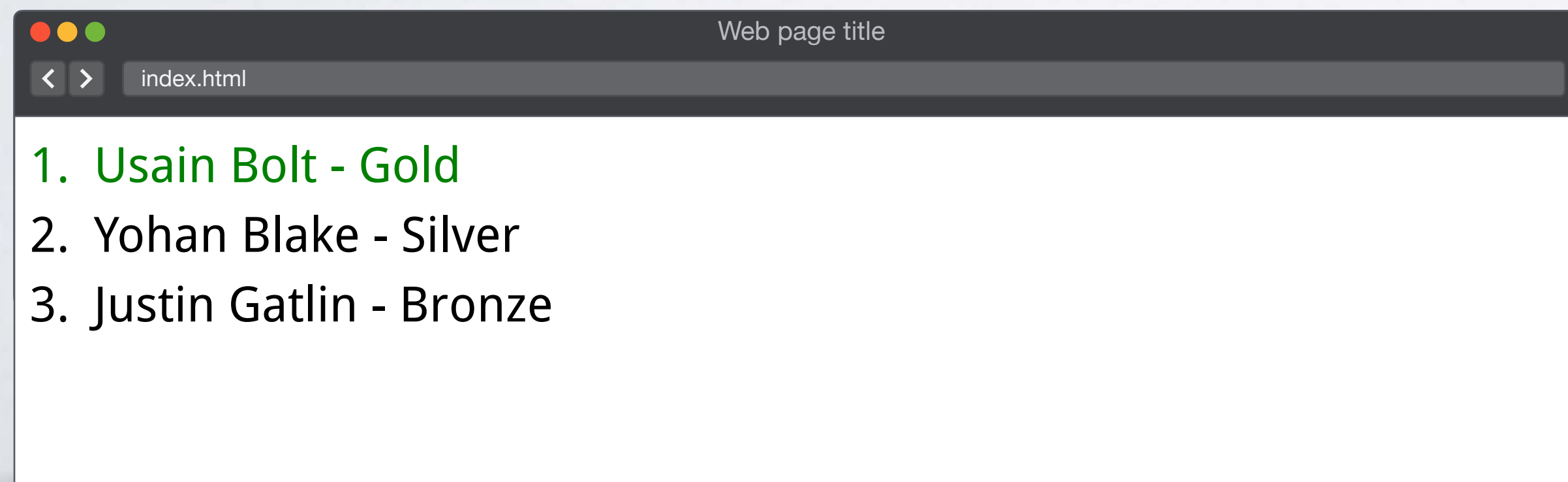
HTML

```
<body>
<ol>
  <li>Usain Bolt - Gold</li>
  <li>Yohan Blake - Silver</li>
  <li>Justin Gatlin - Bronze</li>
</ol>
</body>
```

CSS

```
li:first-child { color: green; }
```

Browser



Structural pseudo-classes

:root
:empty

:first-child
:**last-child**

:nth-child()
:only-child

:first-of-type
:last-of-type
:nth-of-type()
:only-of-type

Selects an element that is last in a list of siblings.

Syntax `element:last-child {style properties}`

`li:last-child {color: green}`

With this code only the last `li` of a list is shown in green.

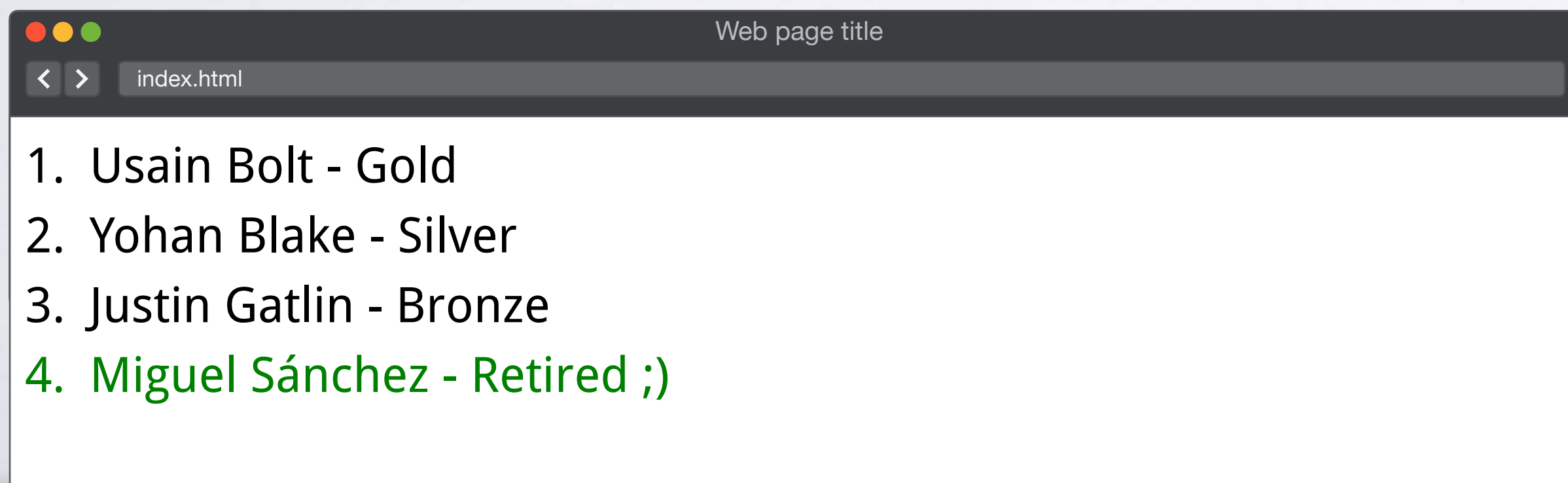
HTML

```
<body>
<ol>
  <li>Usain Bolt - Gold</li>
  <li>Yohan Blake - Silver</li>
  <li>Justin Gatlin - Bronze</li>
  <li>Miguel Sánchez - Retired ;) </li>
</ol>
</body>
```

CSS

```
li:last-child { color: green; }
```

Browser



Structural pseudo-classes

:root
:empty

:first-child
:last-child
:nth-child()
:only-child

:first-of-type
:last-of-type
:nth-of-type()
:only-of-type

Selects an element that has $n+b-1$ siblings before it in the document tree and has a parent element.

Syntax `element:nth-child(an + b) {style properties}`

`li:nth-child(2n+1) {color: green}`

With this code every odd `li` of a list is shown in green.

li:nth-child(4) { color: green; }

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8

Difference

`li:nth-child(2n+1) { color: green; }`

First term

-
- Item 1
 - Item 2
 - Item 3
 - Item 4
 - Item 5
 - Item 6
 - Item 7
 - Item 8

Difference

`li:nth-child(2n+2) { color: green; }`

First term

-
- Item 1
 - Item 2
 - Item 3
 - Item 4
 - Item 5
 - Item 6
 - Item 7
 - Item 8

li:nth-child(odd) { color: green; }

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8

li:nth-child(even) { color: green; }

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8

Difference

`li:nth-child(1n+3) { color: green; }`

First term

-
- Item 1
 - Item 2
 - Item 3
 - Item 4
 - Item 5
 - Item 6
 - Item 7
 - Item 8

`li:nth-child(-1n+3) { color: green; }`

Difference

First term

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8

`li:nth-last-child(-n+3) { color: green; }`

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8

Structural pseudo-classes

:root
:empty

:first-child
:last-child
:nth-child()
:only-child

:first-of-type
:last-of-type
:nth-of-type()
:only-of-type

Selects an element that has no siblings.

Syntax `element:only-child {style properties}`

`li:only-child {color: green}`

With this code only an `li` without siblings is shown in green.

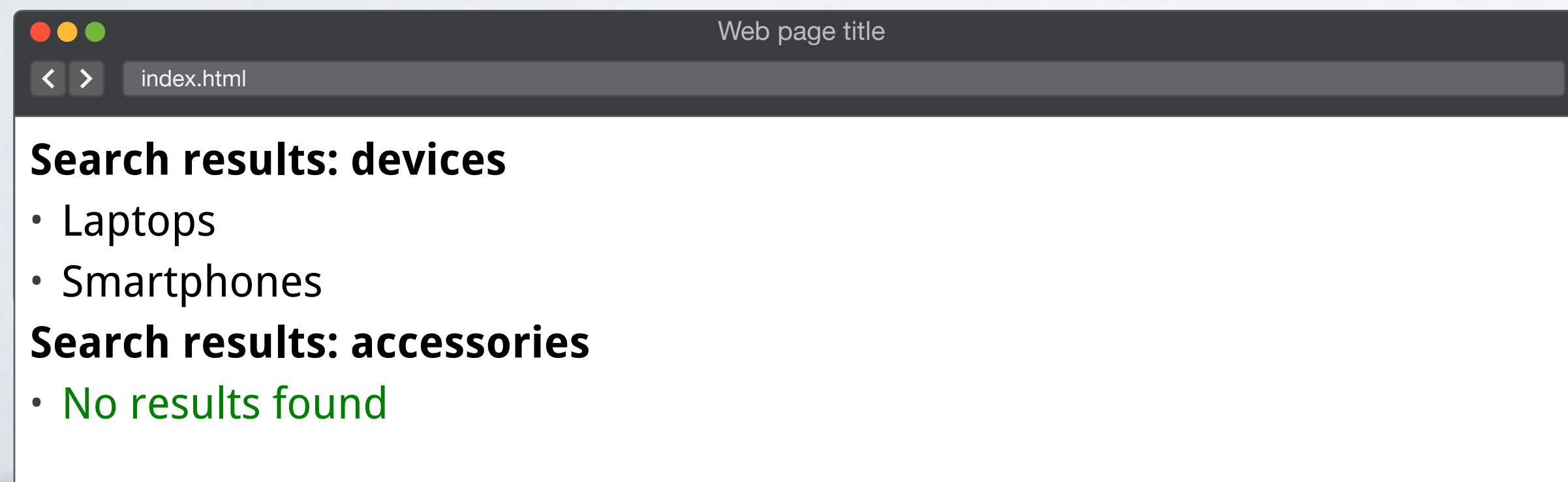
HTML

```
<body>
<h2>Search results: devices</h2>
<ul>
  <li>Laptops</li>
  <li>Smartphones</li>
</ul>
<h2>Search results: accessories</h2>
<ul>
  <li>No results found</li>
</ul>
</body>
```

CSS

```
li:only-child { color: green; }
```

Browser



Structural pseudo-classes

:root
:empty

:first-child
:last-child
:nth-child()
:only-child

:first-of-type
:last-of-type
:nth-of-type()
:only-of-type

Selects an element that is the first sibling of its type.

Syntax `element:first-of-type {style properties}`

`p:first-of-type {color: green}`

With this code only the first p in a group of children paragraphs is shown in green.

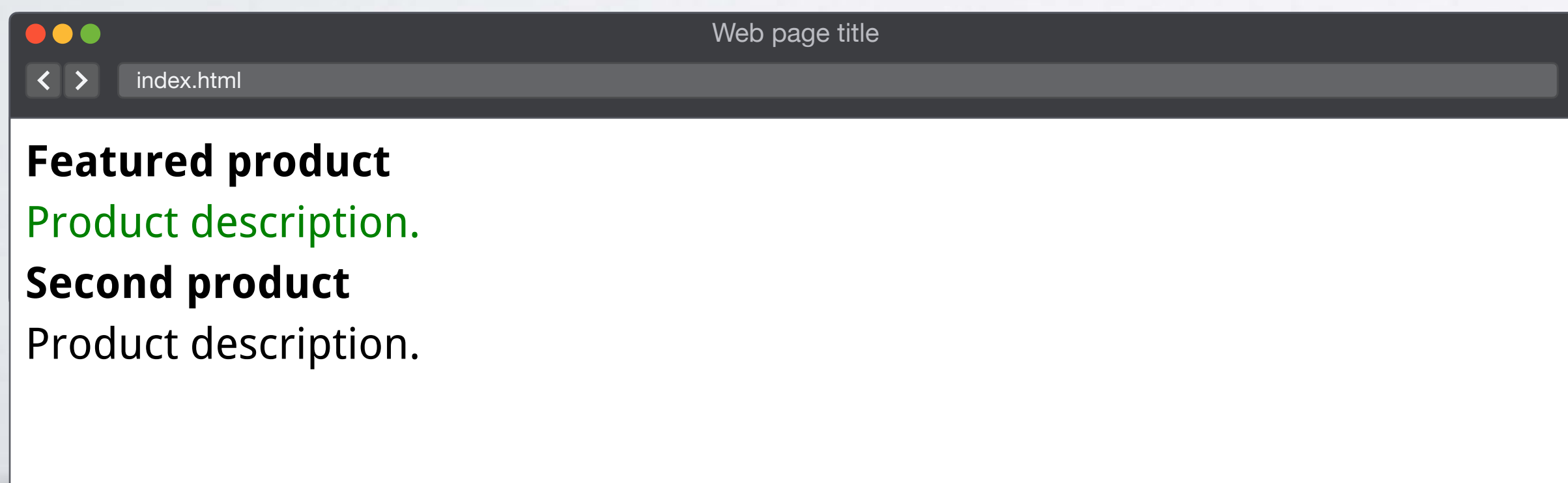
HTML

```
<body>
<main>
  <h2>Featured product</h2>
  <p>Product description.</p>
  <h2>Second product</h2>
  <p>Product description.</p>
</main>
</body>
```

CSS

```
p:first-of-type { color: green; }
```

Browser



Structural pseudo-classes

:root
:empty

:first-child
:last-child
:nth-child()
:only-child

:first-of-type
:**last-of-type**
:nth-of-type()
:only-of-type

Selects an element that is the last sibling of its type.

Syntax `element:first-of-type {style properties}`

`p:last-of-type {color: green}`

With this code only the last p in a group of children paragraphs is shown in green.

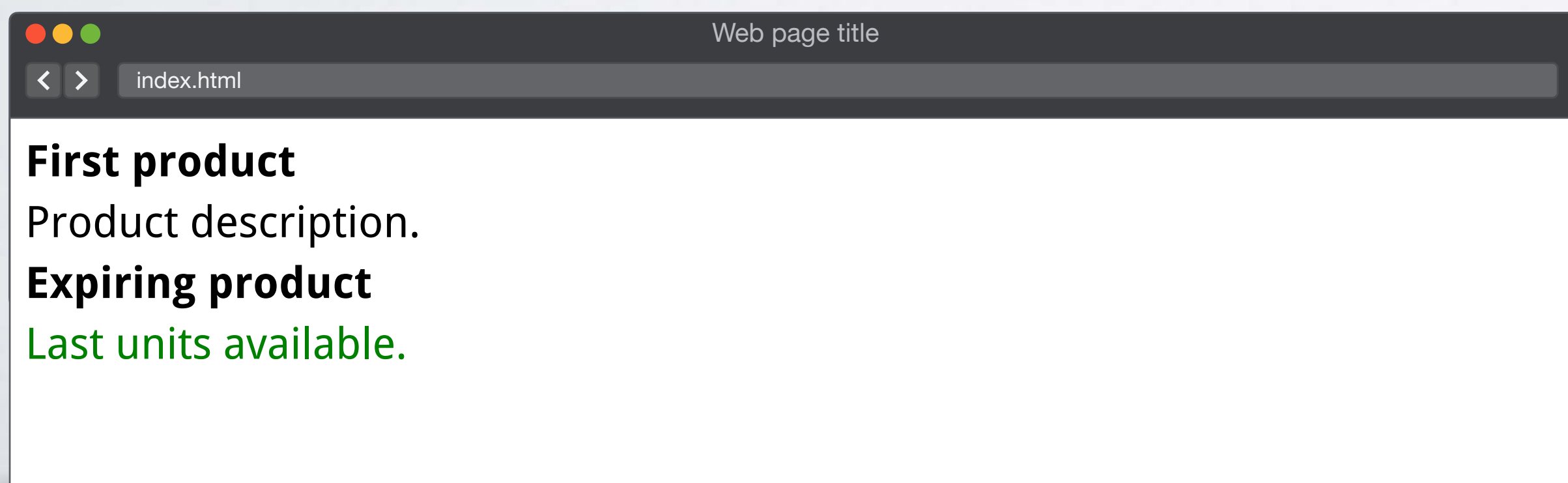
HTML

```
<body>
<main>
  <h2>First product</h2>
  <p>Product description.</p>
  <h2>Expiring product</h2>
  <p>Last units available.</p>
</main>
</body>
```

CSS

```
p:last-of-type { color: green; }
```

Browser



Structural pseudo-classes

:root
:empty

:first-child
:last-child
:nth-child()
:only-child

:first-of-type
:last-of-type
:nth-of-type()
:only-of-type

Selects elements of a given type, based on their position among a group of siblings.

Syntax `element:nth-of-type()` {style properties}

`p:nth-of-type(odd) {color: green}`

With this code only the odd paragraphs in a group of children paragraphs are shown in green.

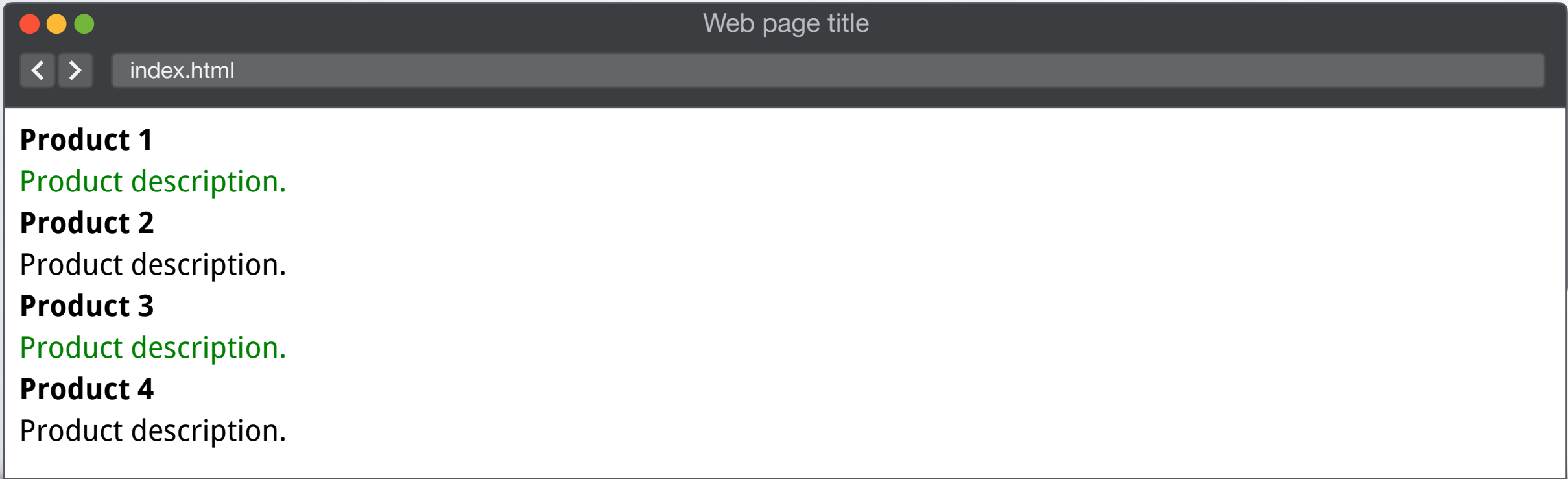
HTML

```
<body>
<main>
  <h2>Product 1</h2>
  <p>Product description.</p>
  <h2>Product 2</h2>
  <p>Product description.</p>
  <h2>Product 3</h2>
  <p>Product description.</p>
  <h2>Product 4</h2>
  <p>Product description.</p>
</main>
</body>
```

CSS

```
p:nth-of-type(odd) { color: green; }
```

Browser



Structural pseudo-classes

:root
:empty

:first-child
:last-child
:nth-child()
:only-child

:first-of-type
:last-of-type
:nth-of-type()
:only-of-type

PSEUDO-CLASSES / STRUCTURAL

Selects an element that has no siblings with the same expanded element name.

Syntax `element:only-of-type() {style properties}`

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

With this code only the paragraph with no other sibling paragraphs is shown in green.

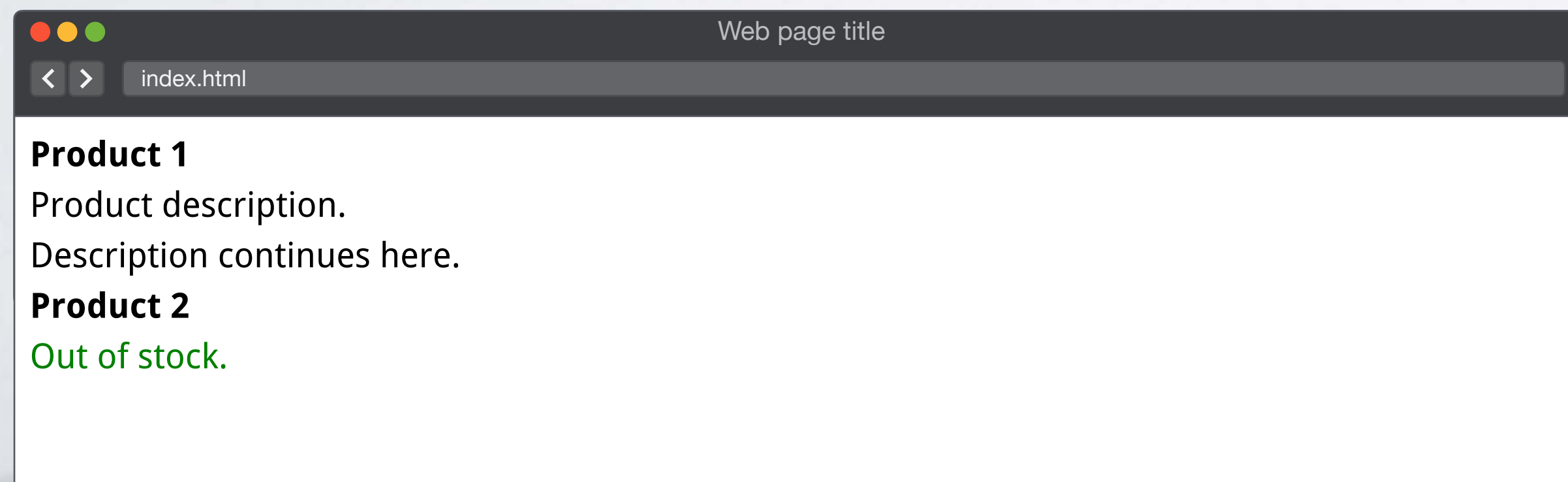
HTML

```
<body>
<article>
  <h2>Product 1</h2>
  <p>Product description.</p>
  <p>Description continues here.</p>
</article>
<article>
  <h2>Product 2</h2>
  <p>Out of stock.</p>
</article>
</body>
```

CSS

```
p:only-of-type { color: green; }
```

Browser



DYNAMIC

**TARGET &
LANG**

**UI ELEMENT
STATES**

STRUCTURAL

NEGATION

PSEUDO-CLASSES

PSEUDO-CLASSES / NEGATION

Selects elements that do not match a list of selectors.

Syntax `element:not(X) {style properties}`

`header :not(h1) {color: green}`

With this code all the elements of a header are shown in green, excluding all h1 headers.

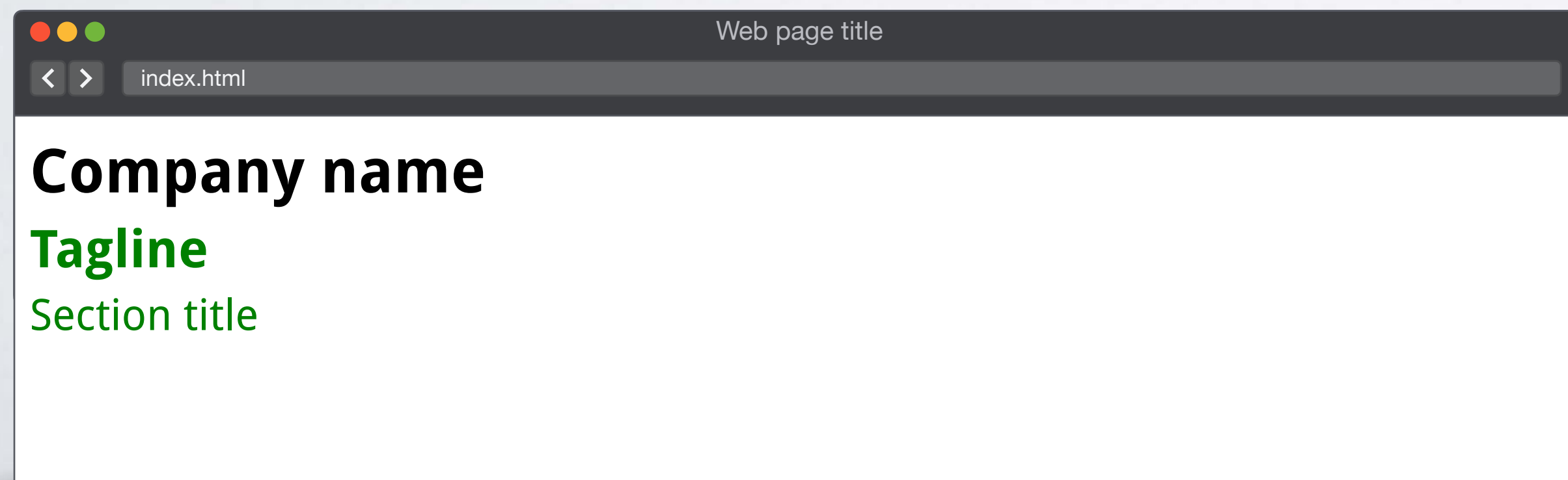
HTML

```
<body>
<header>
  <h1>Company name</h1>
  <h2>Tagline</h2>
  <h3>Section title</h3>
</header>
</body>
```

CSS

```
header:not(h1) { color: green; }
```

Browser



HTML

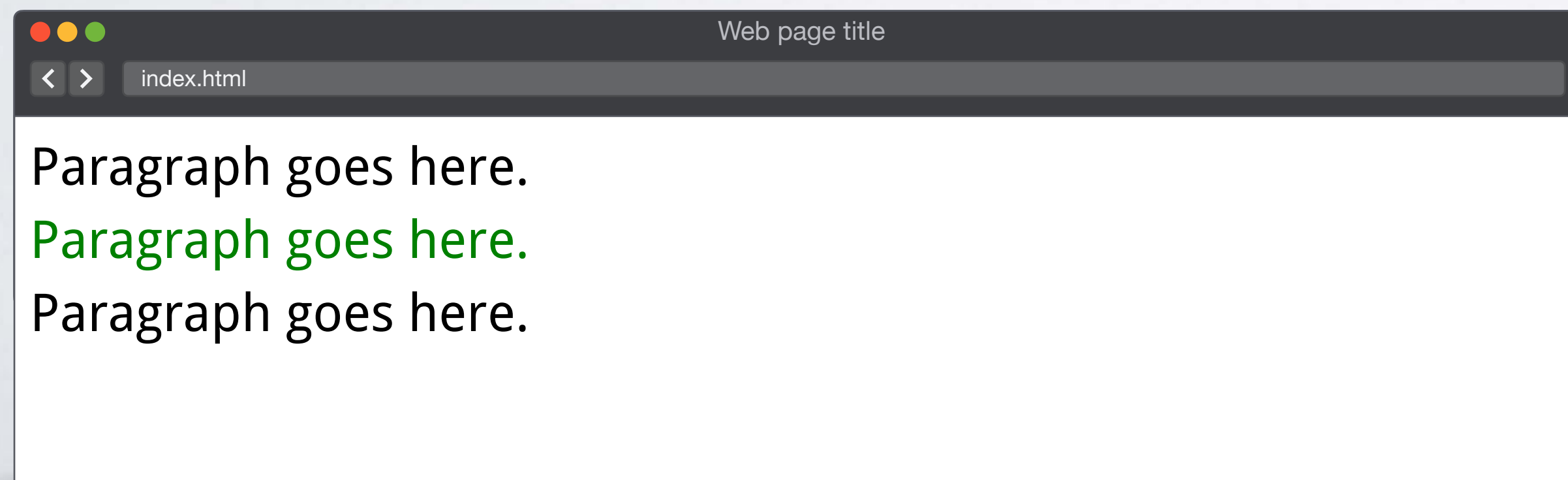
```
<body>
<p class="mini">Paragraph goes here.</p>
<p>Paragraph goes here.</p>
<p class="mini">Paragraph goes here.</p>
</body>
```

CSS

```
.mini { color: black; }

p:not(.mini) { color: green; }
```

Browser



DYNAMIC

PSEUDO-CLASSES



**TARGET &
LANG**

PSEUDO-CLASSES



**UI ELEMENT
STATES**

PSEUDO-CLASSES



STRUCTURAL

PSEUDO-CLASSES



NEGATION

PSEUDO-CLASSES



W3C Recommendation

TABLE OF CONTENTS

1.

Introduction

1.1.

Dependencies

1.2.

Terminology

1.3.

Changes from CSS2

2.

Selectors

3.

Case sensitivity

4.

Selector syntax

5.

Groups of selectors

6.

Simple selectors

6.1.

Type selector

6.1.1.

Type selectors and namespaces

6.2.

Universal selector

6.2.1.

Universal selector and namespaces

6.3.

Attribute selectors

6.3.1.

Attribute presence and value selectors

6.3.2.

Substring matching attribute selectors

6.3.3.

Attribute selectors and namespaces

6.3.4.

Default attribute values in DTDs

6.4.

Class selectors

6.5.

ID selectors

6.6.

Pseudo-classes

6.6.1.

Dynamic pseudo-classes

6.6.1.1.

The link pseudo-classes: :link and :visited

6.6.1.2.

The user action pseudo-classes :hover, :active, and :focus

6.6.2.

The target pseudo-class :target

6.6.3.

The language pseudo-class :lang

Selectors Level 3

W3C Recommendation 06 November 2018



This version:

<https://www.w3.org/TR/2018/REC-selectors-3-20181106/>

Latest version:

<https://www.w3.org/TR/selectors-3/>

Previous version:

<https://www.w3.org/TR/2018/PR-selectors-3-20180911/>

Latest version of Selectors:

<https://www.w3.org/TR/selectors/>

Editor's Draft

<https://drafts.csswg.org/selectors-3/>

Feedback:

File an [issue on GitHub](#)

Editors:

[Tantek Çelik](#) (Invited Expert)

[Elika J. Etemad](#) (Invited Expert)

Daniel Glazman (Disruptive Innovations SARL)

[Ian Hickson](#) (Google)

Peter Linss (former editor, [Netscape/AOL](#))

John Williams (former editor, [Quark, Inc.](#))

Please check the [errata](#) for any errors or issues reported since publication.

Copyright © 2018 W3C® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

Selectors are patterns that match against elements in a tree, and as such form one of several technologies that

SOURCE: [Selectors Level 3 by W3C](#).

inrocket.com

Learn front-end development at rocket speed



We respect your time

Just direct, clear
videos and slides.



Step-by-step guides

Actionable steps for creating
real-world solutions.



Ready-to-use code

Code you can just copy &
paste into your projects.



No more subscriptions

Subscription-free
premium courses.

TRADEMARK DISCLOSURE

This course is an independent initiative and is not affiliated, endorsed, sponsored, or officially connected with any of the trademarks mentioned in these educational materials. Trademarks, service marks, and logos used in this course belong to their respective owners.

The use of these marks in our course is solely for educational purposes and descriptive identification. It does not imply any association with the trademark holder or endorsement by them.

All information provided is intended as educational material and should not be taken as legal advice. If you have questions regarding the use of copyrighted material or trademarks, please consult with an intellectual property lawyer.



This work is licensed under
CC Attribution-NonCommercial-NoDerivatives 4.0 International.

