

# Intro to CSS

# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

... <div>

... <h2>

... <p>

... <p>

... <span>

... <p>

# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

# Thinking inside the box

## The Cottage Garden

The cottage garden is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in *England* and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained *English estate gardens*.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.



# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

Margins, Padding,  
Width

hello

# Margins, Padding, Width

hello

Border

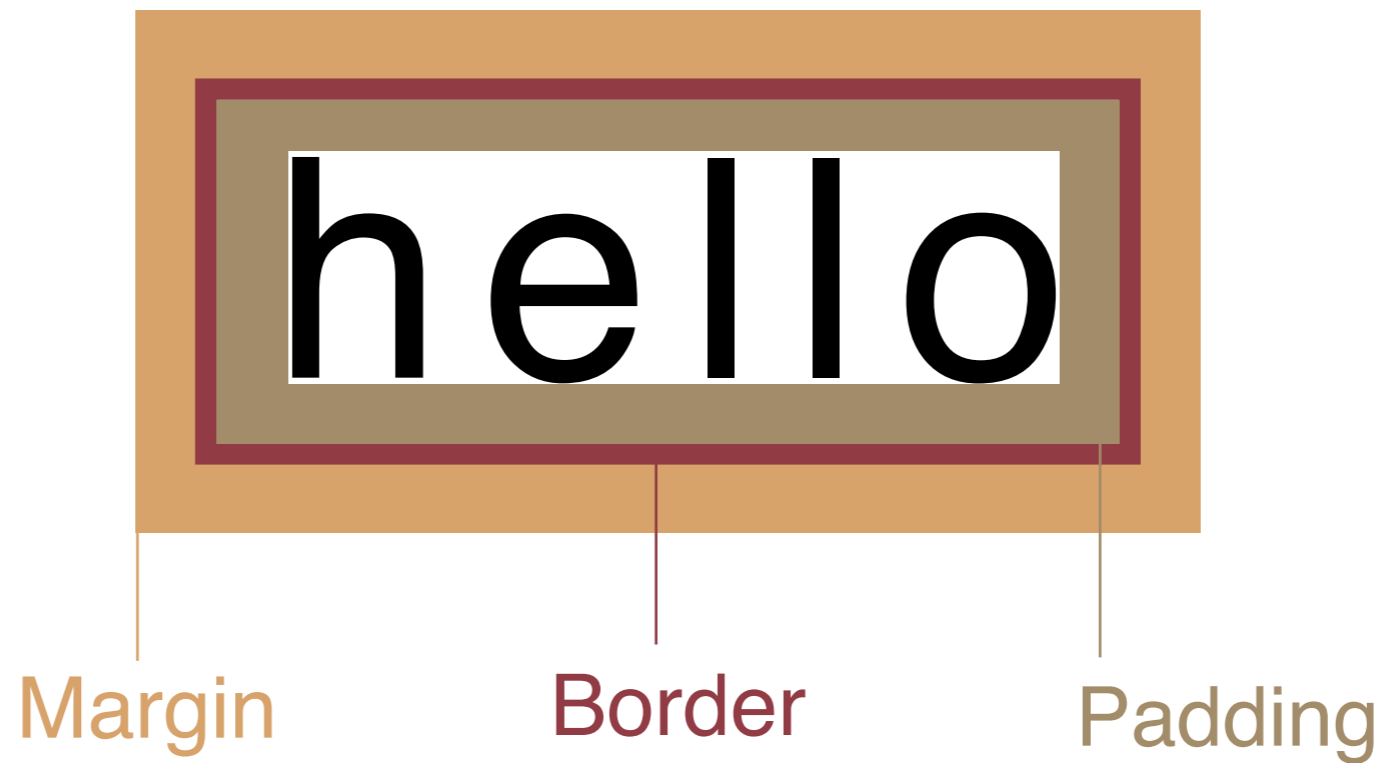
# Margins, Padding, Width



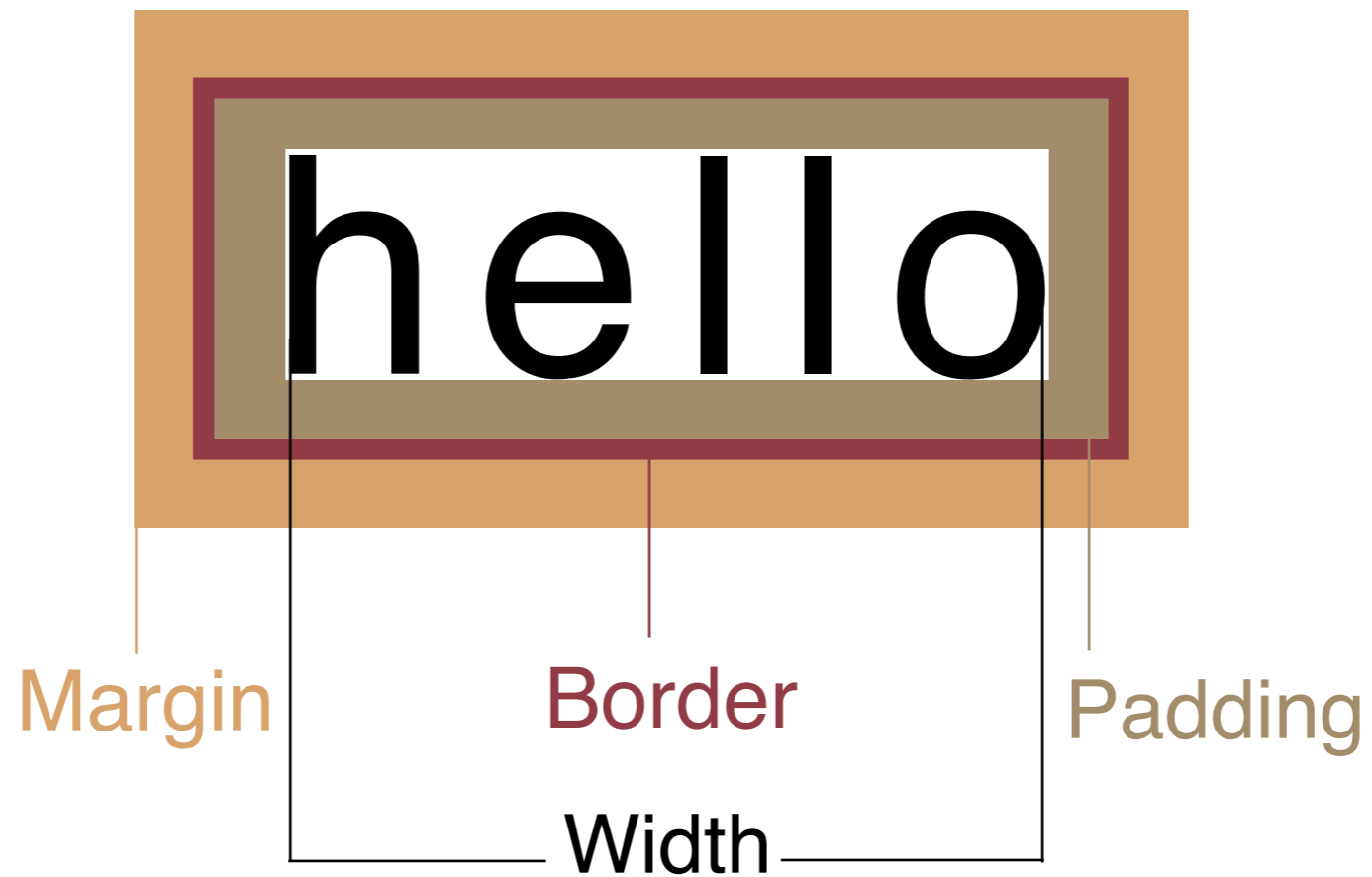
Border

Padding

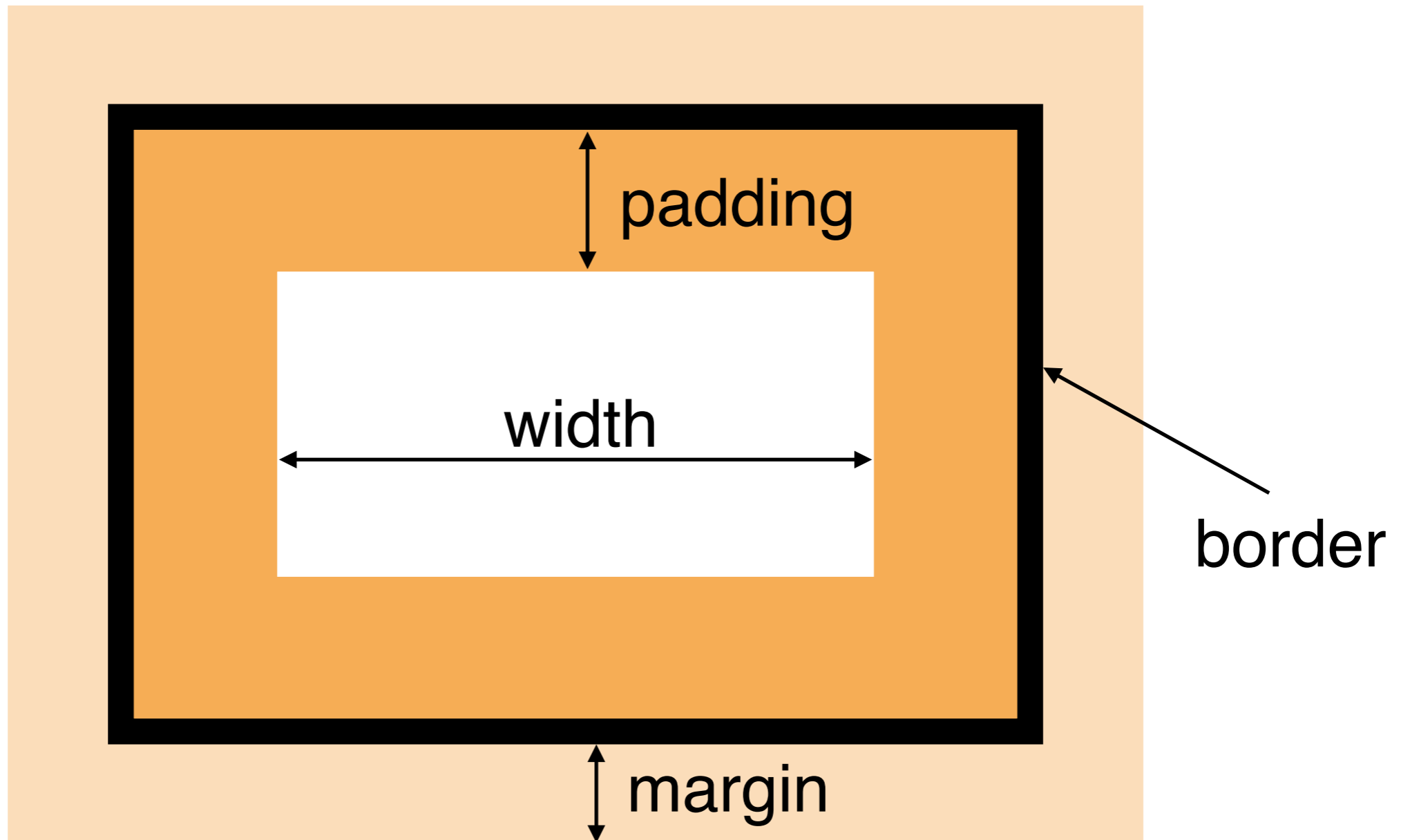
# Margins, Padding, Width



# Margins, Padding, Width



# The Box Model Properties



# Block vs Inline



# Thinking inside the box

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

Block  
Elements

# Thinking inside the box

## The Cottage Garden

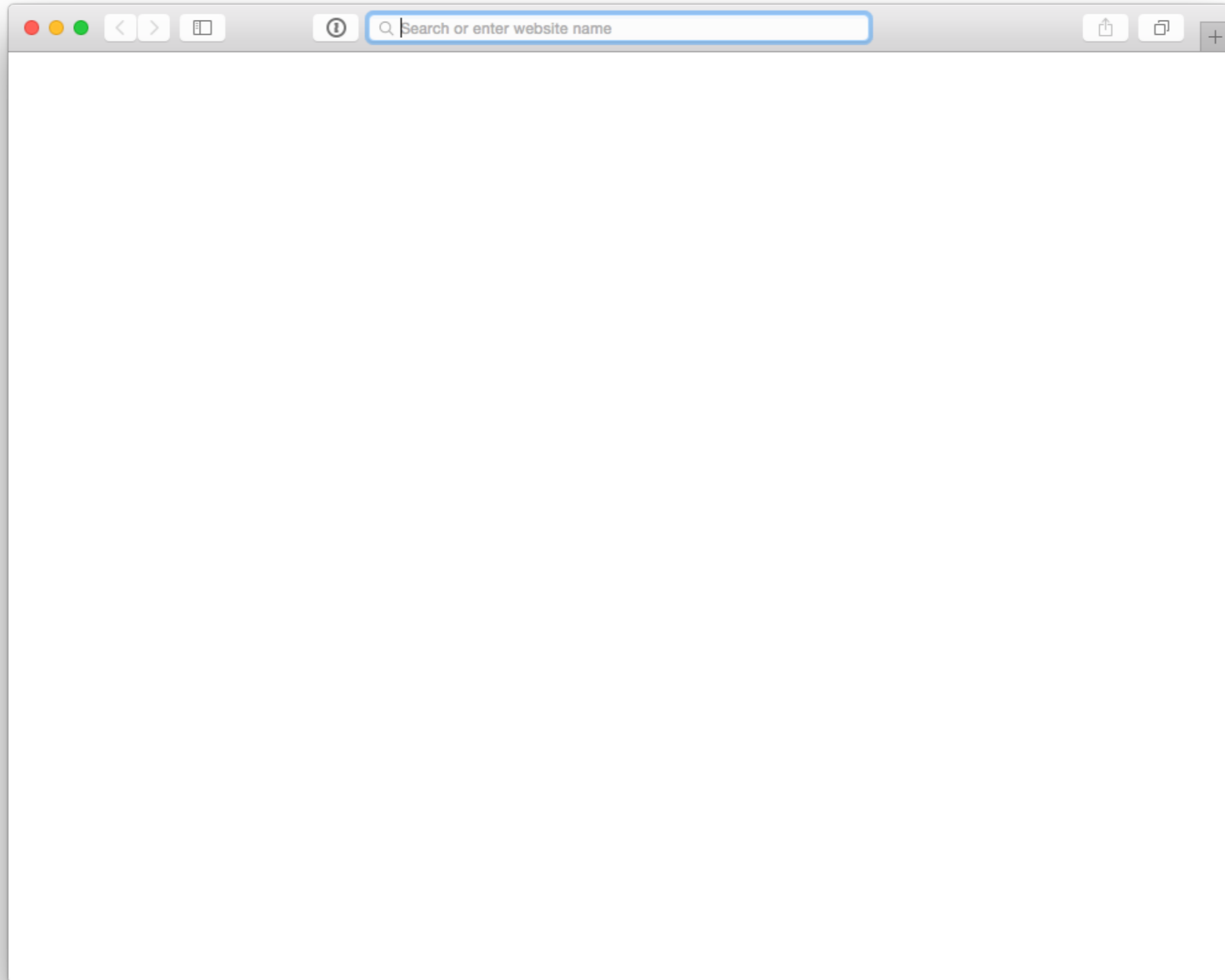
The **cottage garden** is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in **England** and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained **English estate gardens**.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

Inline  
Elements

# Thinking inside the box



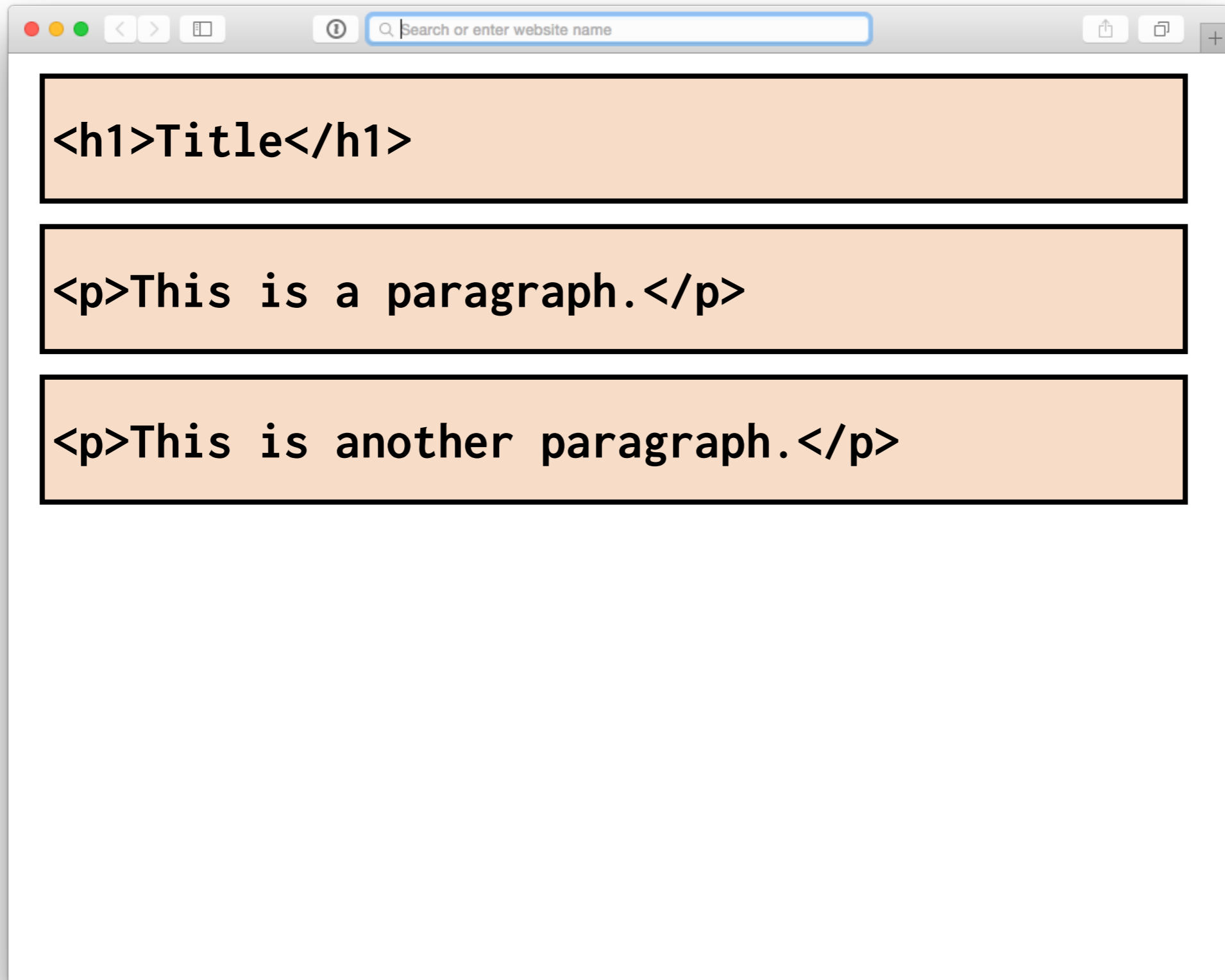
# Thinking inside the box



# Thinking inside the box



# Thinking inside the box

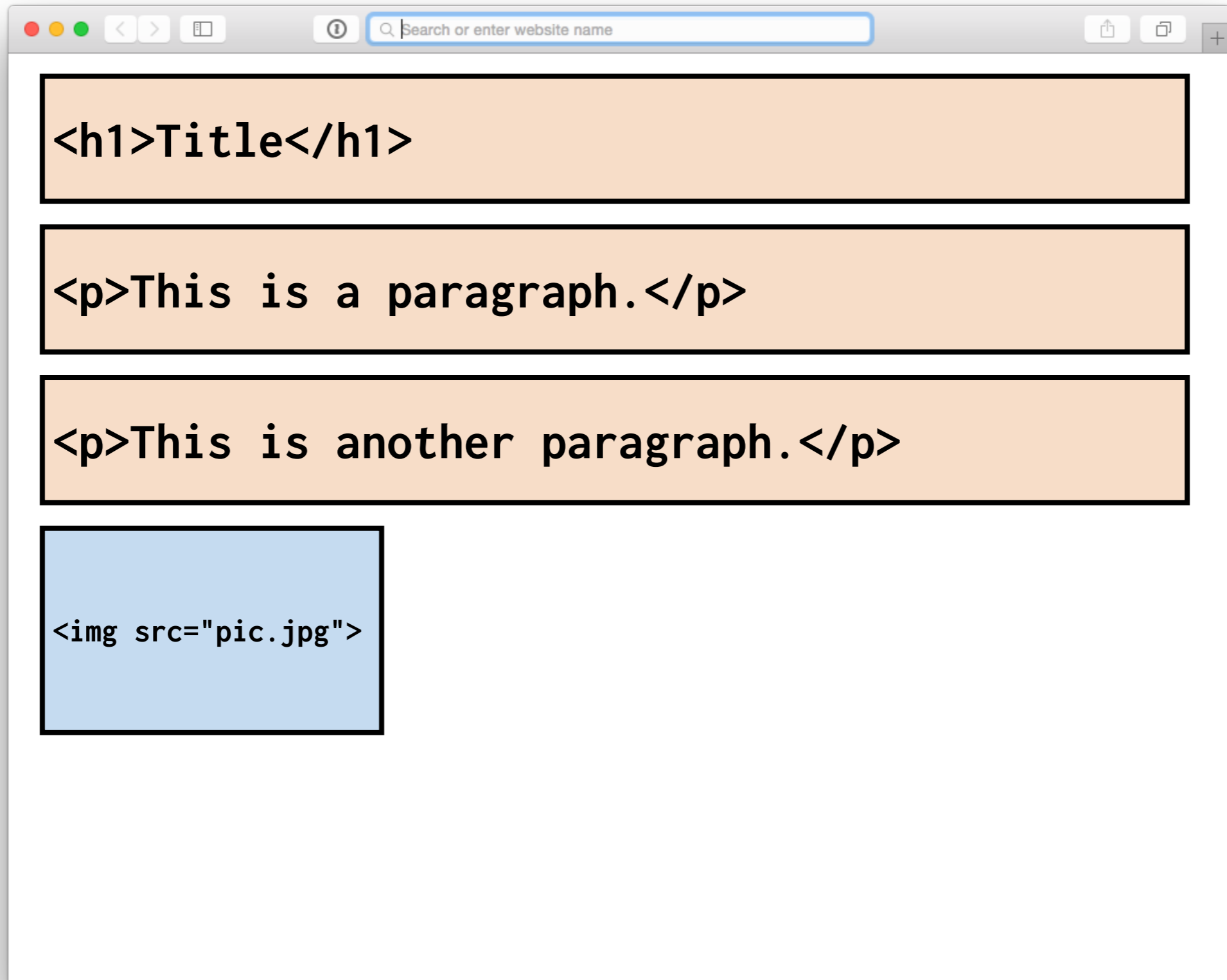


```
<h1>Title</h1>
```

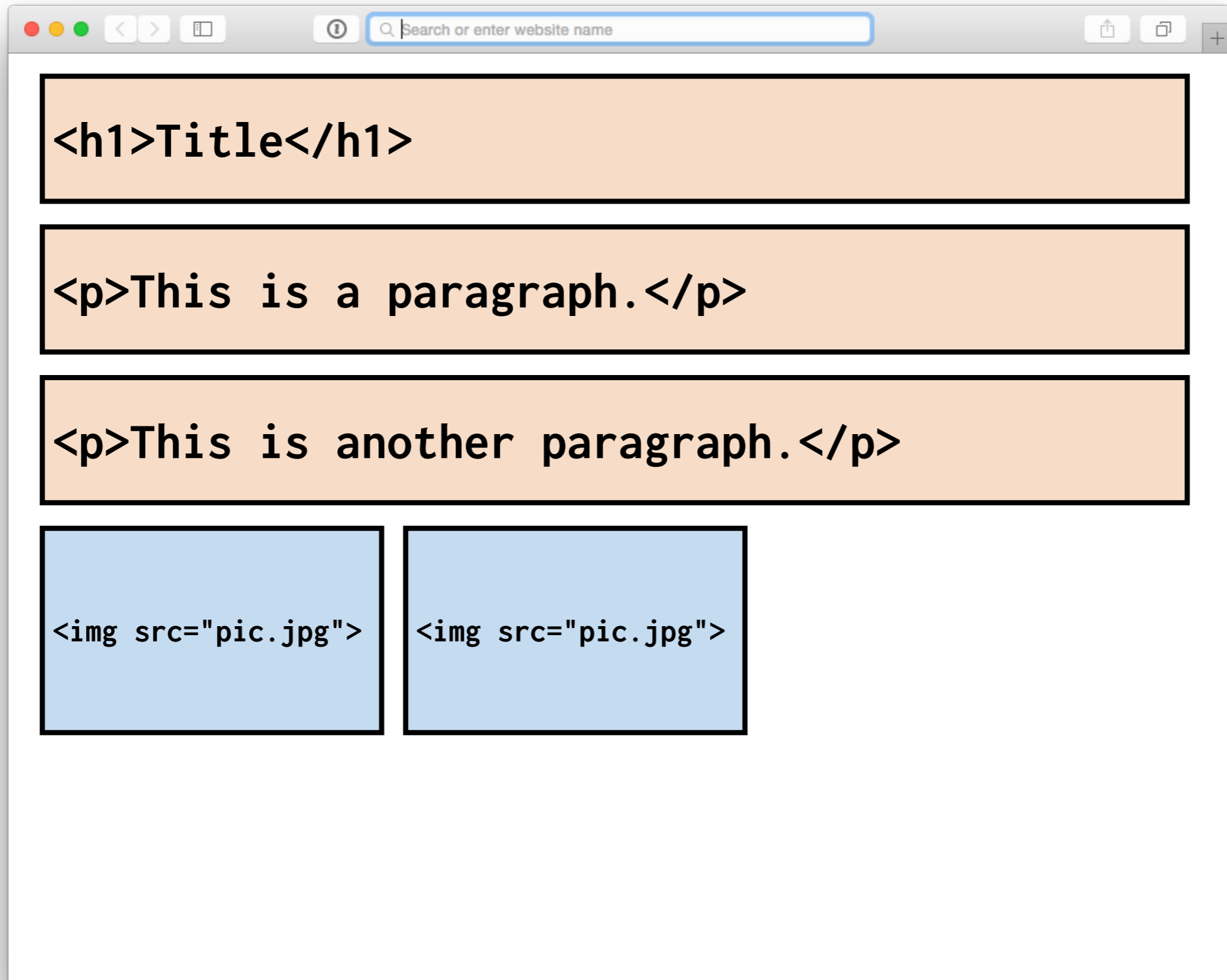
```
<p>This is a paragraph.</p>
```

```
<p>This is another paragraph.</p>
```

# Thinking inside the box

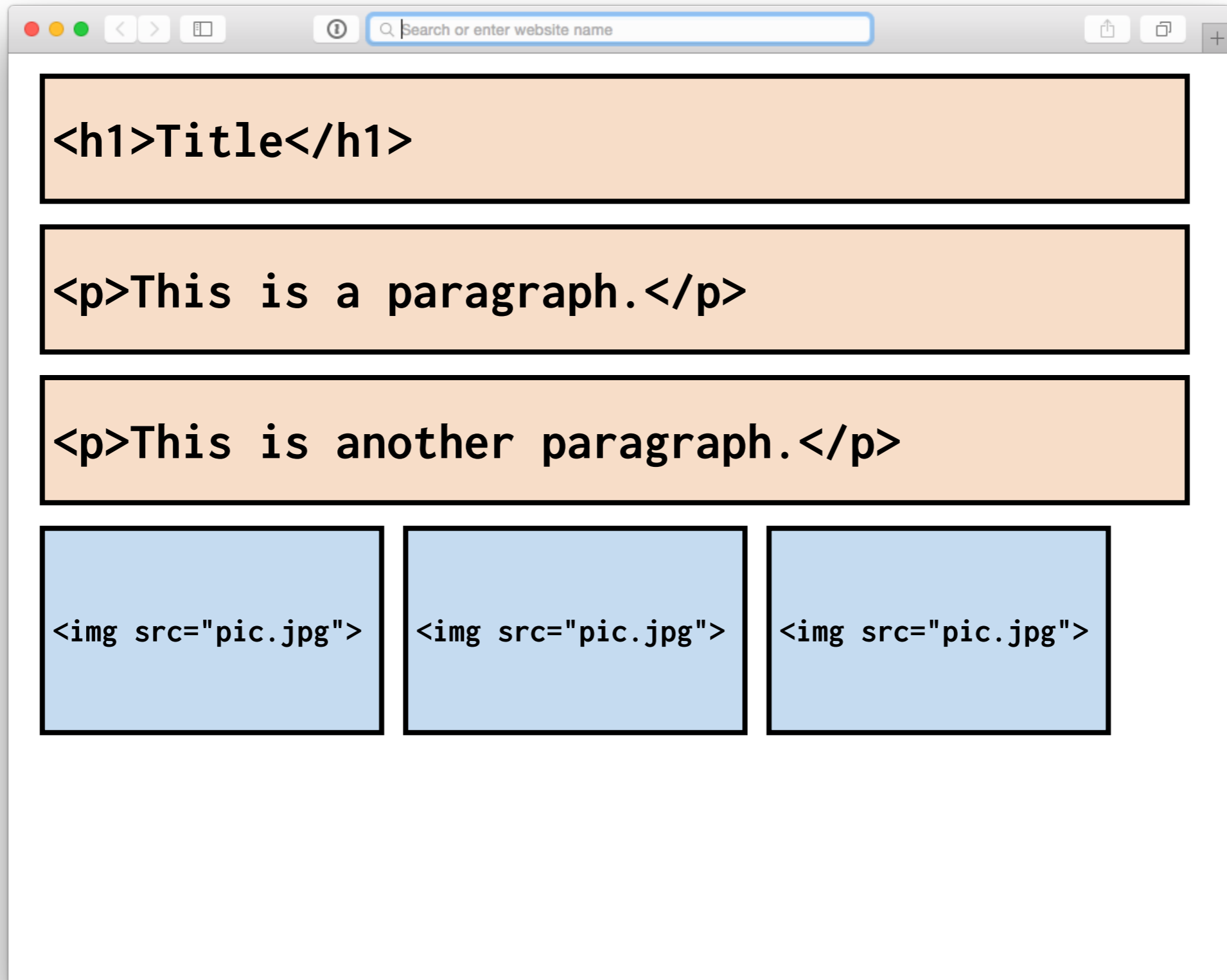


# Thinking inside the box

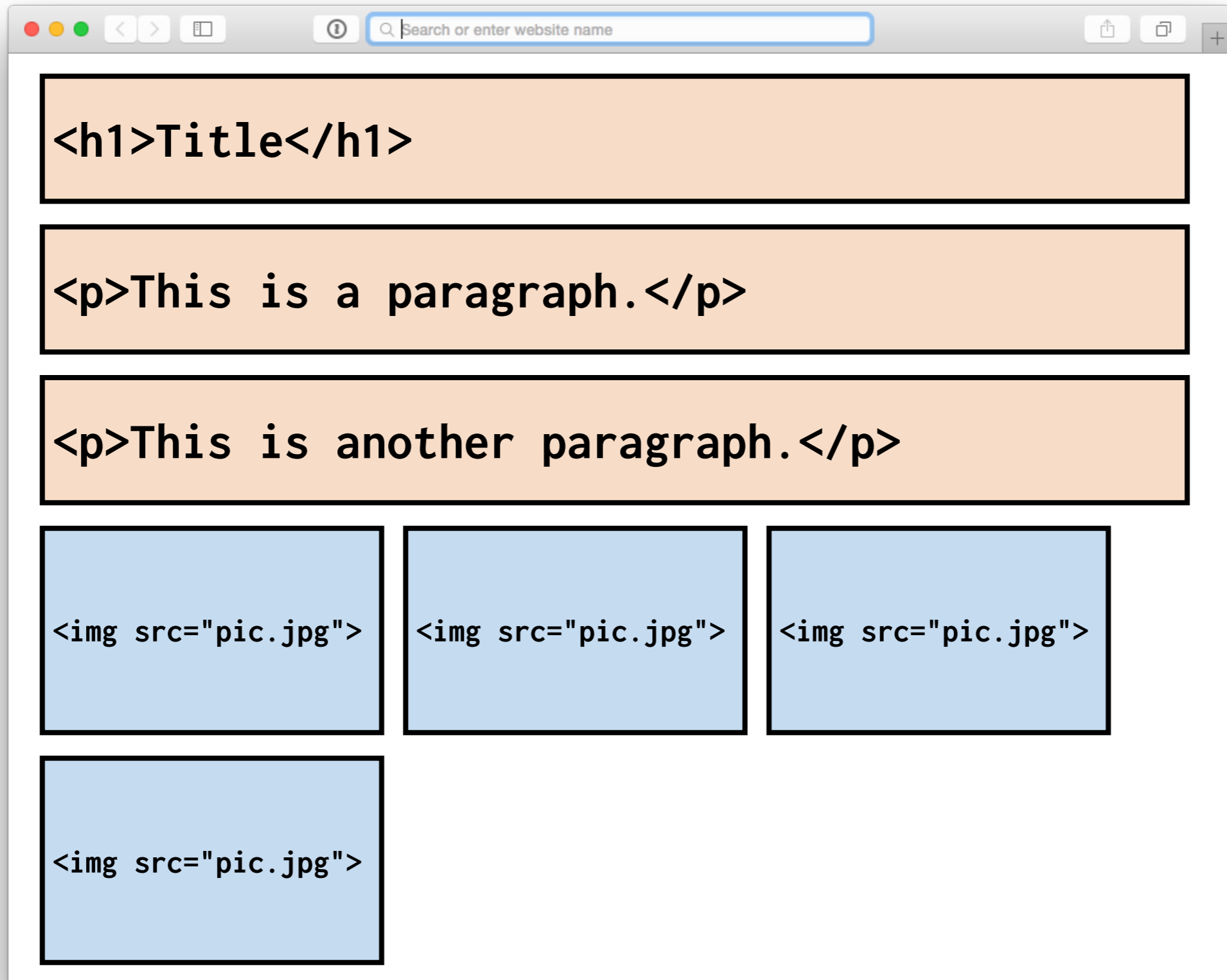




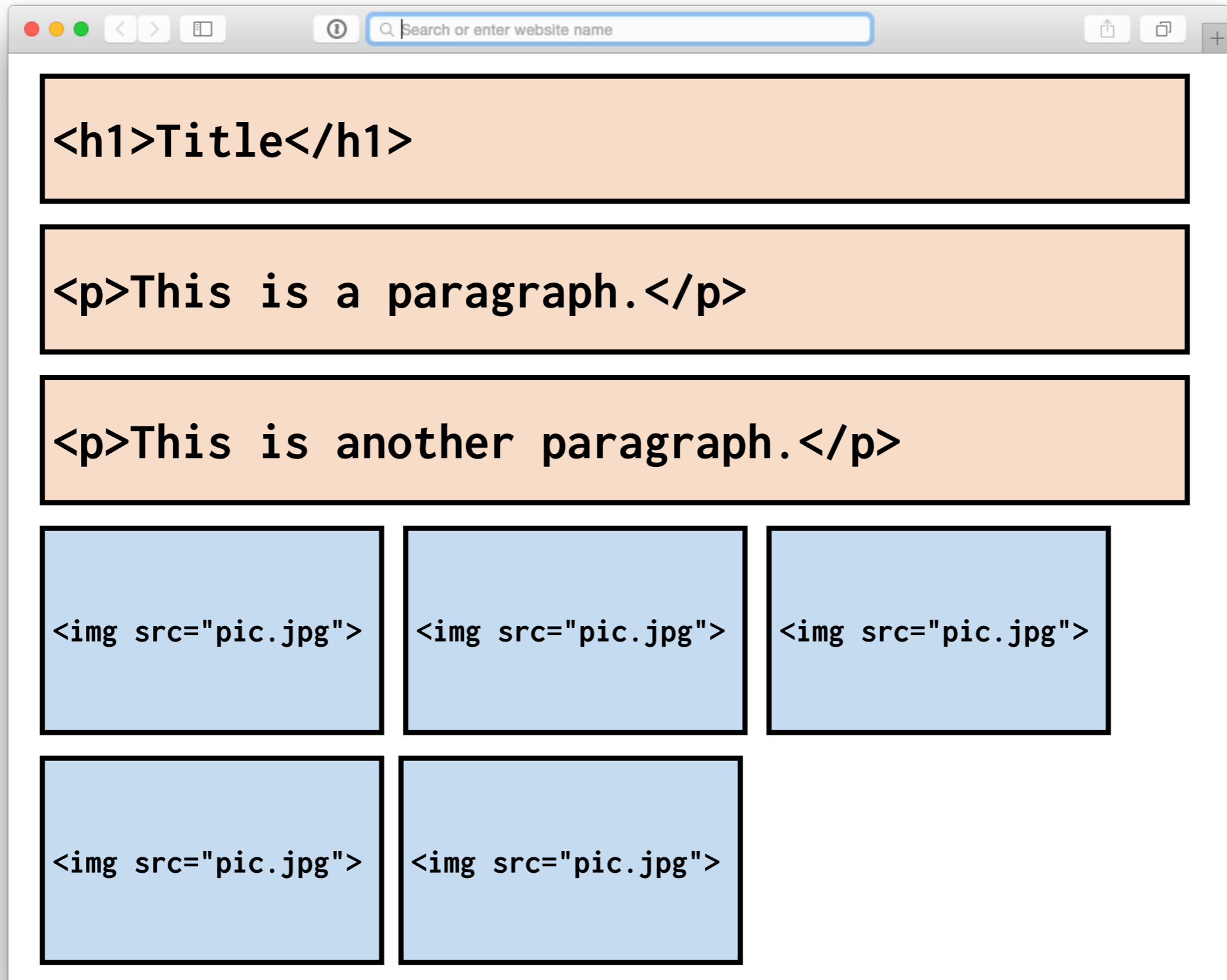
# Thinking inside the box



# Thinking inside the box



# Thinking inside the box





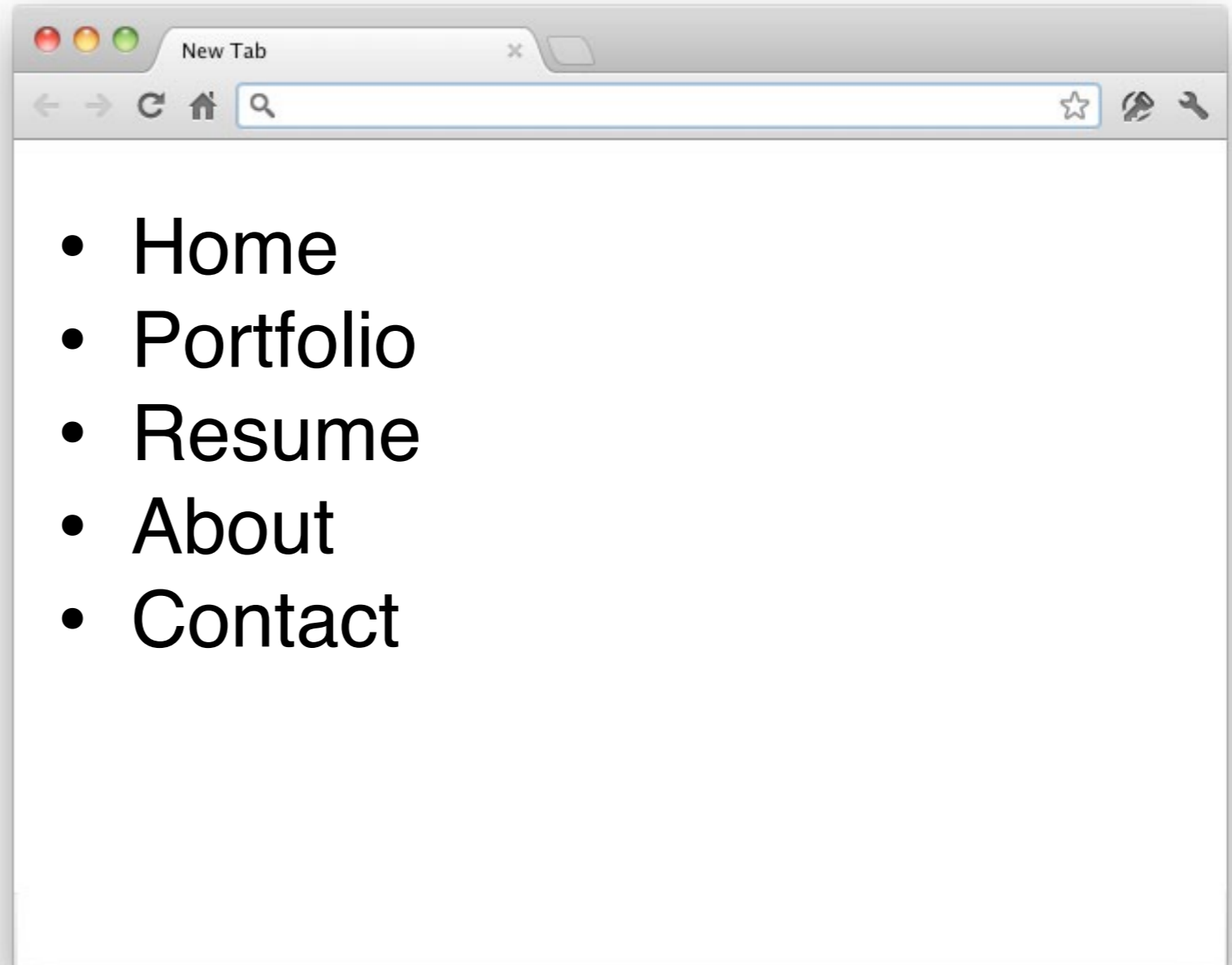
# Inline vs Block

## CSS

```
li {  
  
}
```

## HTML

```
<ul>  
  <li>Home</li>  
  <li>Portfolio</li>  
  <li>Resume</li>  
  <li>About</li>  
  <li>Contact</li>  
</ul>
```



- Home
- Portfolio
- Resume
- About
- Contact

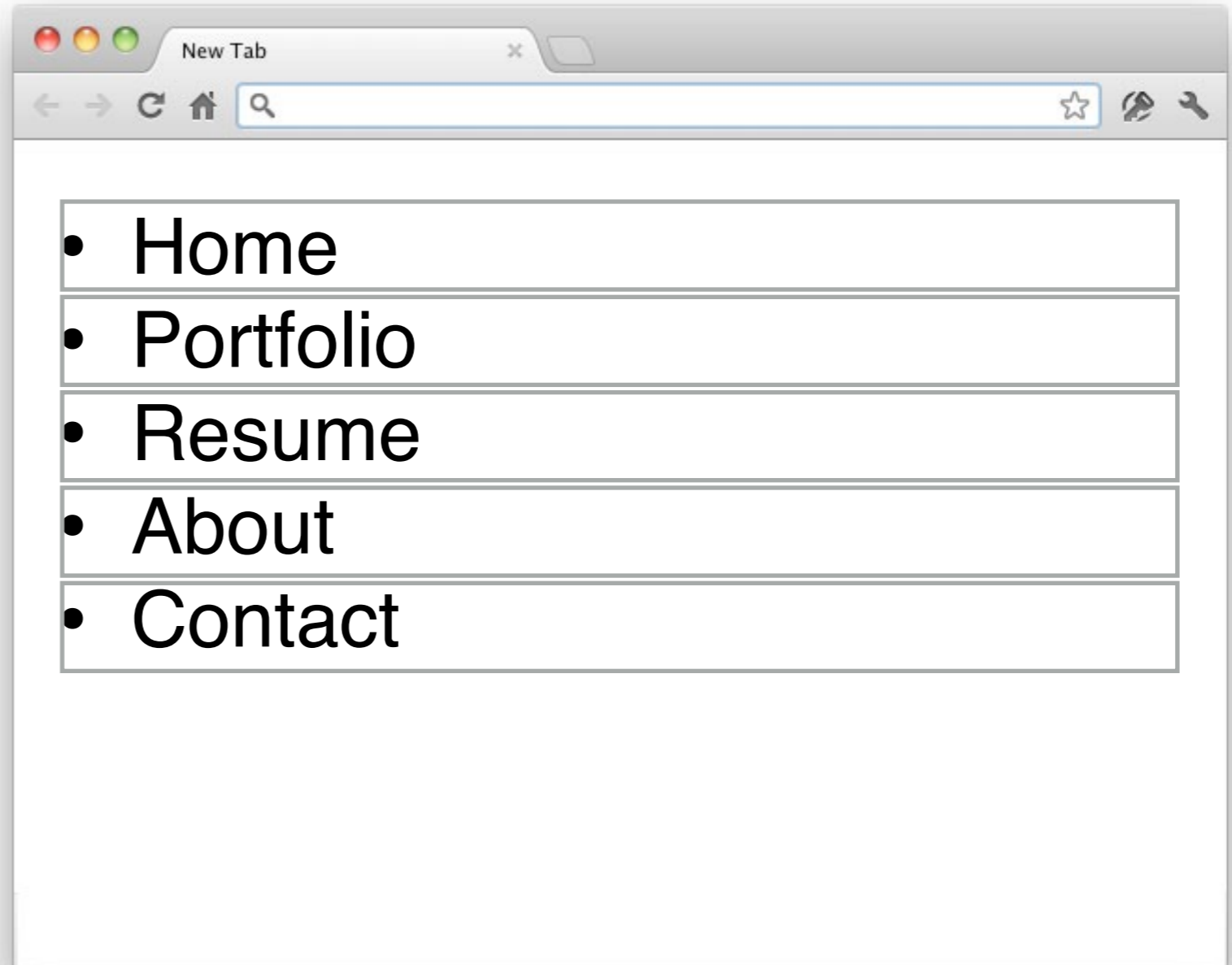
# Inline vs Block

## CSS

```
li {  
  
}
```

## HTML

```
<ul>  
  <li>Home</li>  
  <li>Portfolio</li>  
  <li>Resume</li>  
  <li>About</li>  
  <li>Contact</li>  
</ul>
```



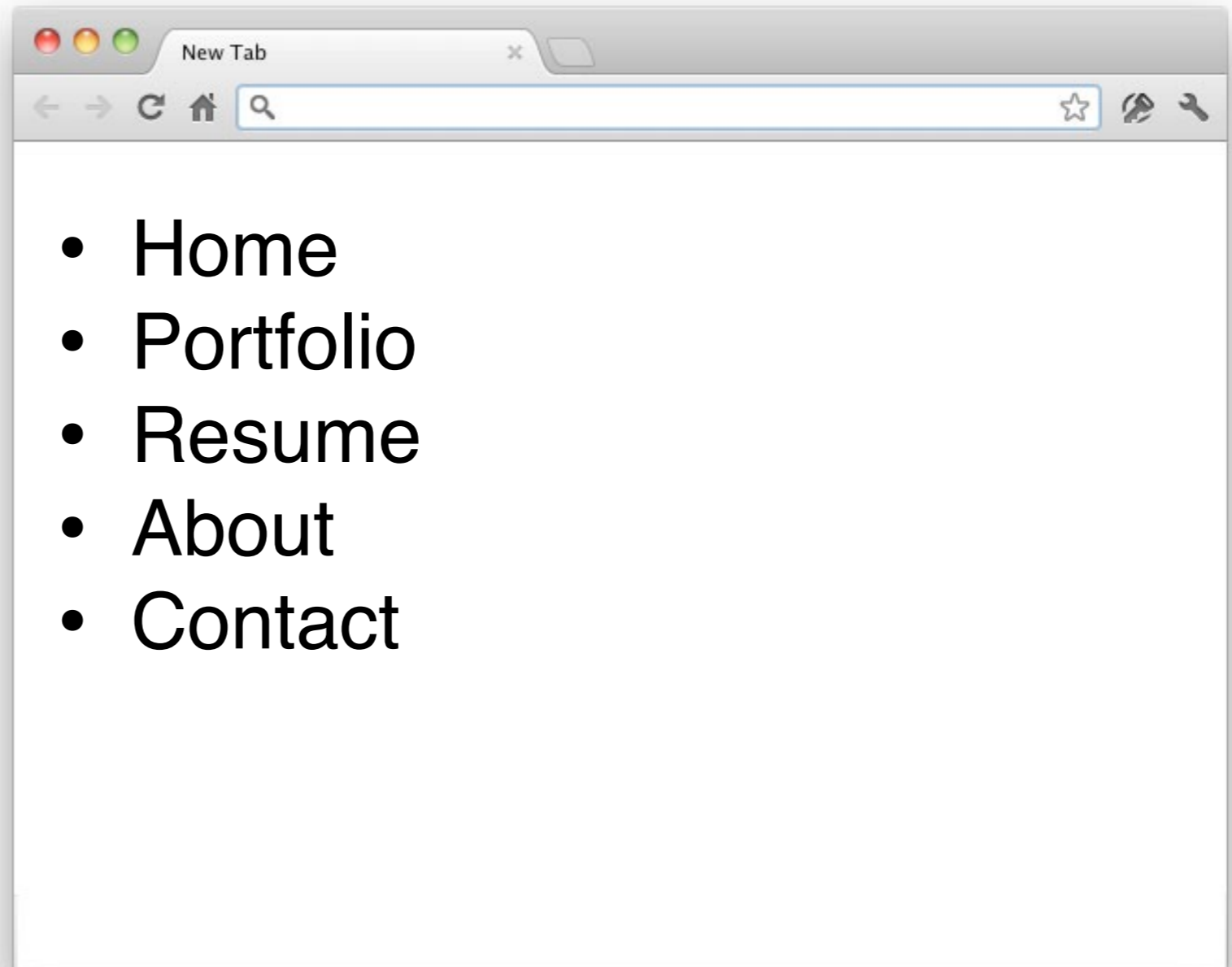
# Inline vs Block

## CSS

```
li {  
  display: inline;  
}
```

## HTML

```
<ul>  
  <li>Home</li>  
  <li>Portfolio</li>  
  <li>Resume</li>  
  <li>About</li>  
  <li>Contact</li>  
</ul>
```



- Home
- Portfolio
- Resume
- About
- Contact

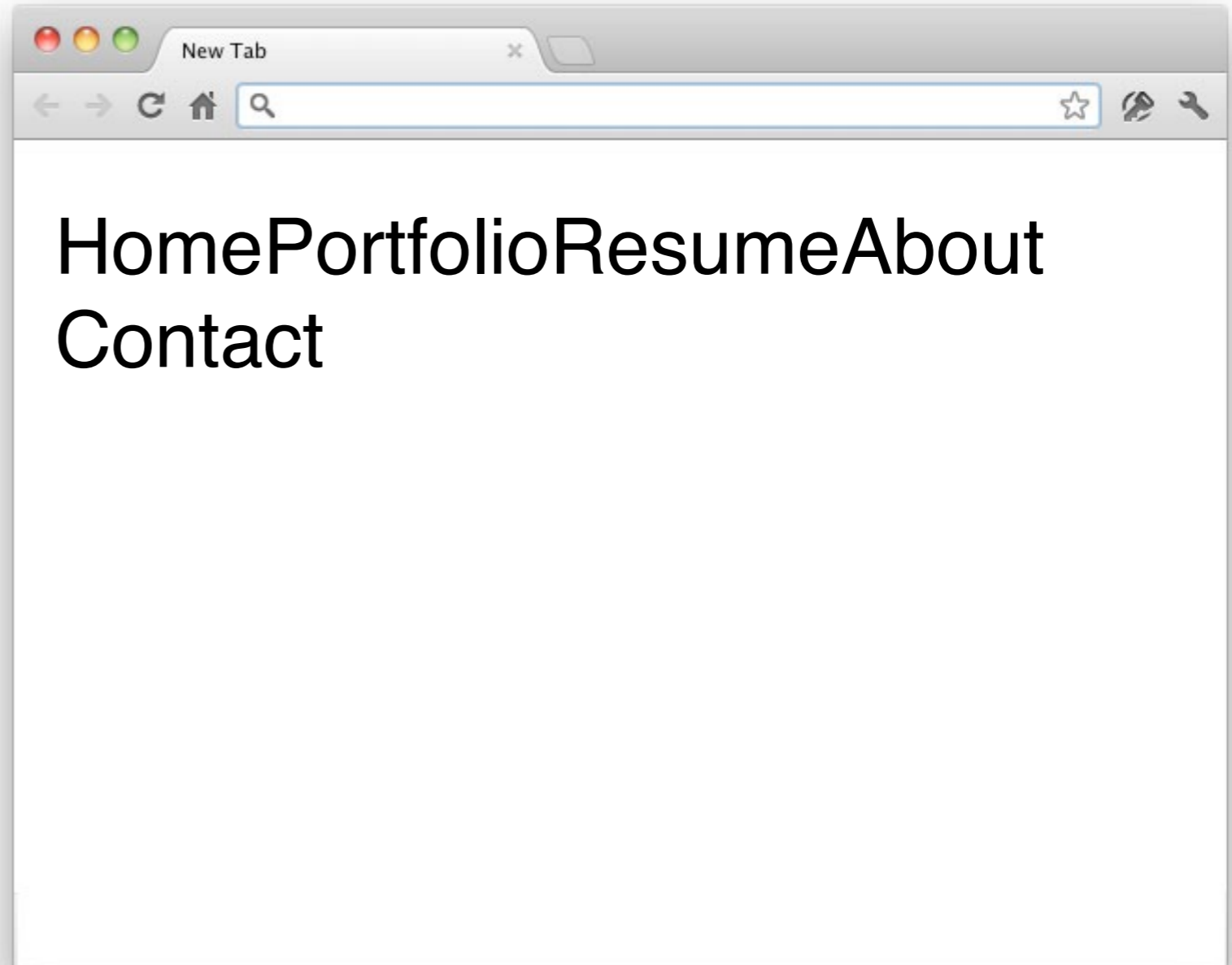
# Inline vs Block

## CSS

```
li {  
  display: inline;  
  
}
```

## HTML

```
<ul>  
  <li>Home</li>  
  <li>Portfolio</li>  
  <li>Resume</li>  
  <li>About</li>  
  <li>Contact</li>  
</ul>
```





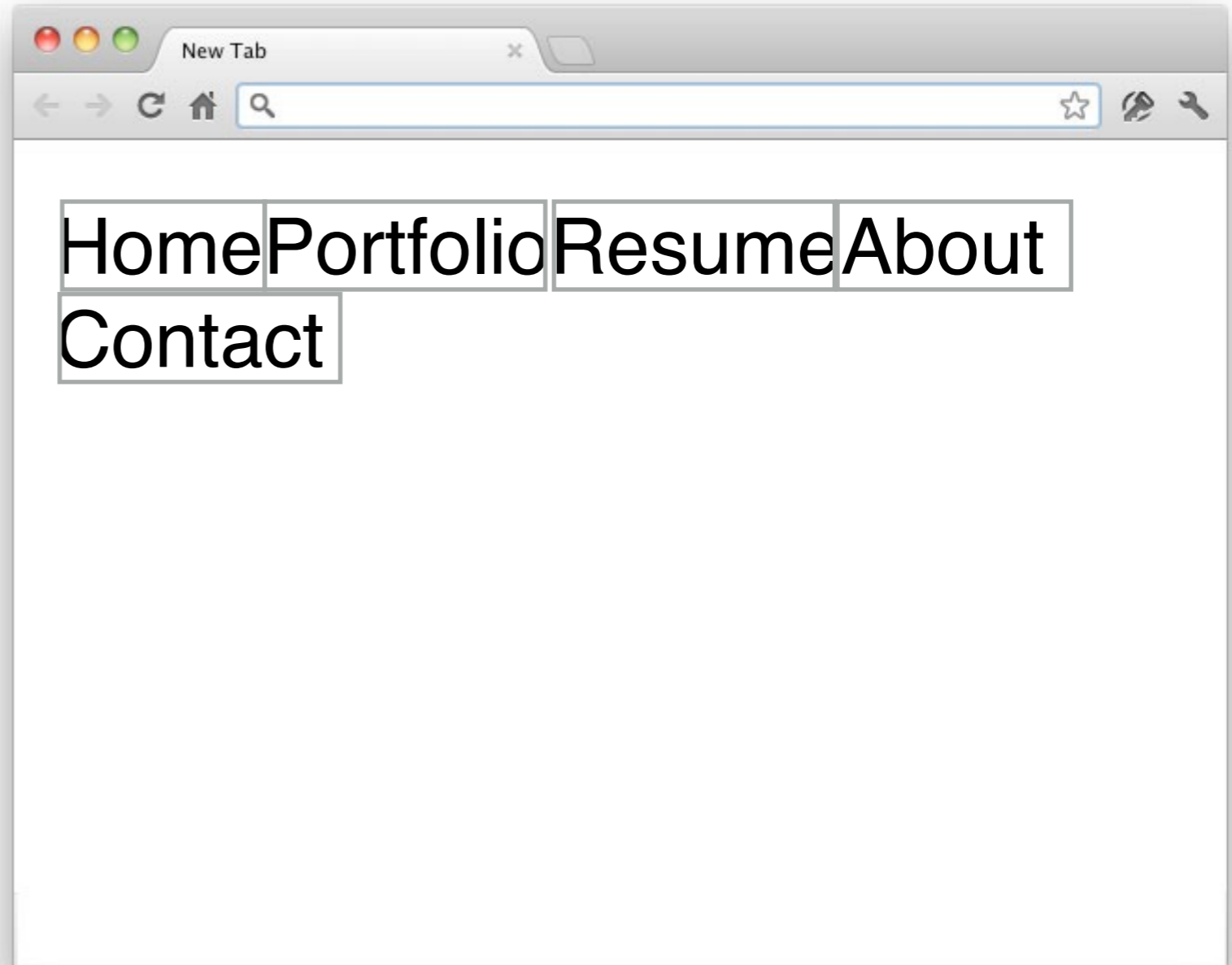
# Inline vs Block

## CSS

```
li {  
  display: inline;  
  
}
```

## HTML

```
<ul>  
  <li>Home</li>  
  <li>Portfolio</li>  
  <li>Resume</li>  
  <li>About</li>  
  <li>Contact</li>  
</ul>
```



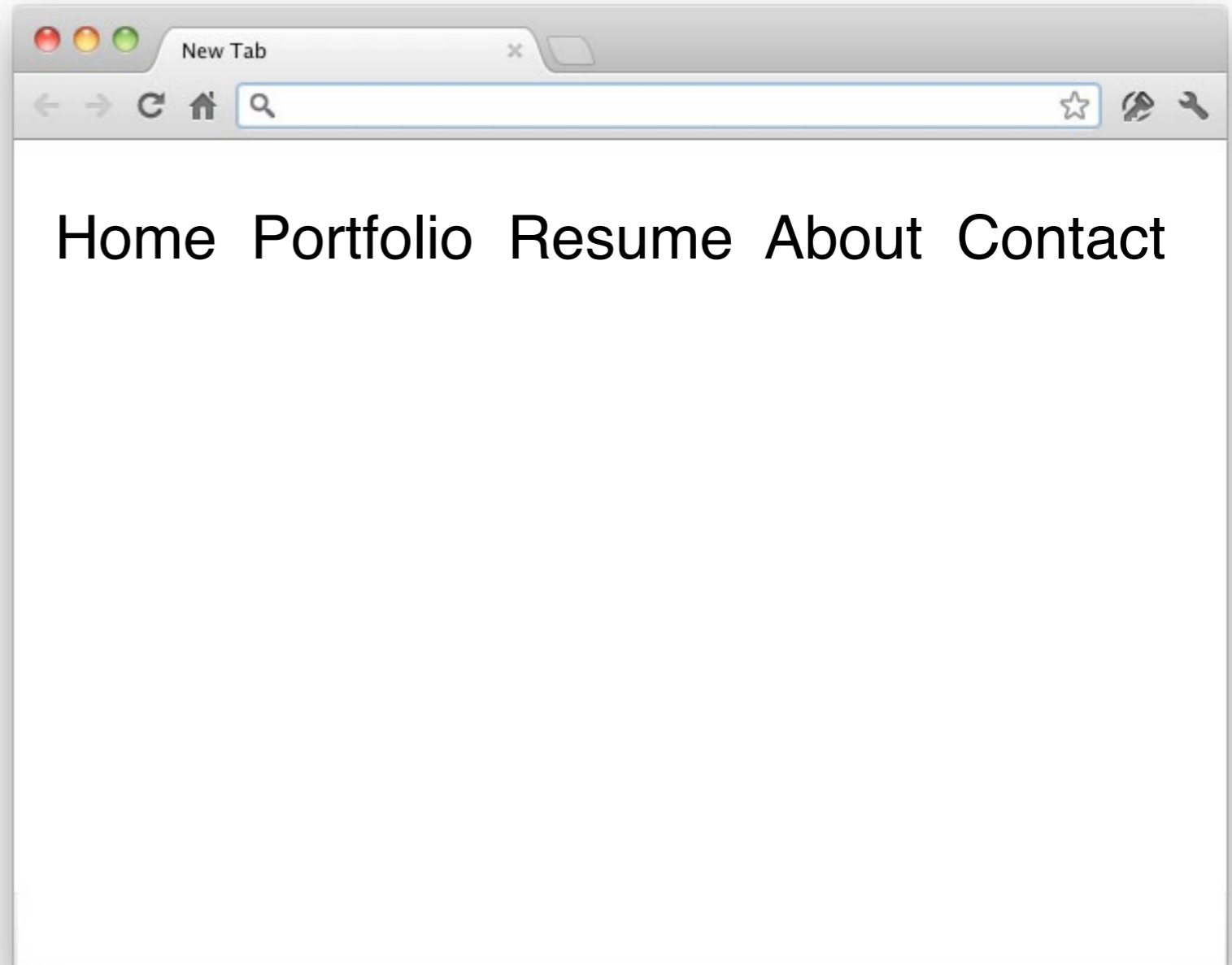
# Inline vs Block

## CSS

```
li {  
  display: inline;  
  padding: 5px;  
  font-size: 12px;  
}
```

## HTML

```
<ul>  
  <li>Home</li>  
  <li>Portfolio</li>  
  <li>Resume</li>  
  <li>About</li>  
  <li>Contact</li>  
</ul>
```



**Learning the rules**

# Rules for Block-level Elements

- By default a box will have a width of "auto", or the full width of its container.
- By default a box will have a height that conforms to the content of that box.
- If you set a height in CSS, that overrides the content.
- By default a box will stack from the top down.

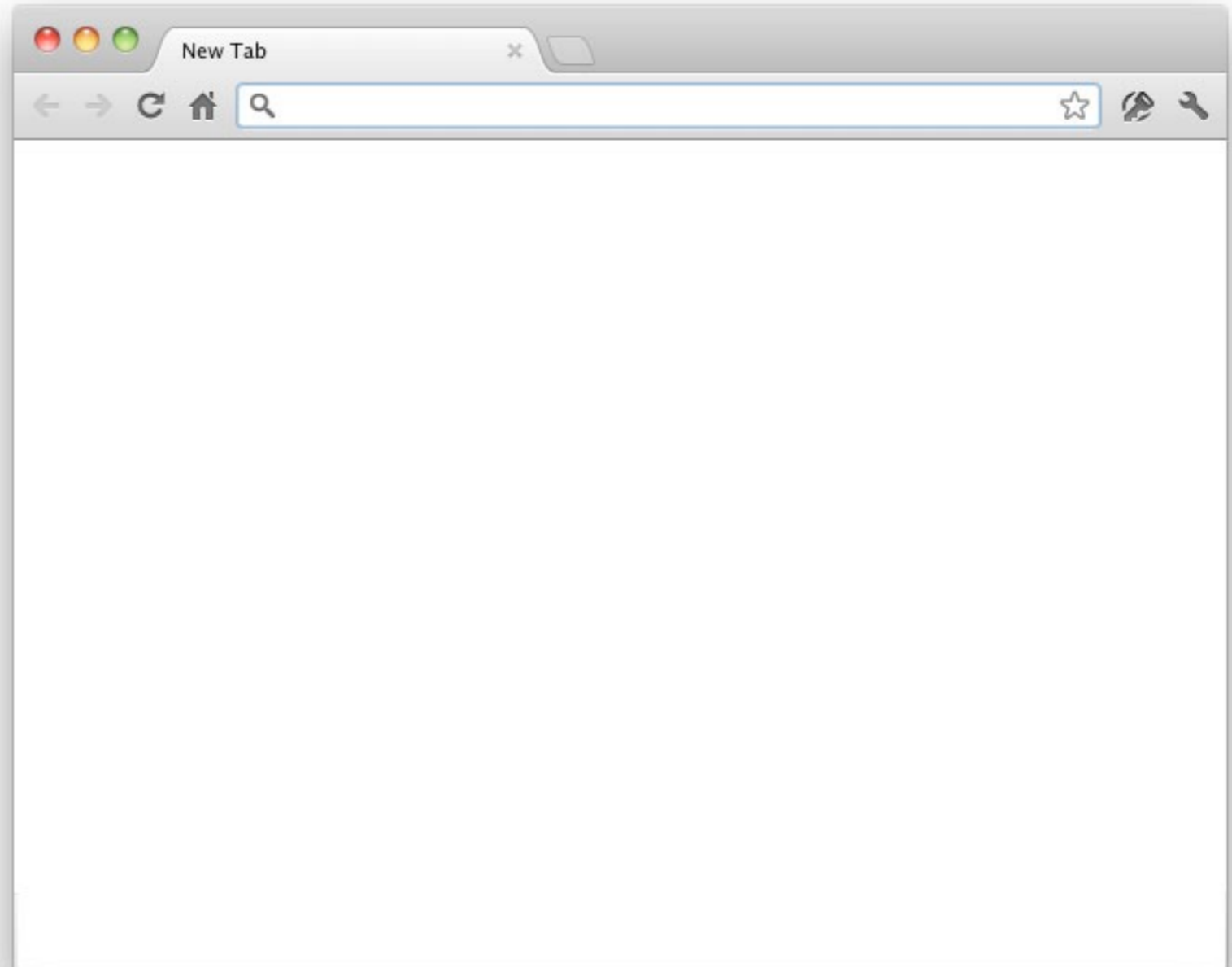
# Rules of Block Elements

## CSS

```
div {  
  
}
```

## HTML

```
<div></div>
```



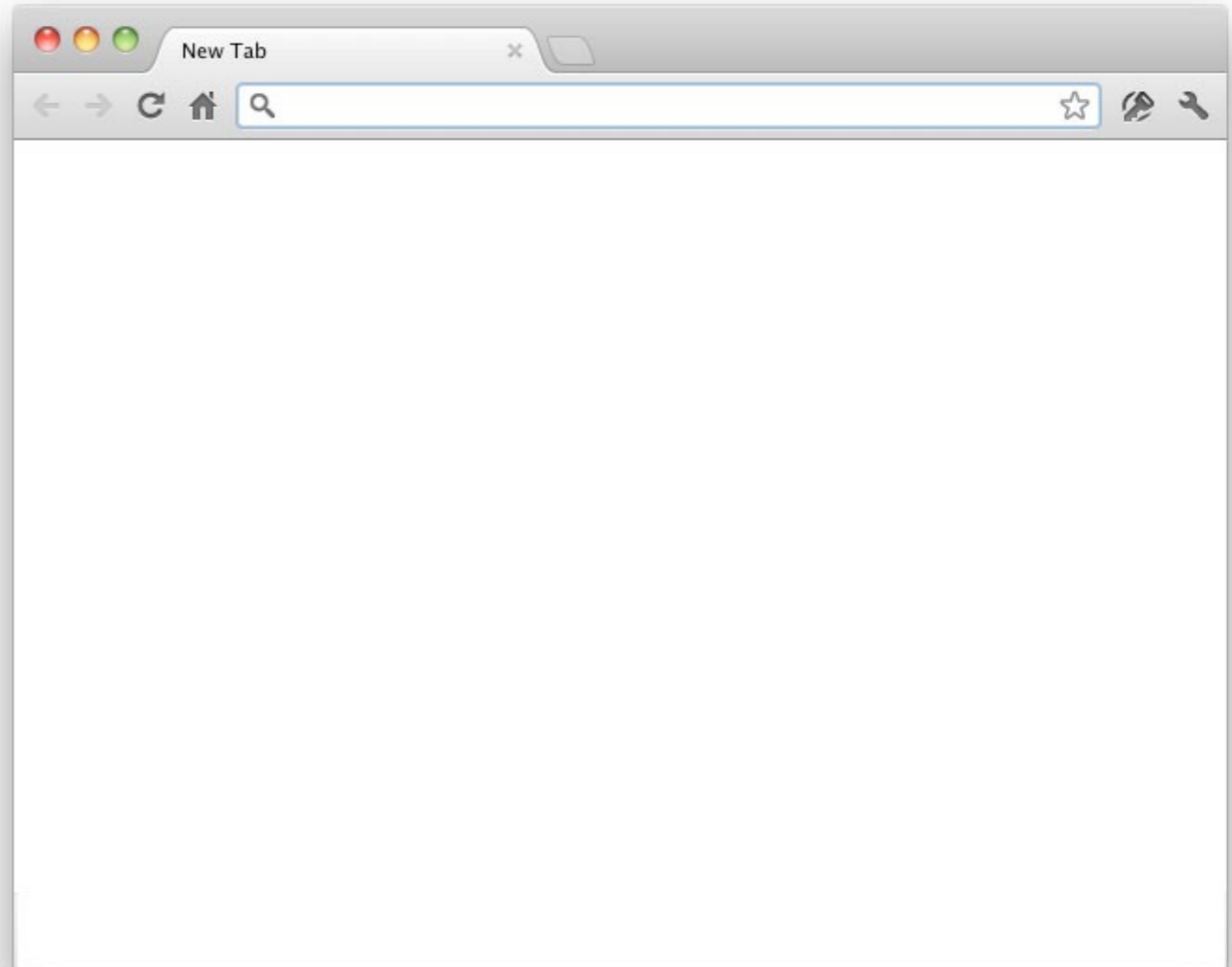
# Rules of Block Elements

## CSS

```
div {  
  border:1px solid black;  
  
}
```

## HTML

```
<div></div>
```



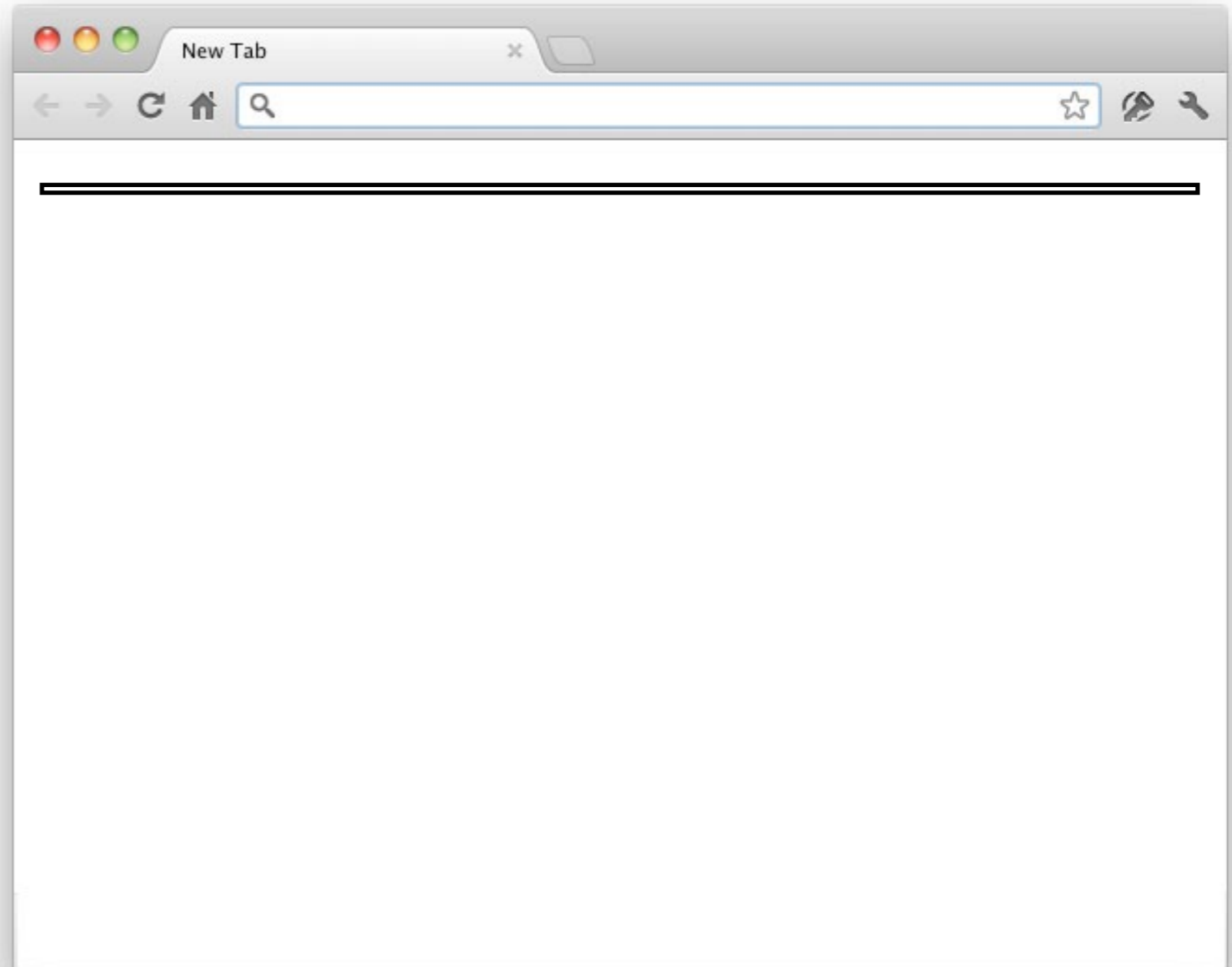
# Rules of Block Elements

## CSS

```
div {  
  border:1px solid black;  
  
}
```

## HTML

```
<div></div>
```



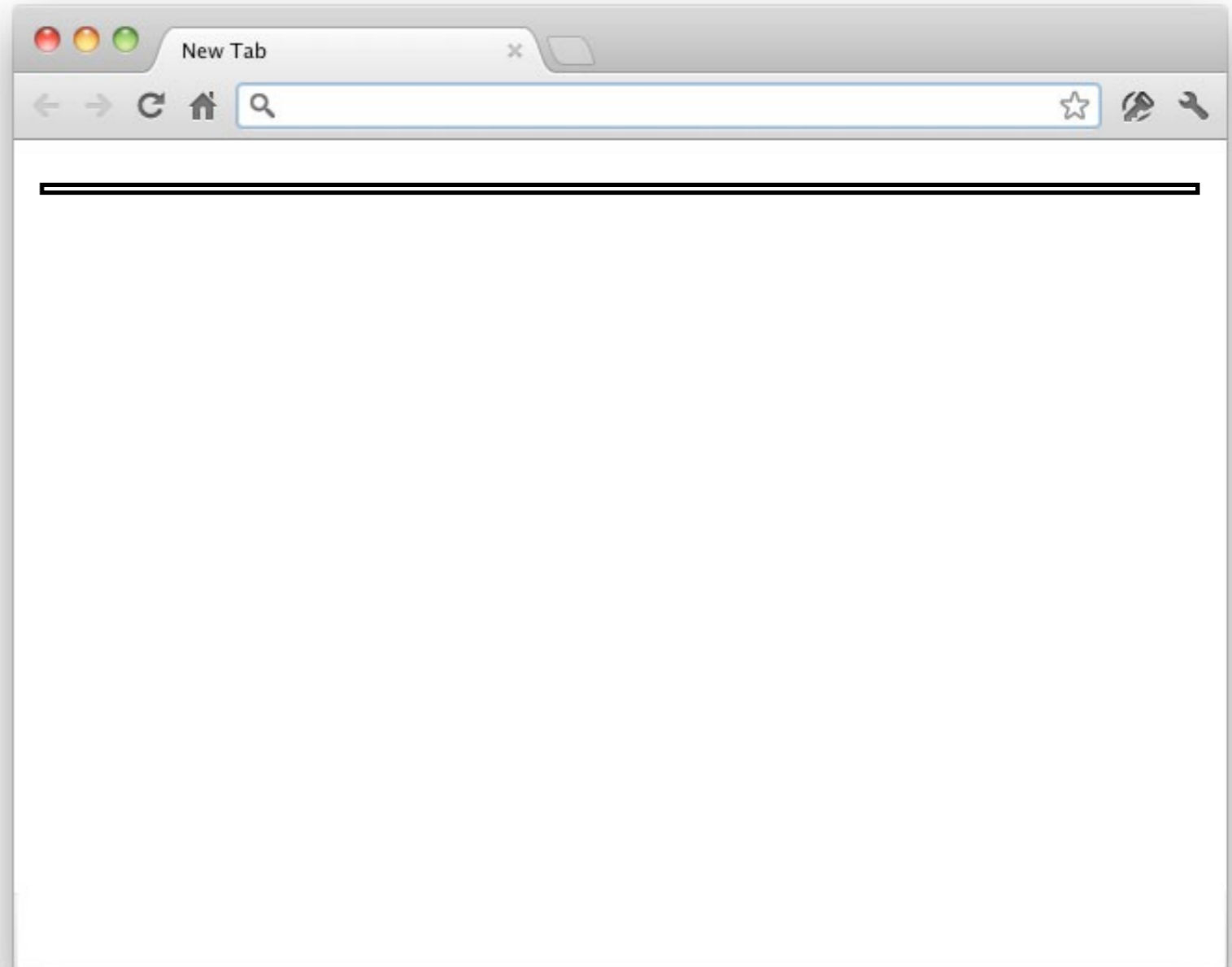
# Rules of Block Elements

## CSS

```
div {  
  border:1px solid black;  
  
}
```

## HTML

```
<div>  
Hello World!  
</div>
```





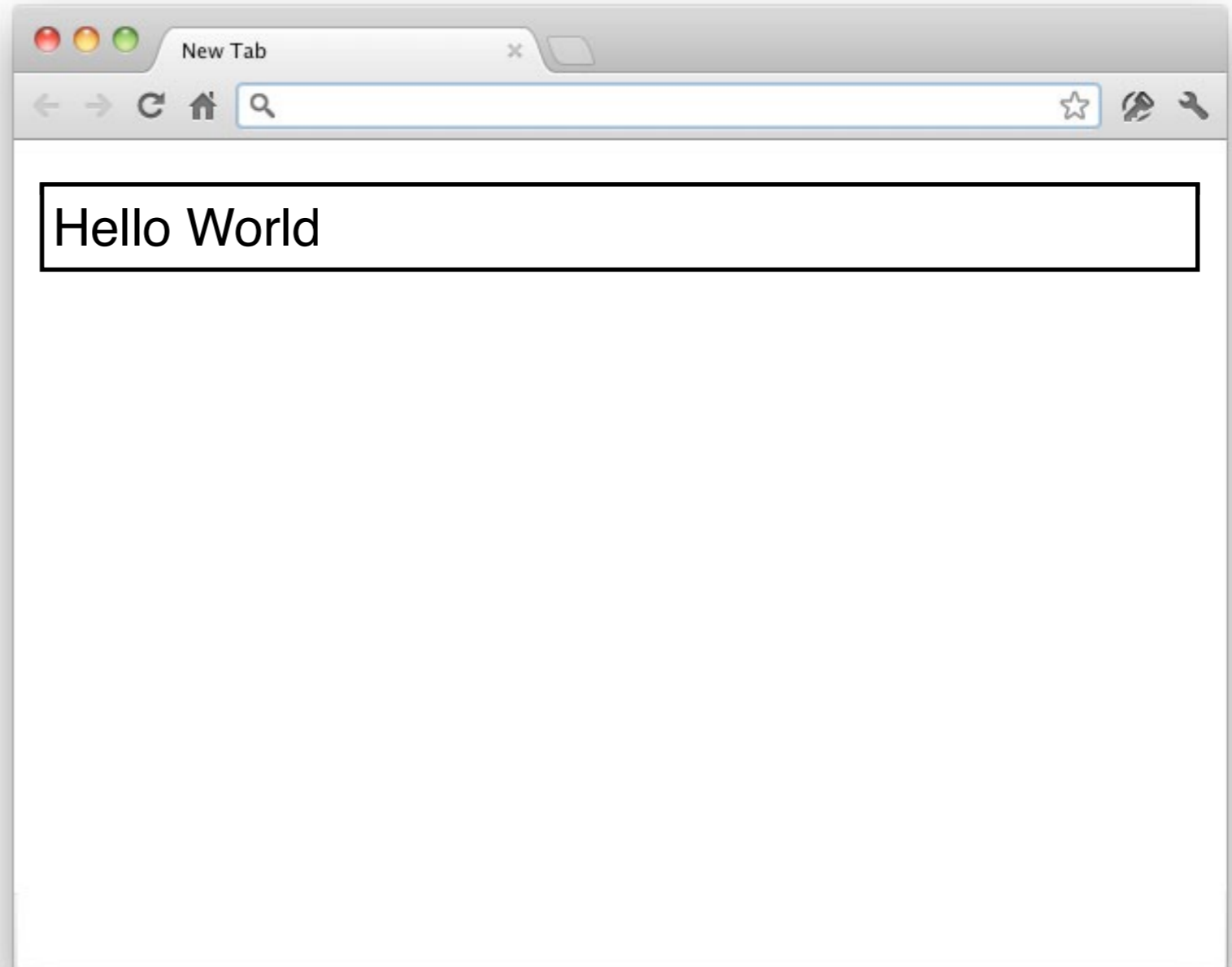
# Rules of Block Elements

## CSS

```
div {  
  border:1px solid black;  
  
}
```

## HTML

```
<div>  
Hello World!  
</div>
```



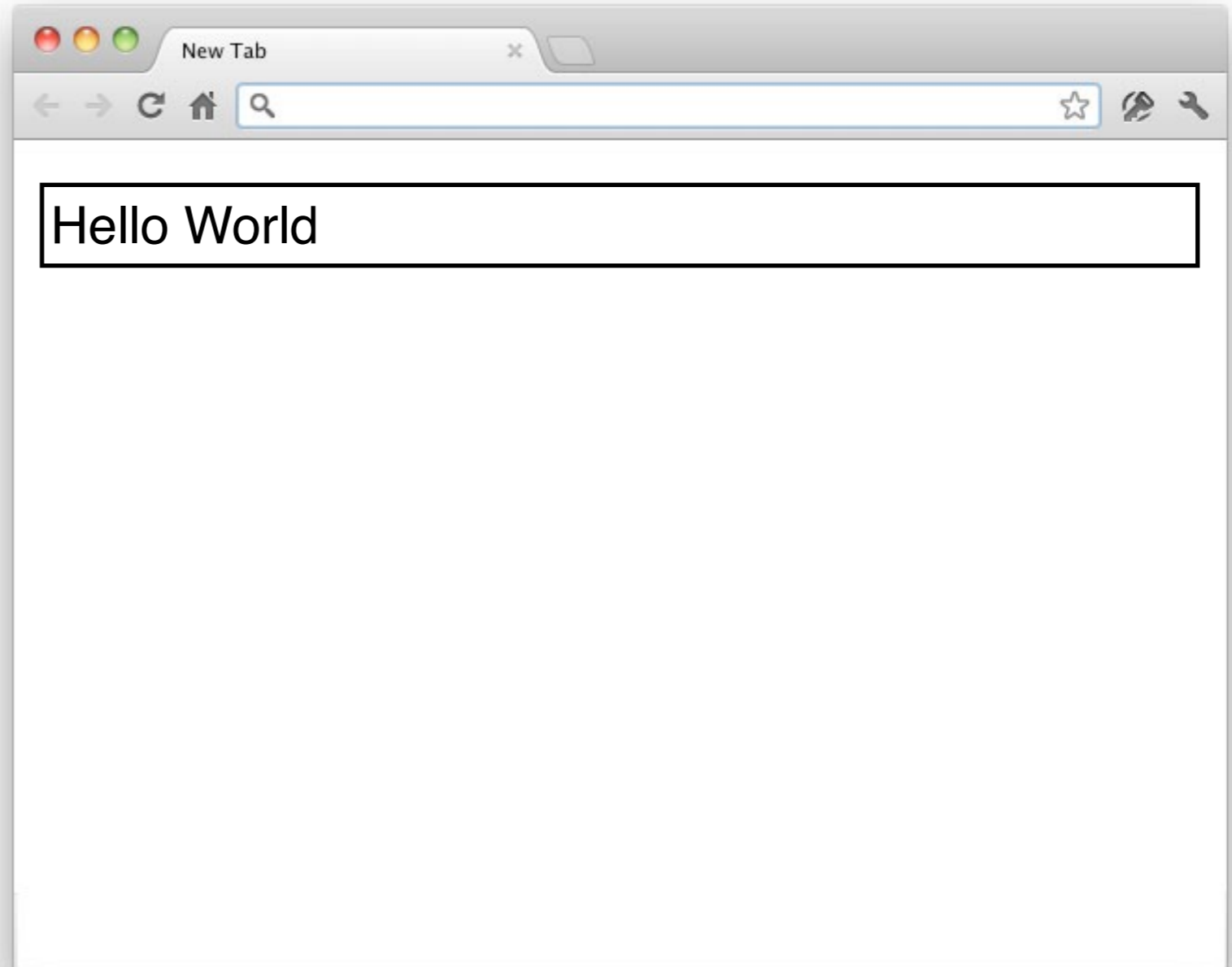
# Rules of Block Elements

## CSS

```
div {  
  border: 1px solid black;  
  height: 5px;  
  
}
```

## HTML

```
<div>  
Hello World!  
</div>
```



# Rules of Block Elements

## CSS

```
div {  
  border: 1px solid black;  
  height: 5px;  
  
}
```

## HTML

```
<div>  
Hello World!  
</div>
```



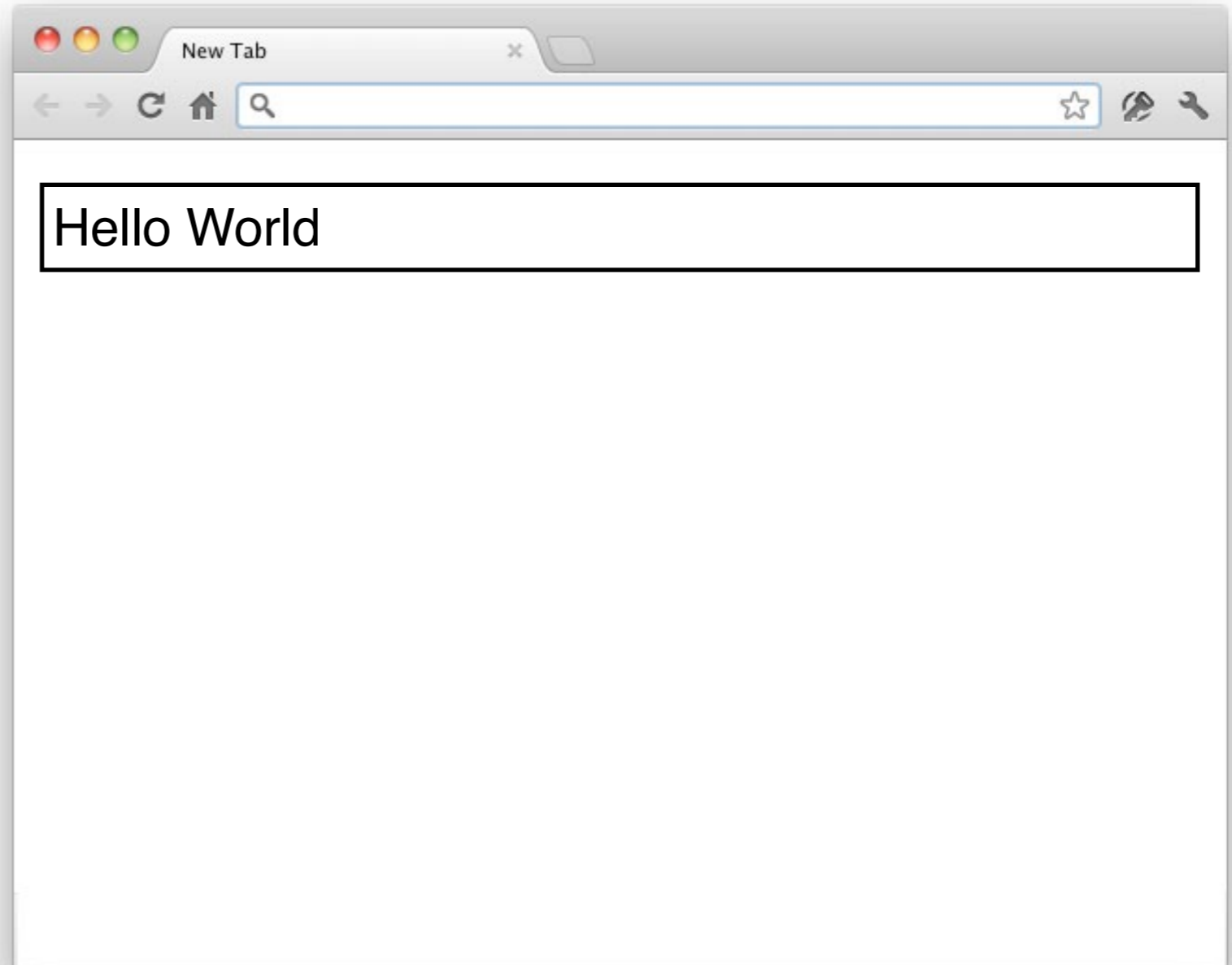
# Rules of Block Elements

## CSS

```
div {  
  border:1px solid black;  
  
}
```

## HTML

```
<div>Hello World!</div>  
<div>Hello World!</div>  
<div>Hello World!</div>  
<div>Hello World!</div>  
<div>Hello World!</div>
```



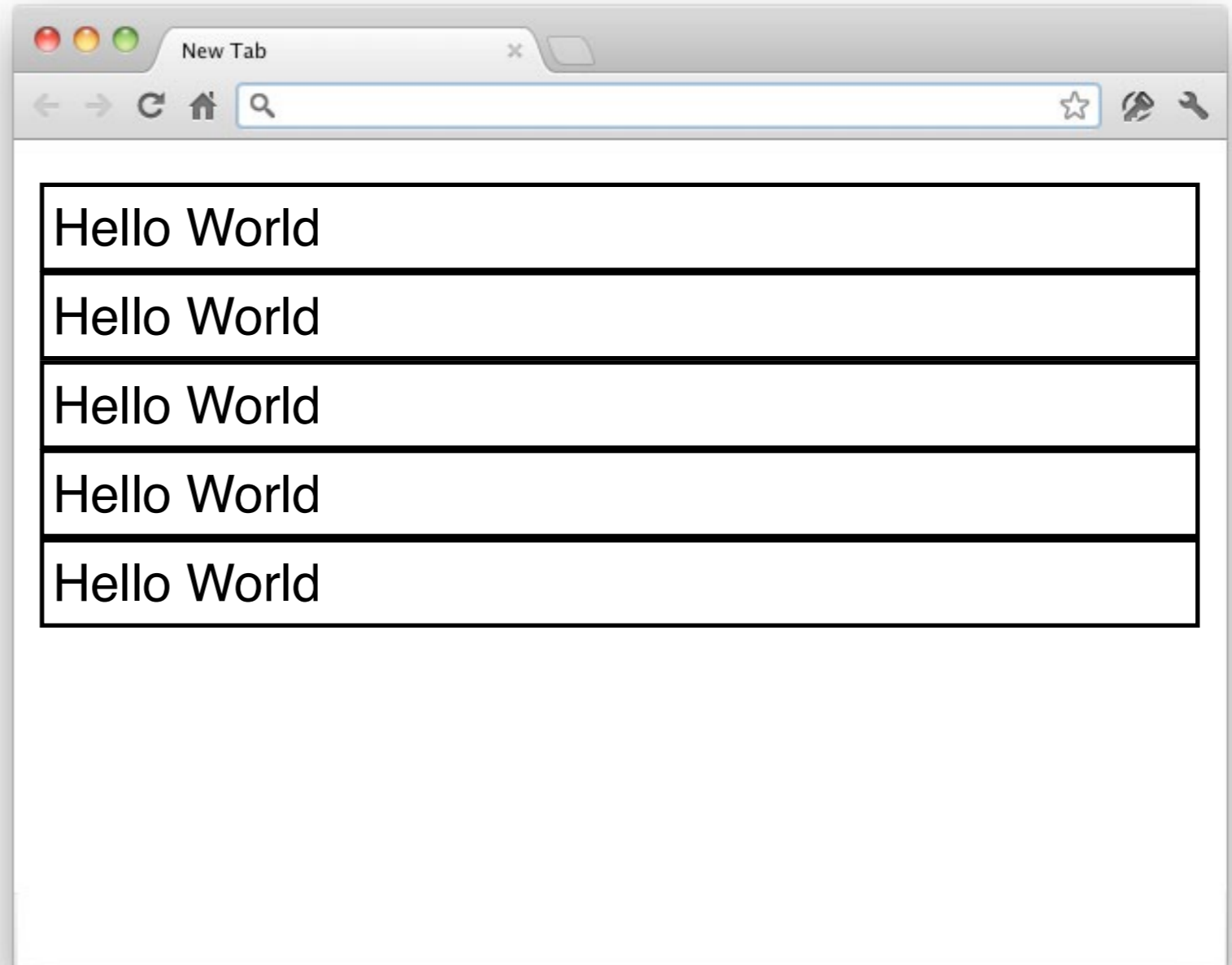
# Rules of Block Elements

## CSS

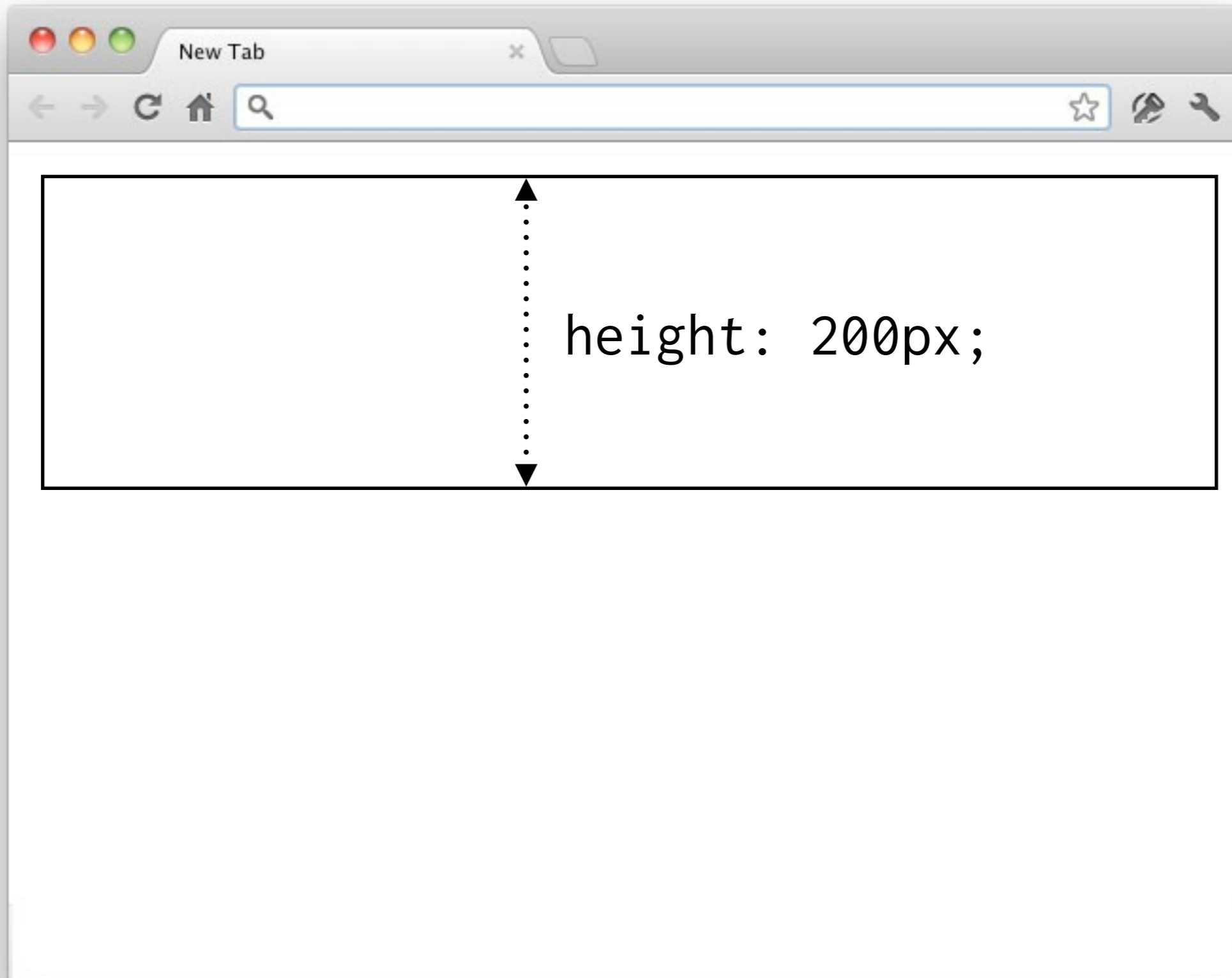
```
div {  
  border:1px solid black;  
  
}
```

## HTML

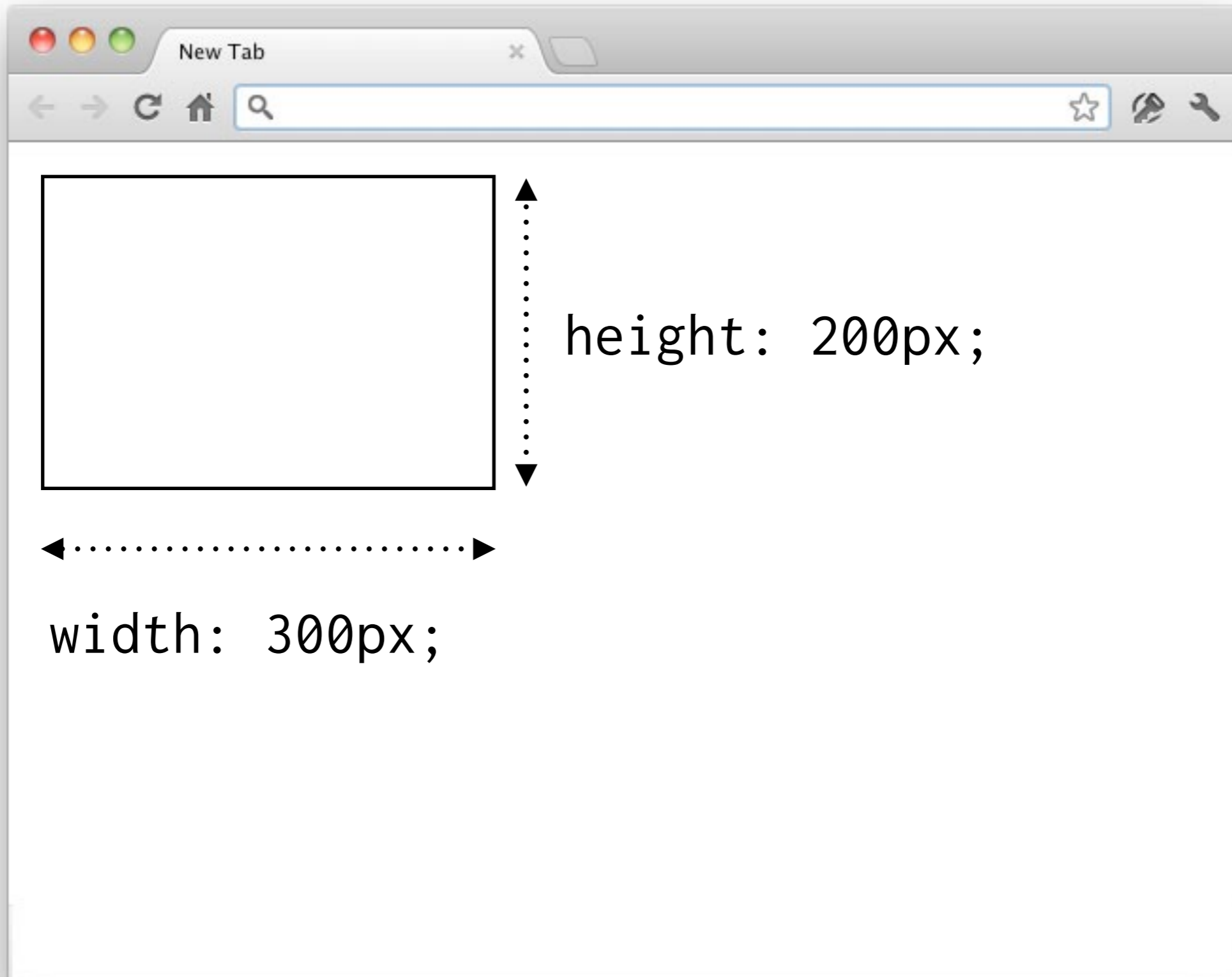
```
<div>Hello World!</div>  
<div>Hello World!</div>  
<div>Hello World!</div>  
<div>Hello World!</div>  
<div>Hello World!</div>
```



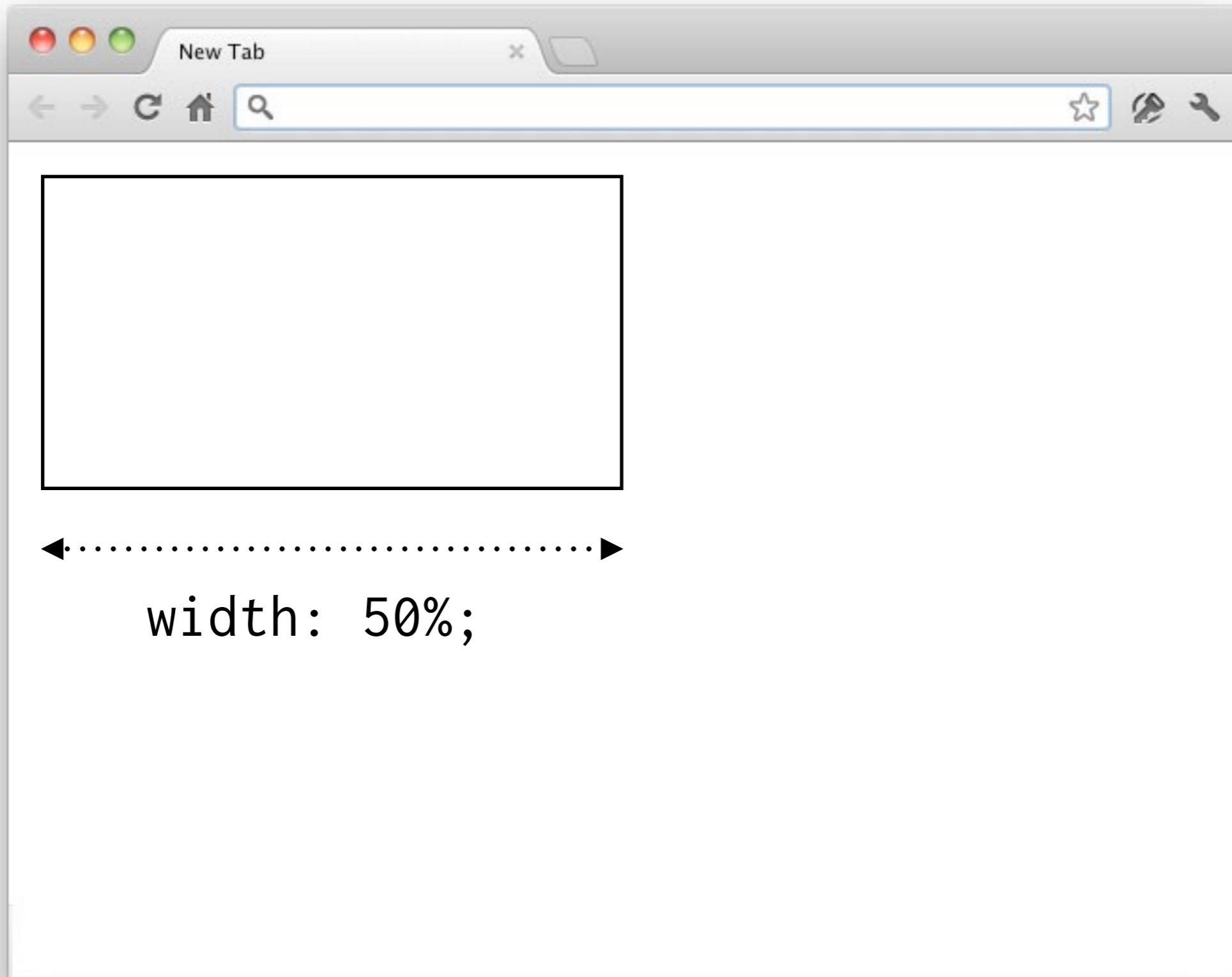
# Peculiarities of a <div> box



# Peculiarities of a <div> box

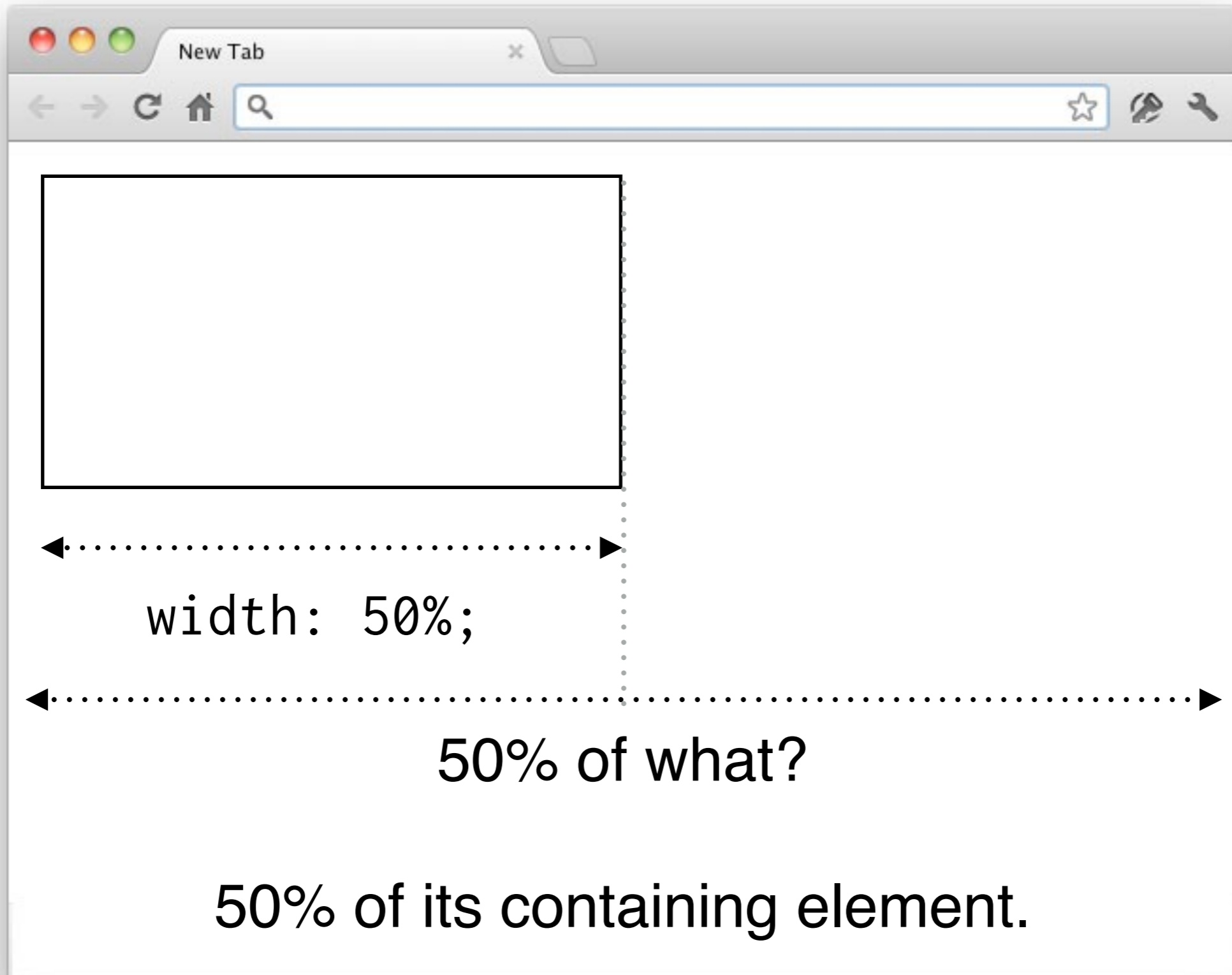


# Peculiarities of a <div> box

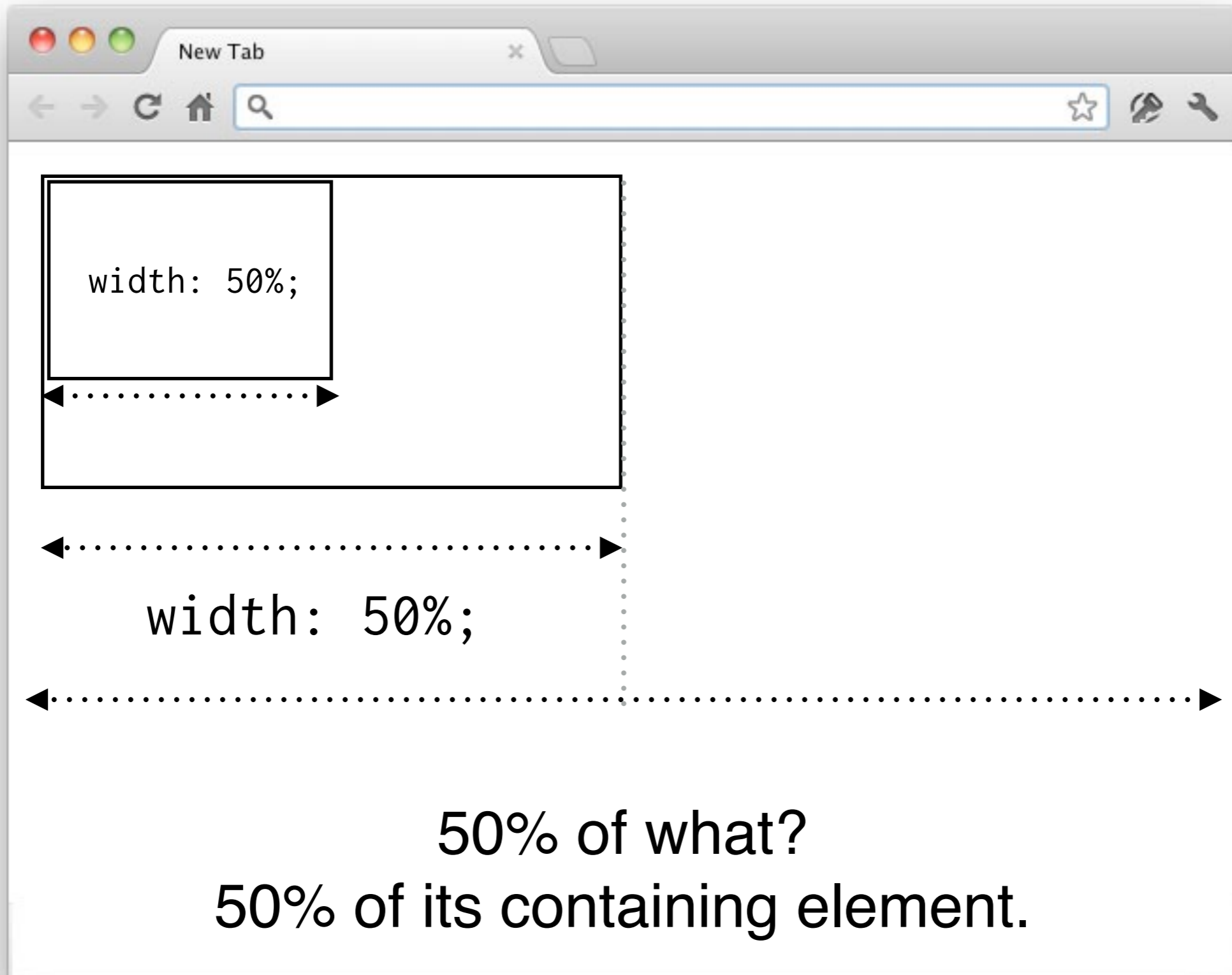




# Peculiarities of a <div> box



# Peculiarities of a <div> box



width: 50%;

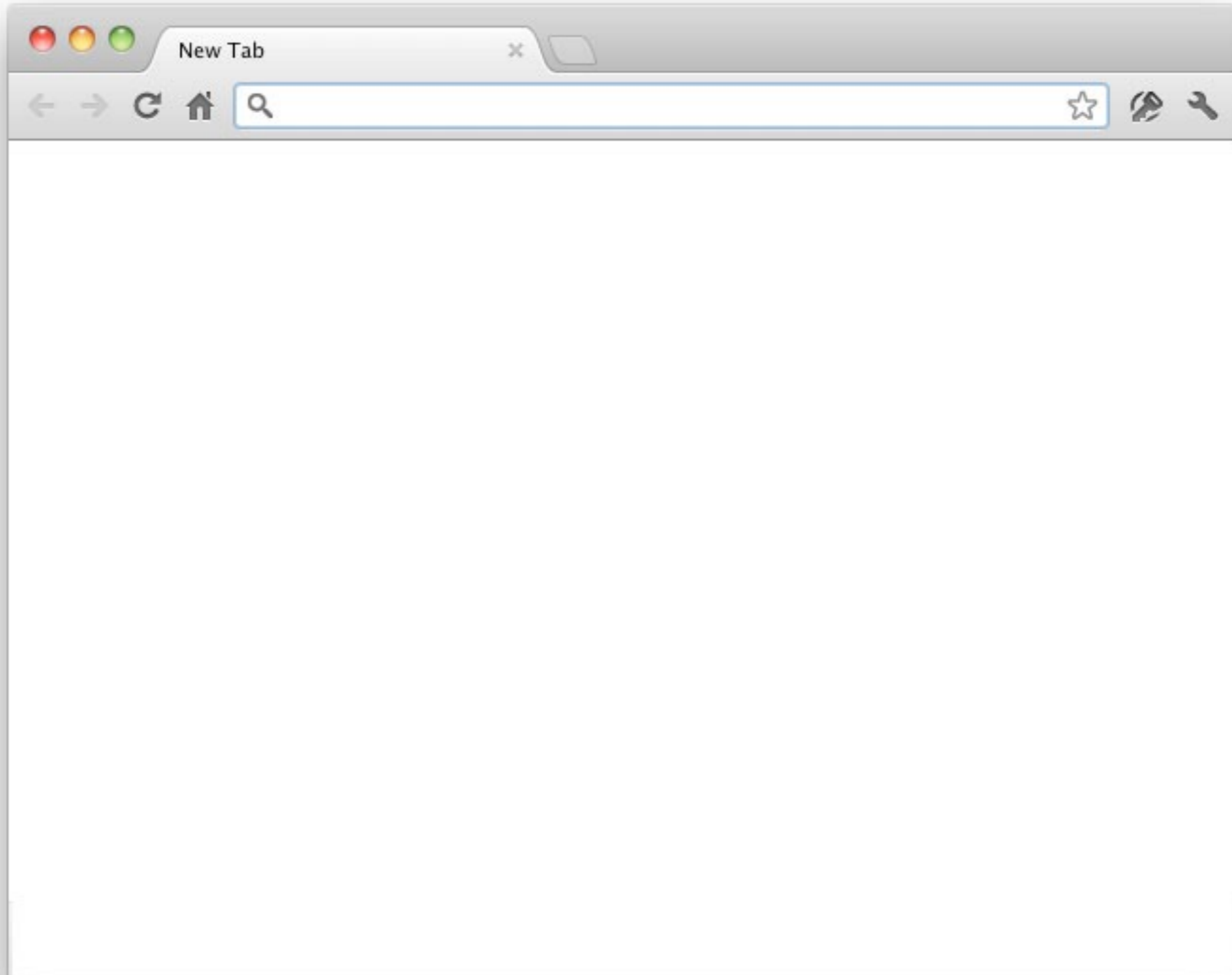
width: 50%;

50% of what?  
50% of its containing element.

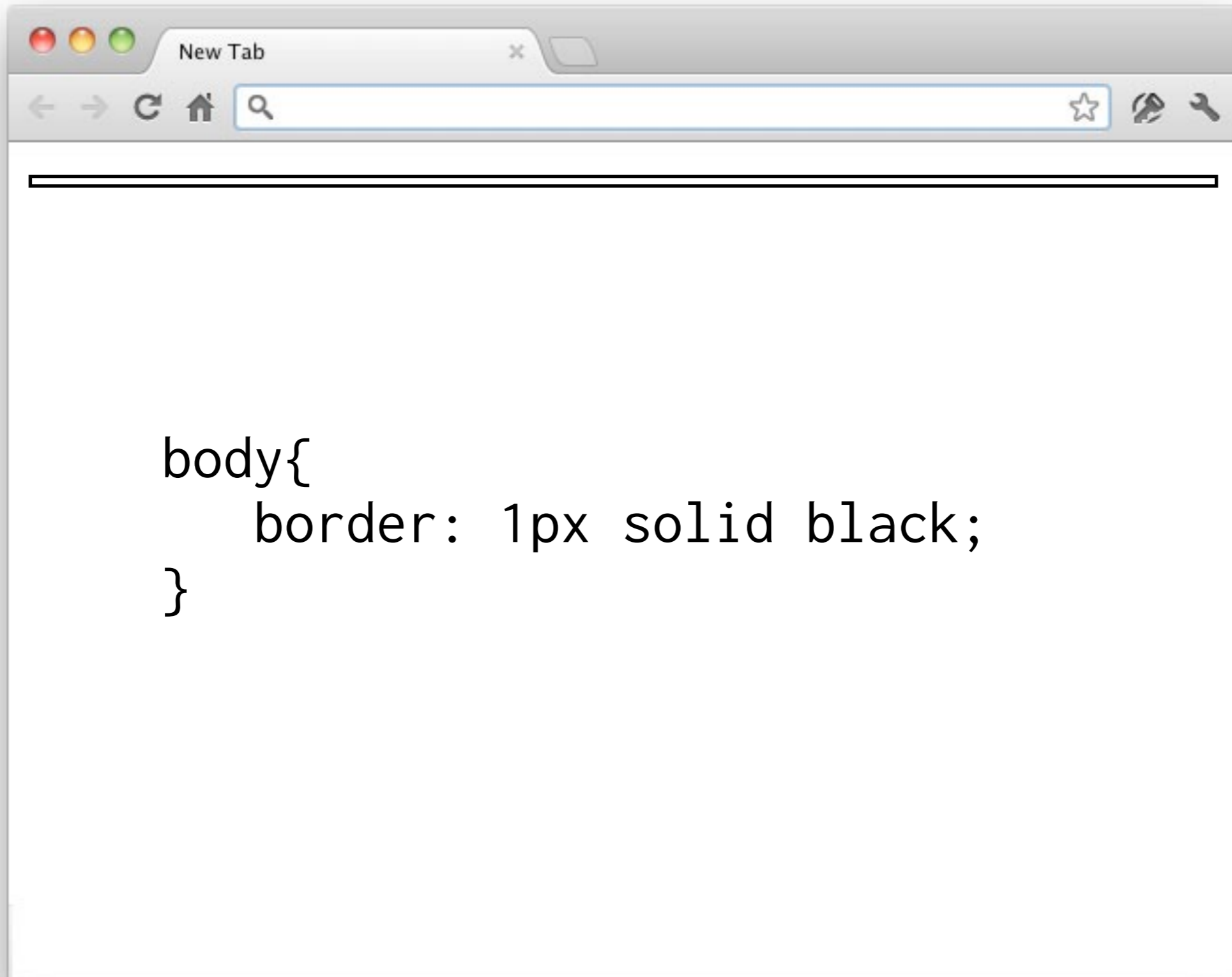
# Learning the quirks

The `<body>` tag is a box on your page.

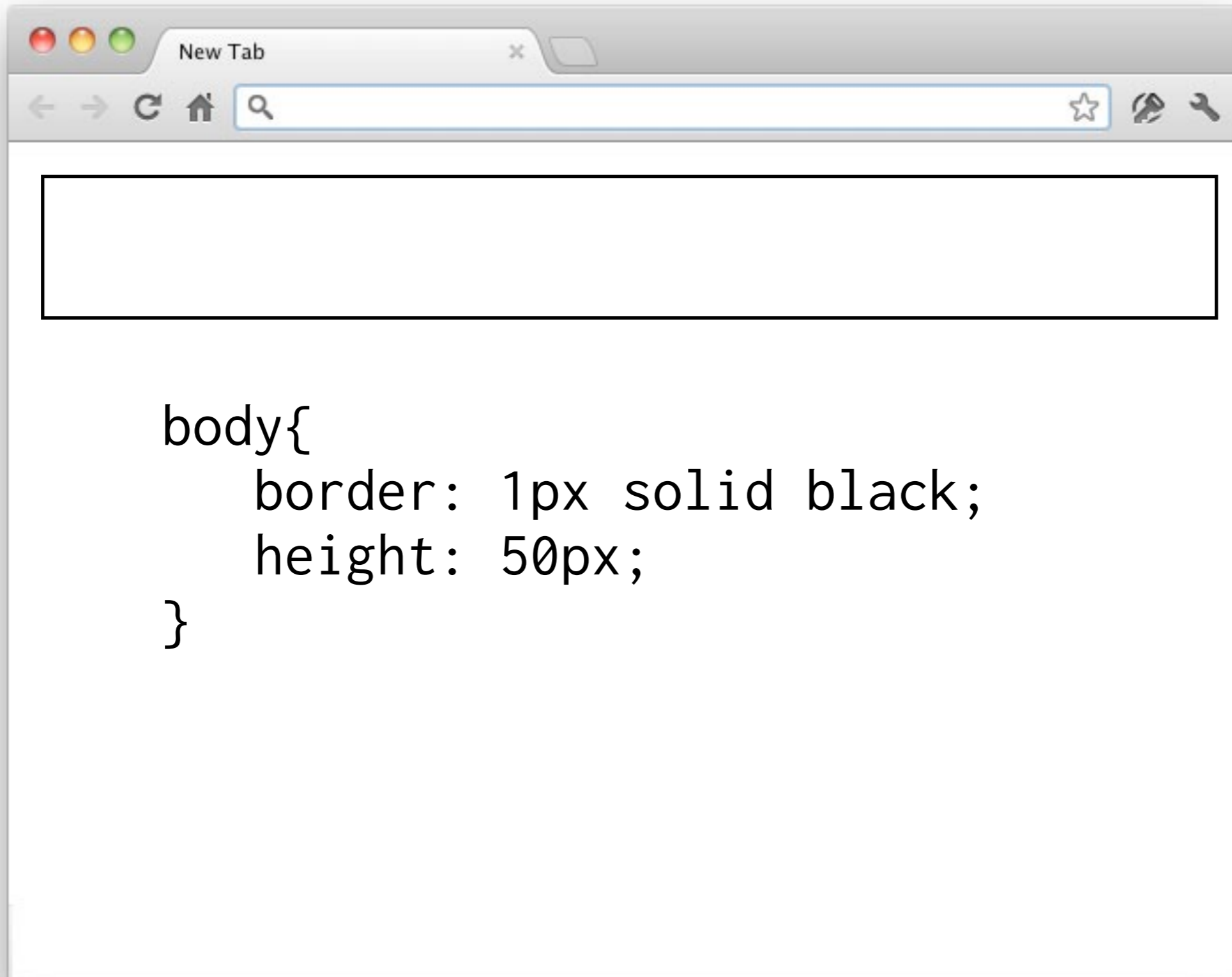
# The `<body>` tag is there



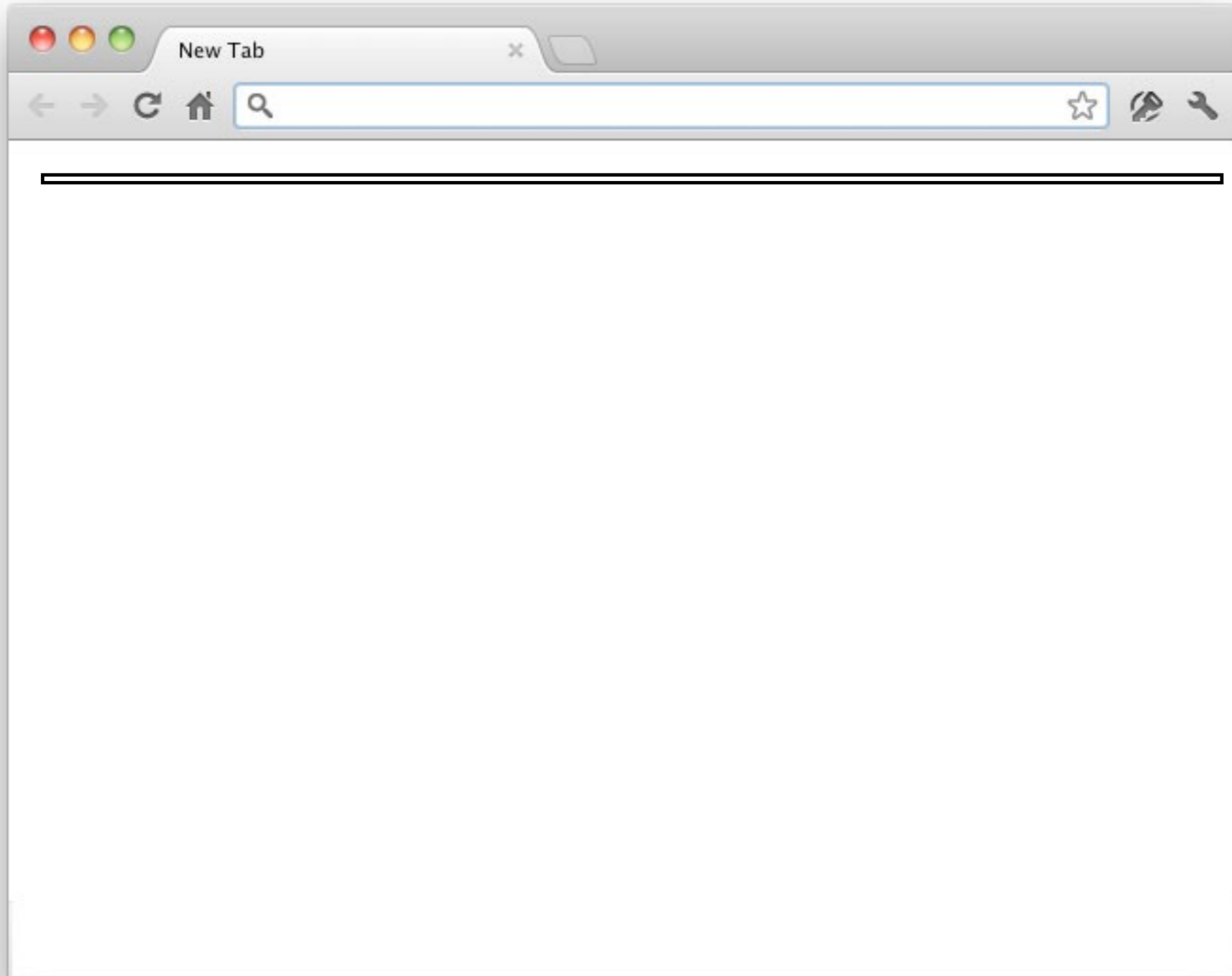
Style it with a border and you'll see it.



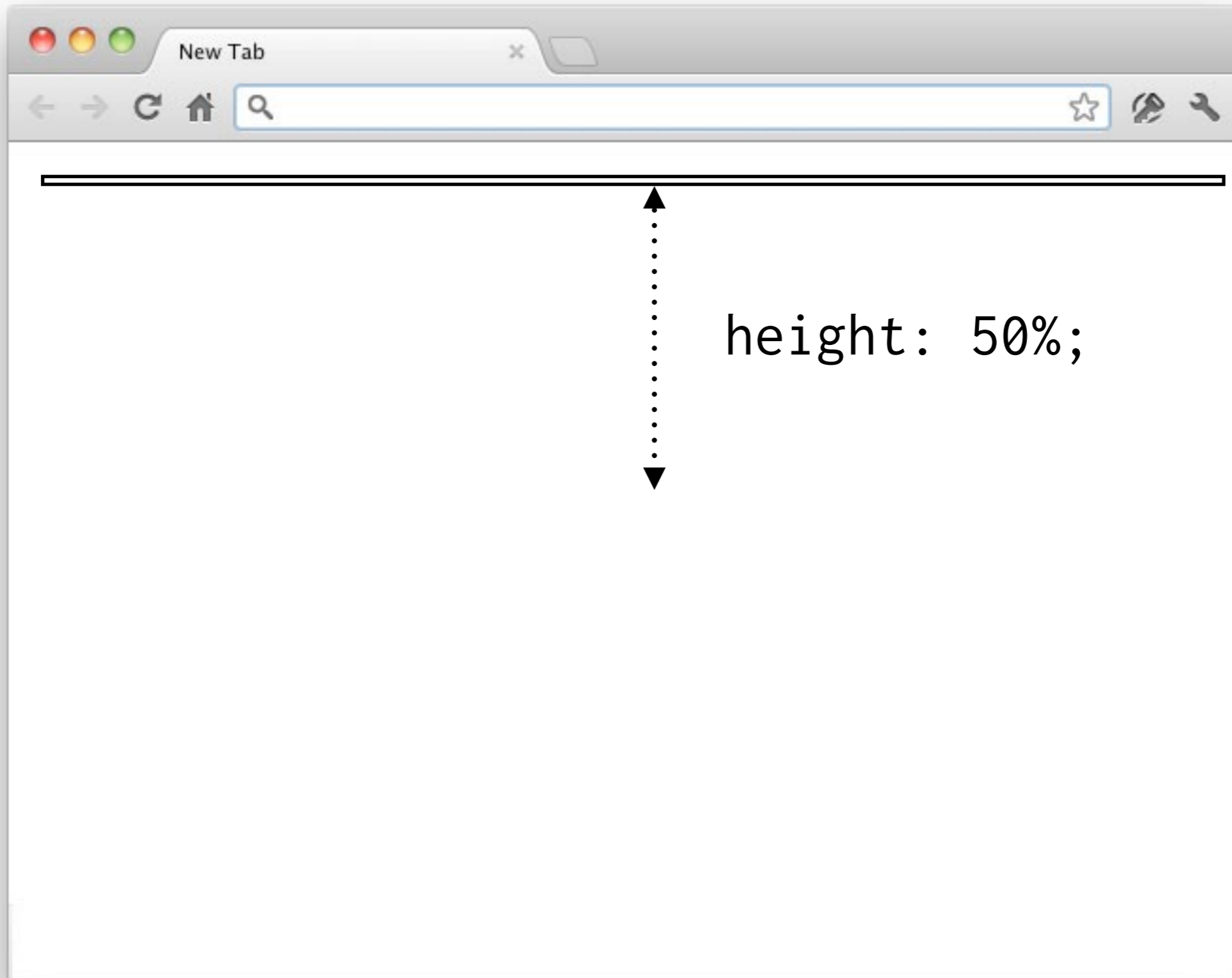
# You can give it a height.



Which means, if we have a `<div>` box

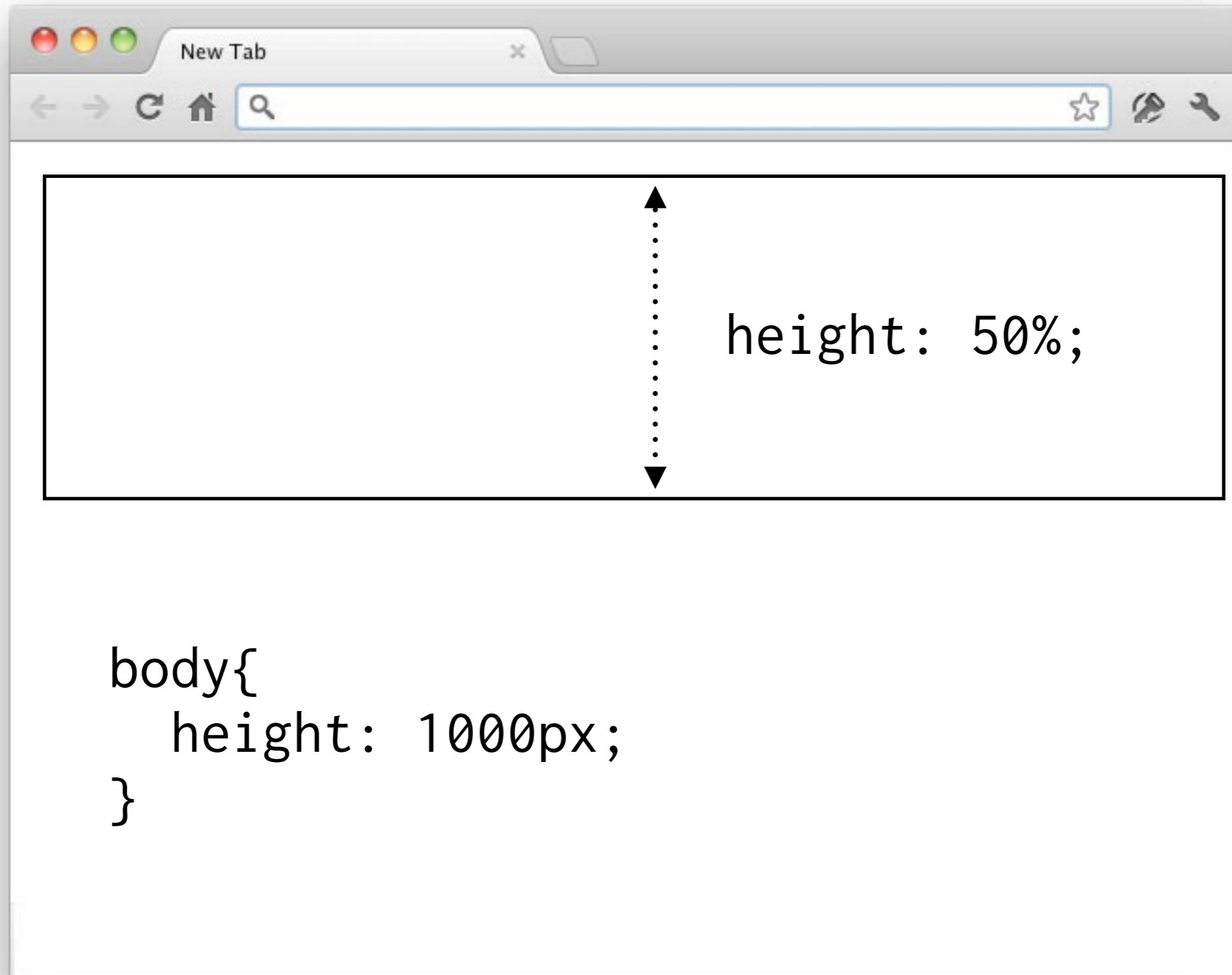


# 50% of what?

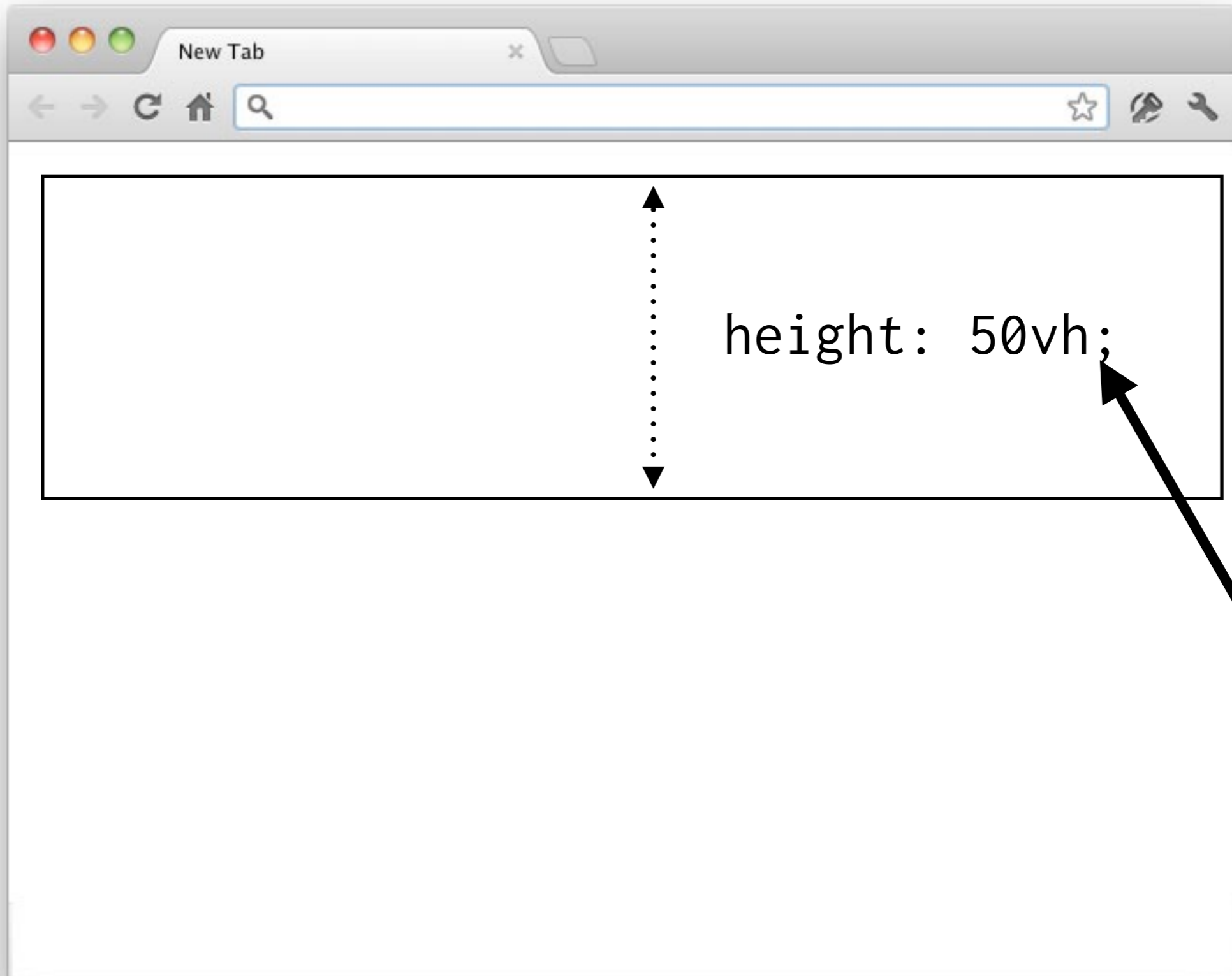




# Width and Height

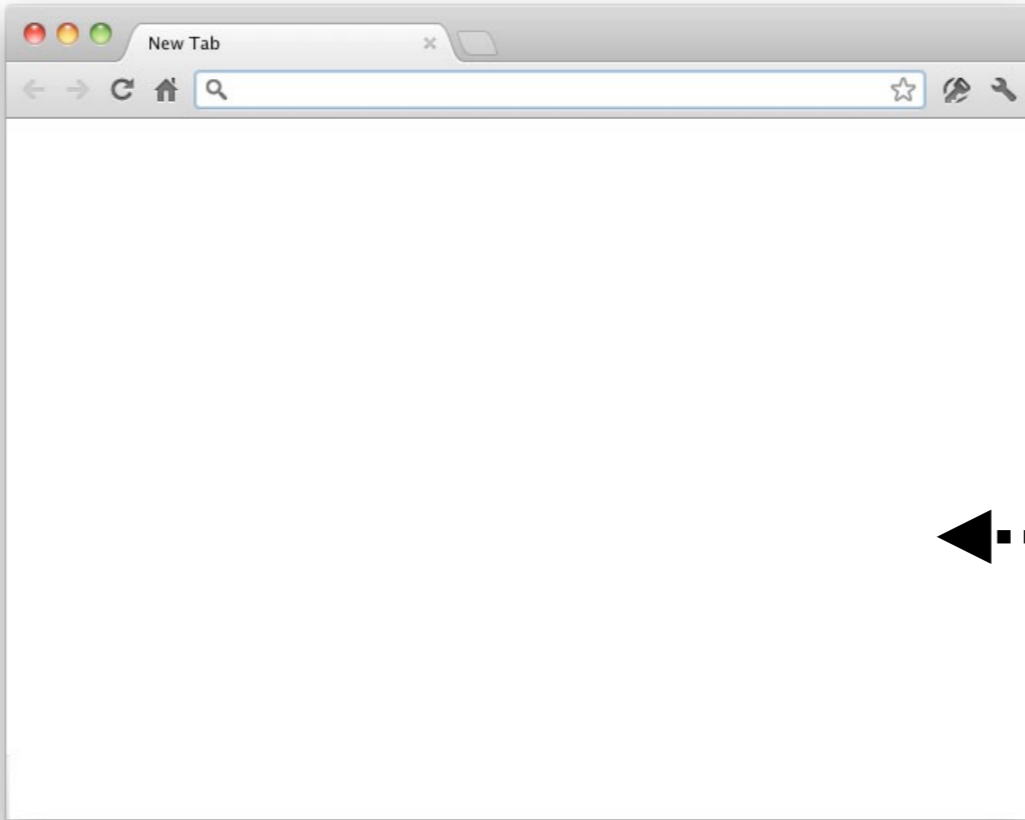


The viewport height (vh) measurement is based on the browser

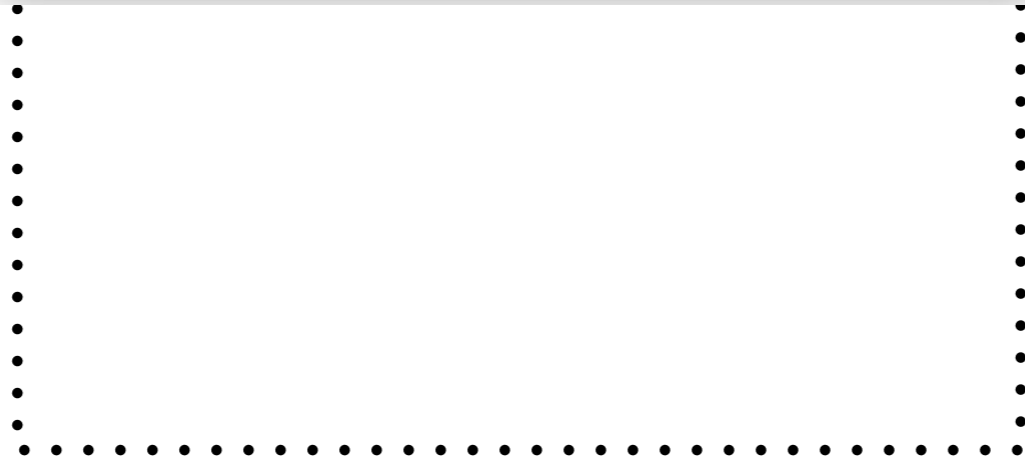


Your Page

web page  
or  
**document**



your browser  
or  
**viewport**

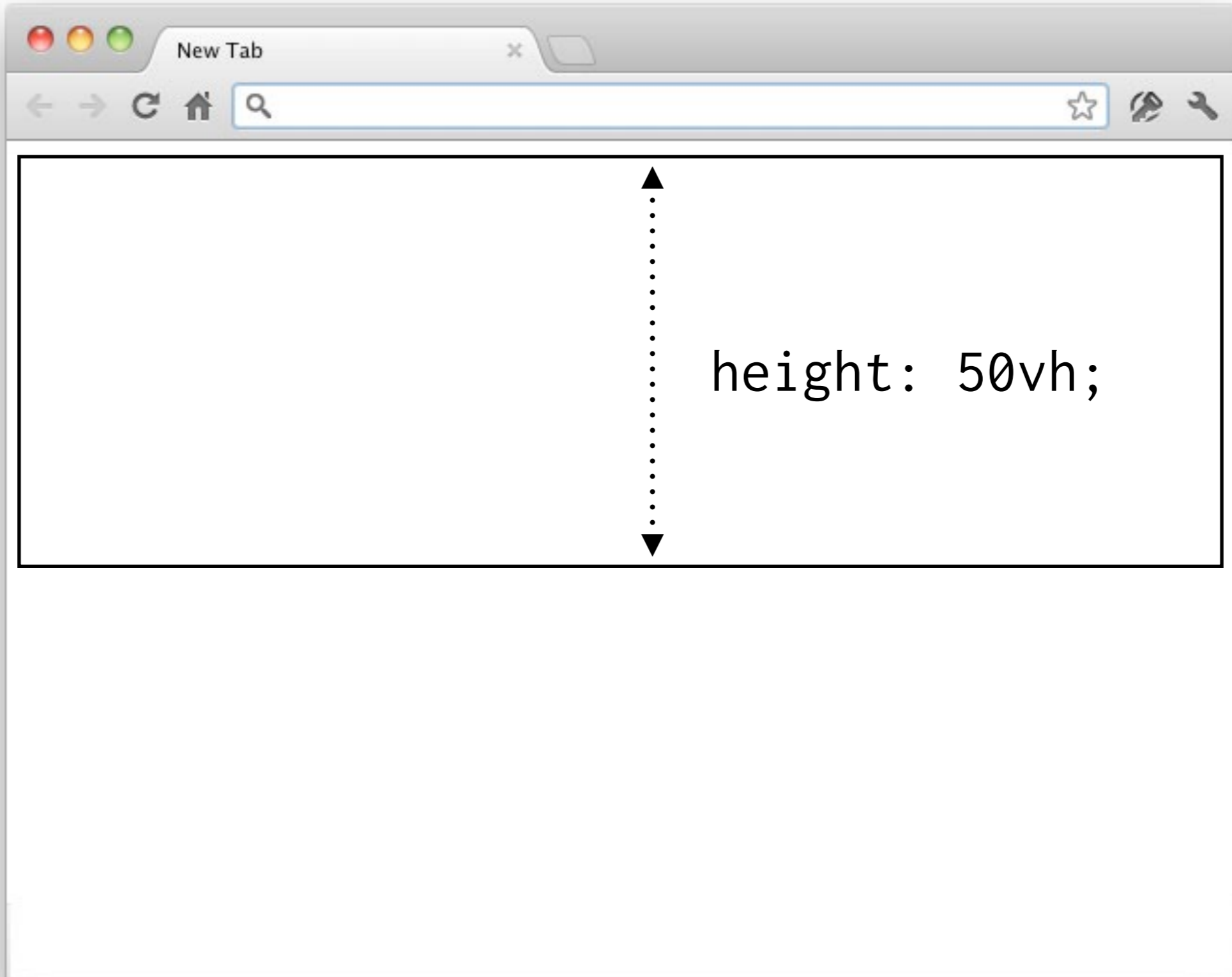


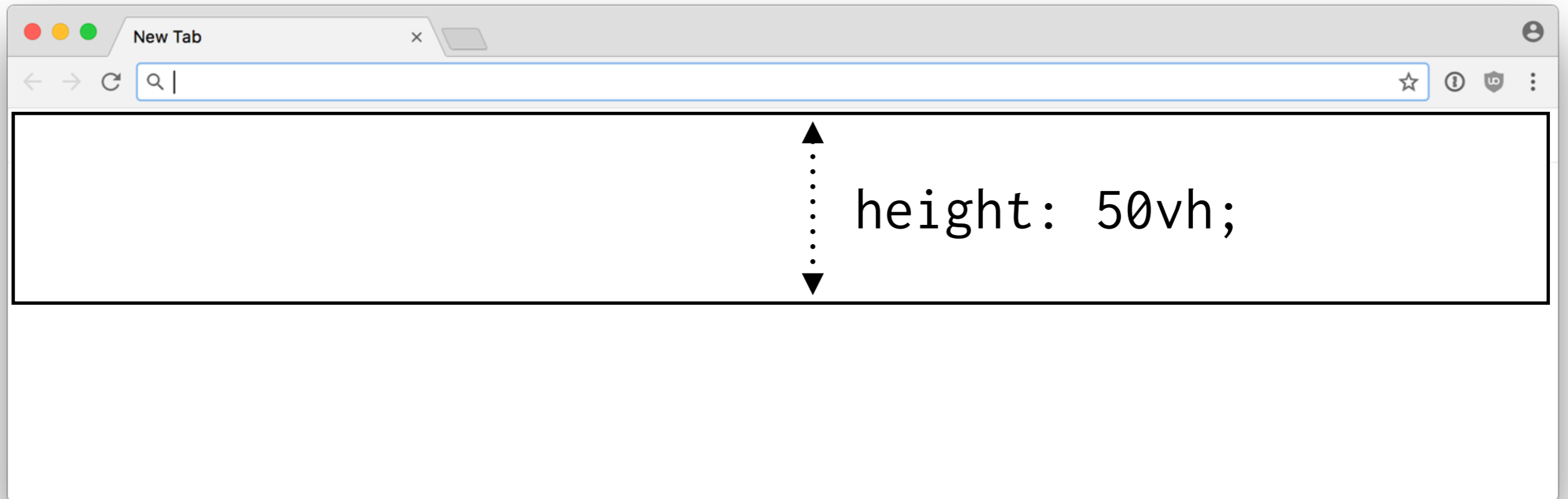
# Viewport Measurements

**80vw** = 80% of the viewport width

**100vh** = 100% of the viewport height

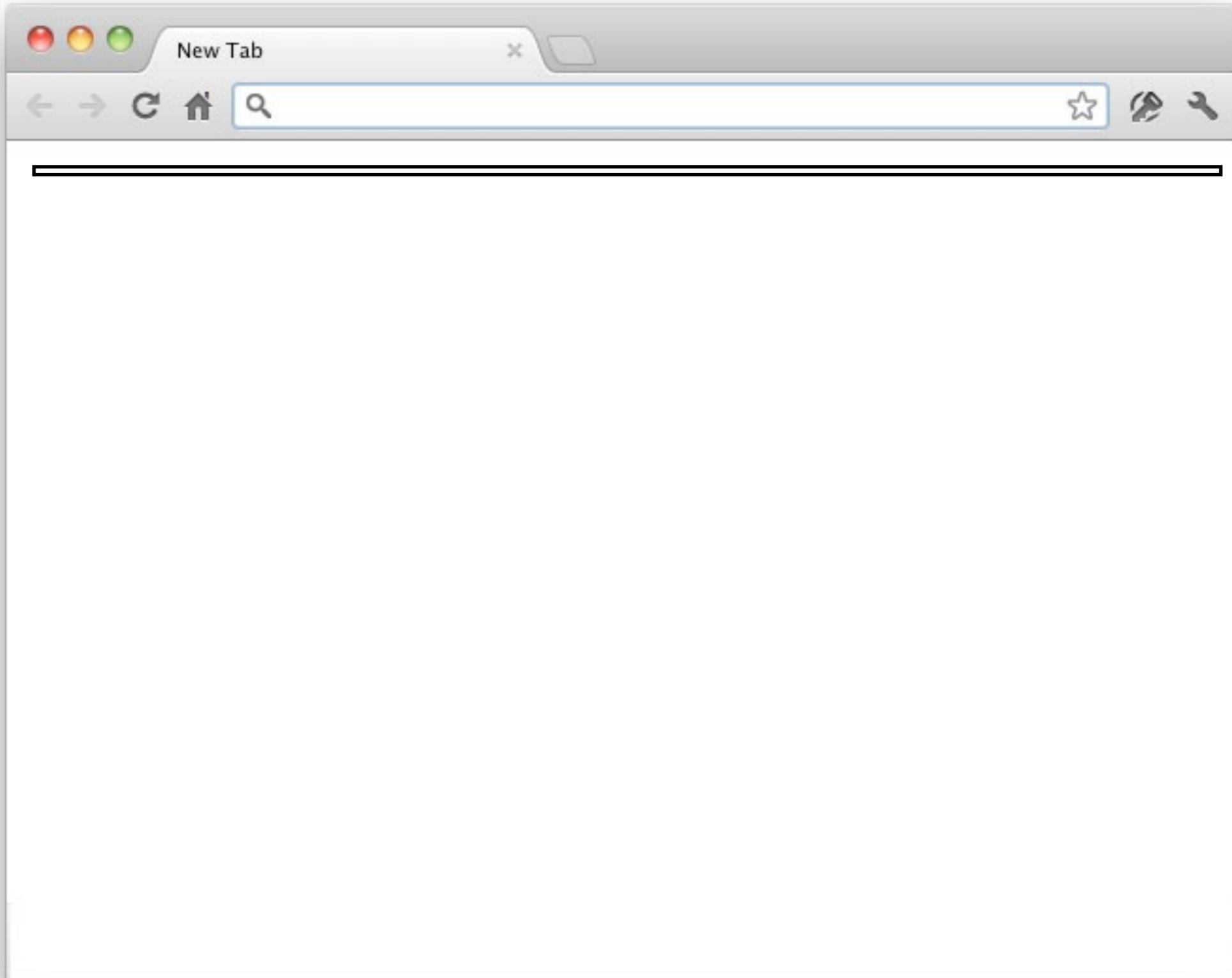
**5vh** = 5% of the viewport height





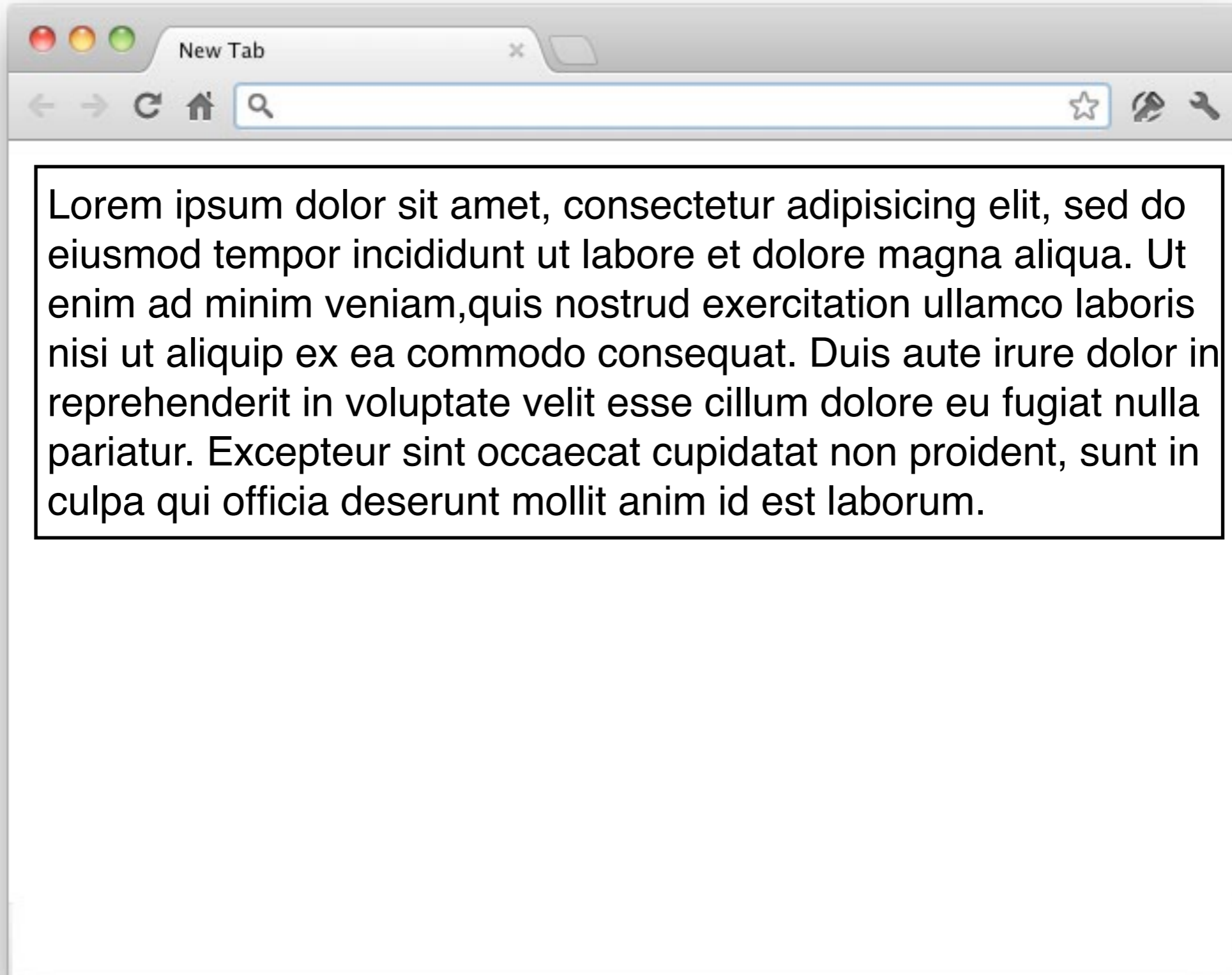
**overflow**

# overflow

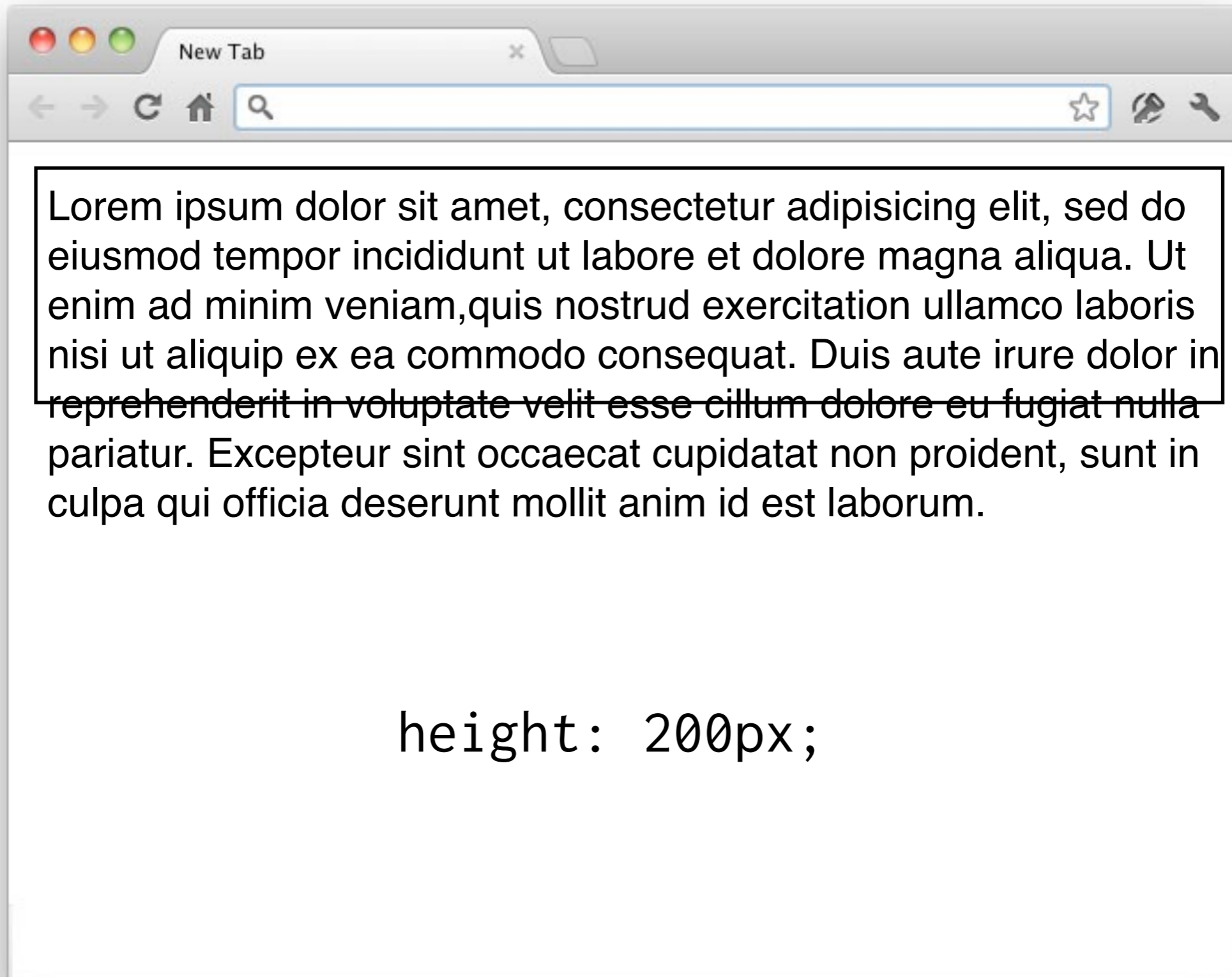




# overflow

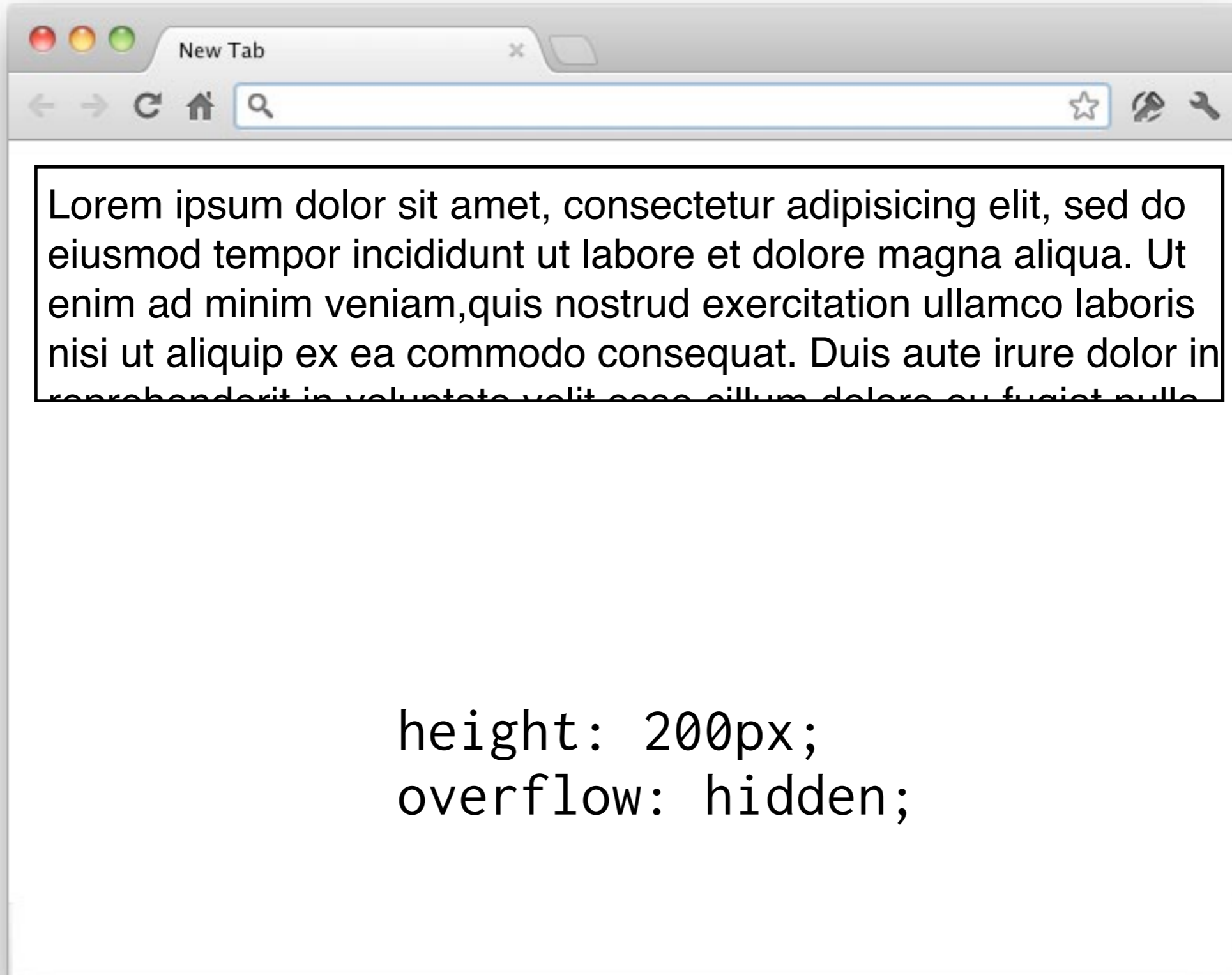


# overflow



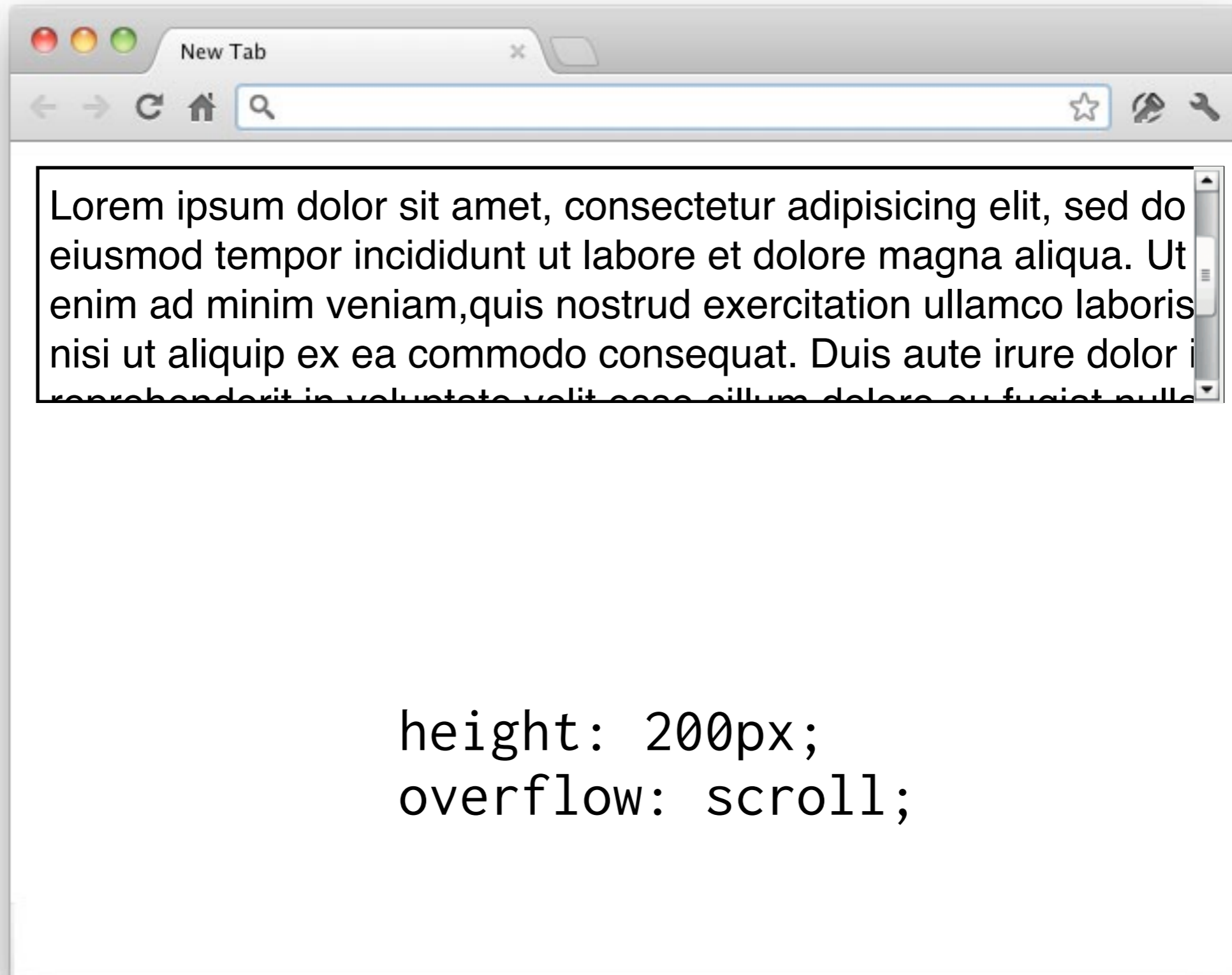
```
height: 200px;
```

# overflow



```
height: 200px;  
overflow: hidden;
```

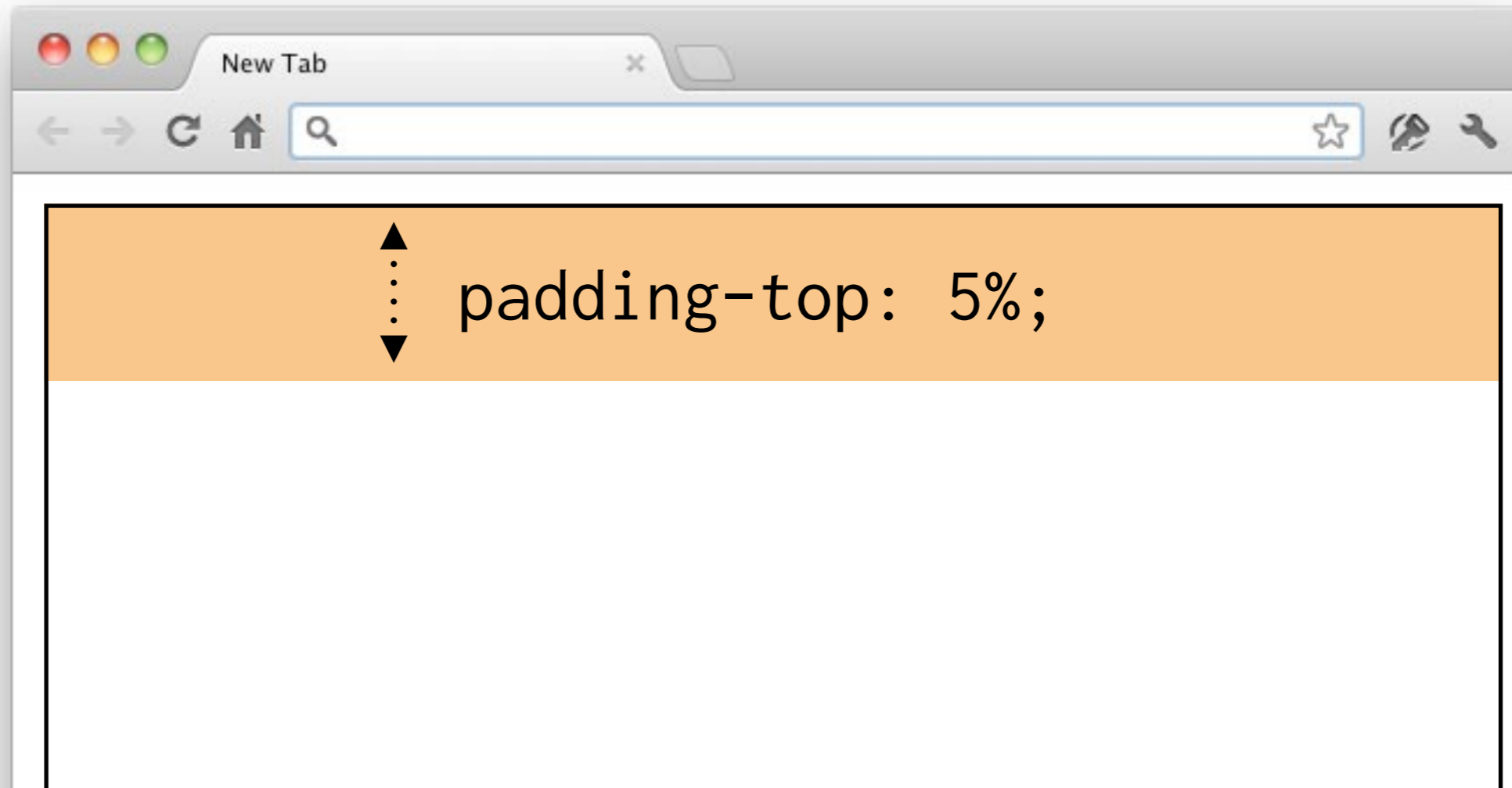
# overflow



```
height: 200px;  
overflow: scroll;
```

# Padding Margin

# Padding/Margin

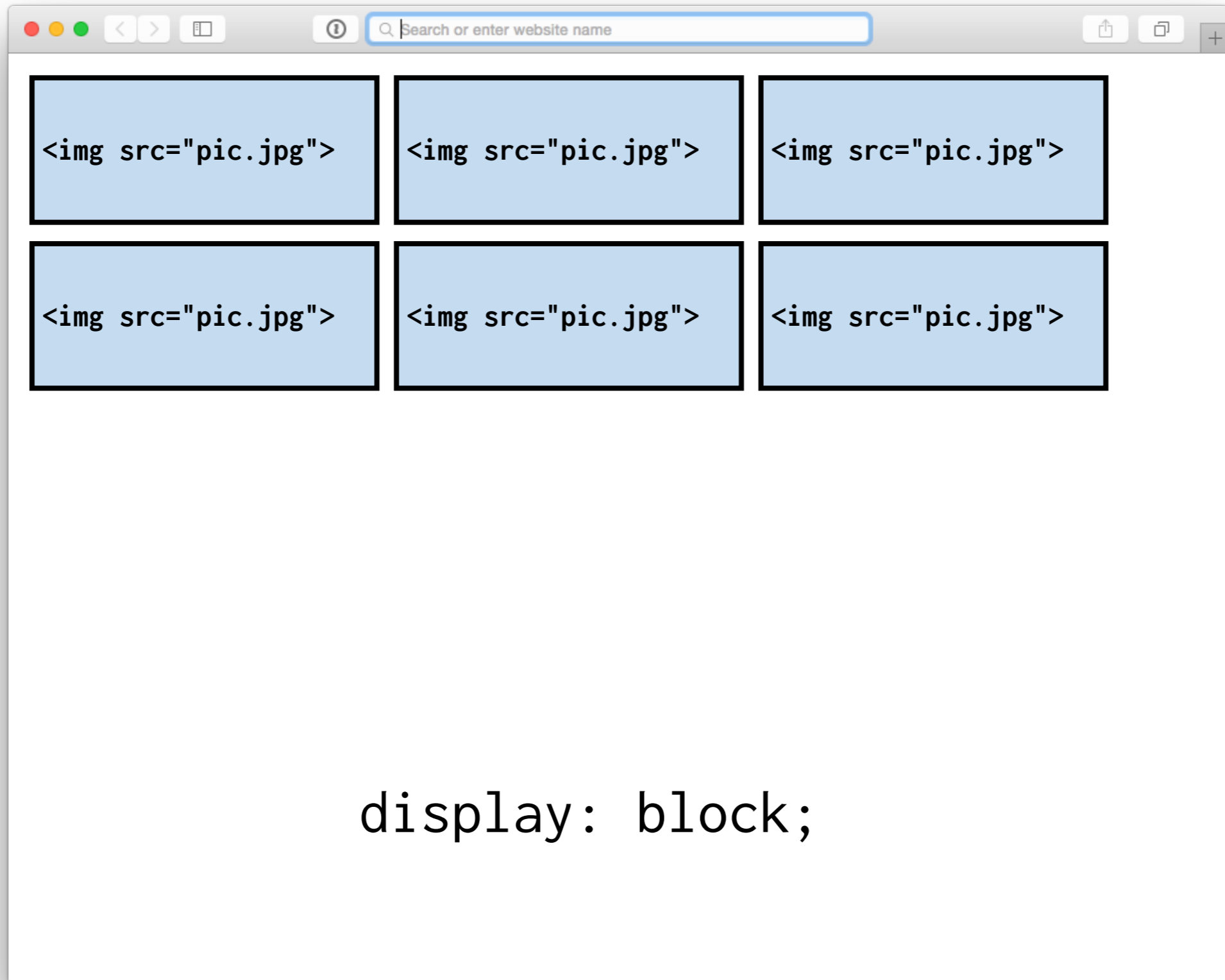


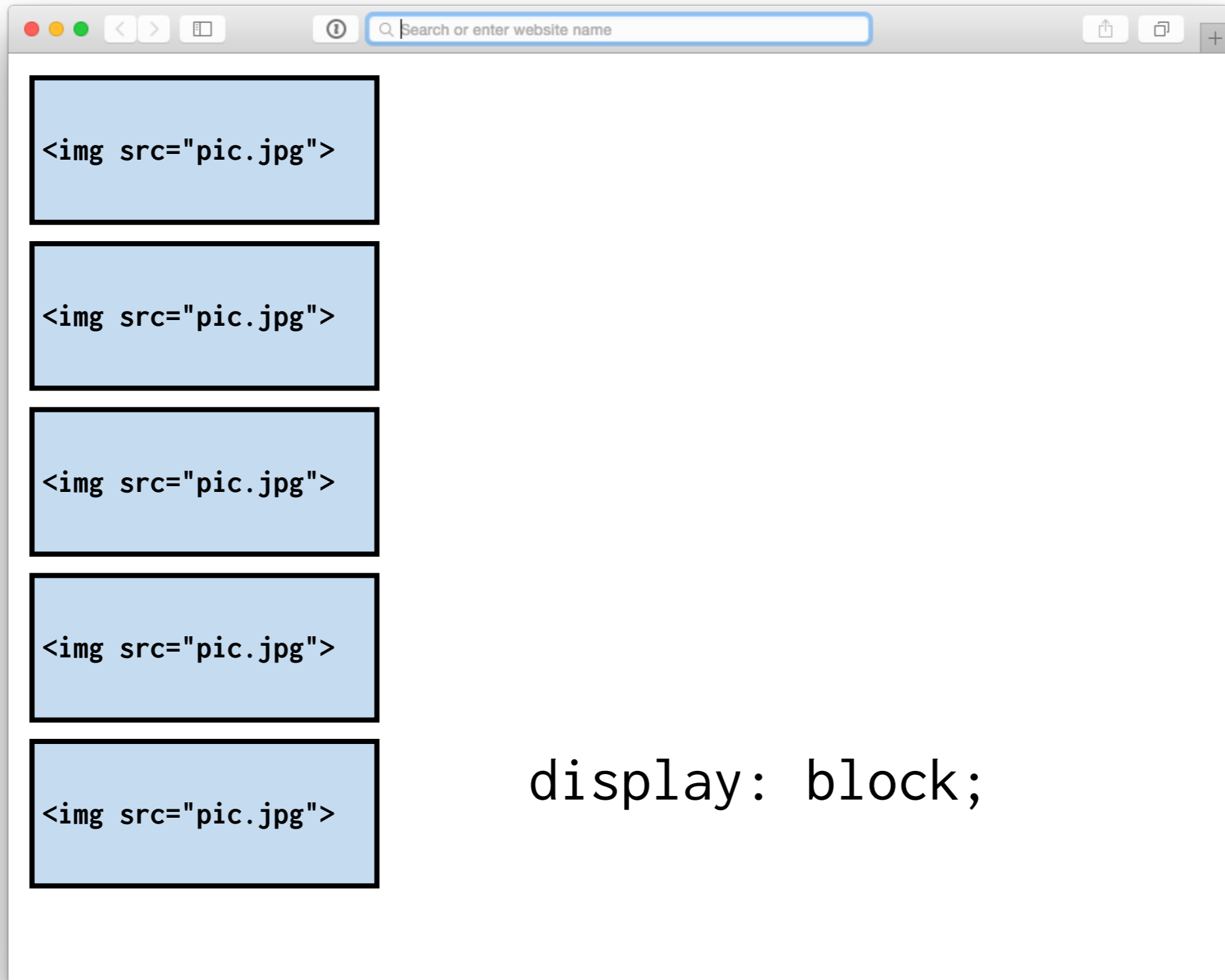
Padding top or bottom as a percentage will use the width of the container/ browser as a reference point.

**Display Property**









```

```

```

```

```

```

```

```

```

```

```
display: block;
```

# Hiding Elements

# Hiding Elements

`display: none;`

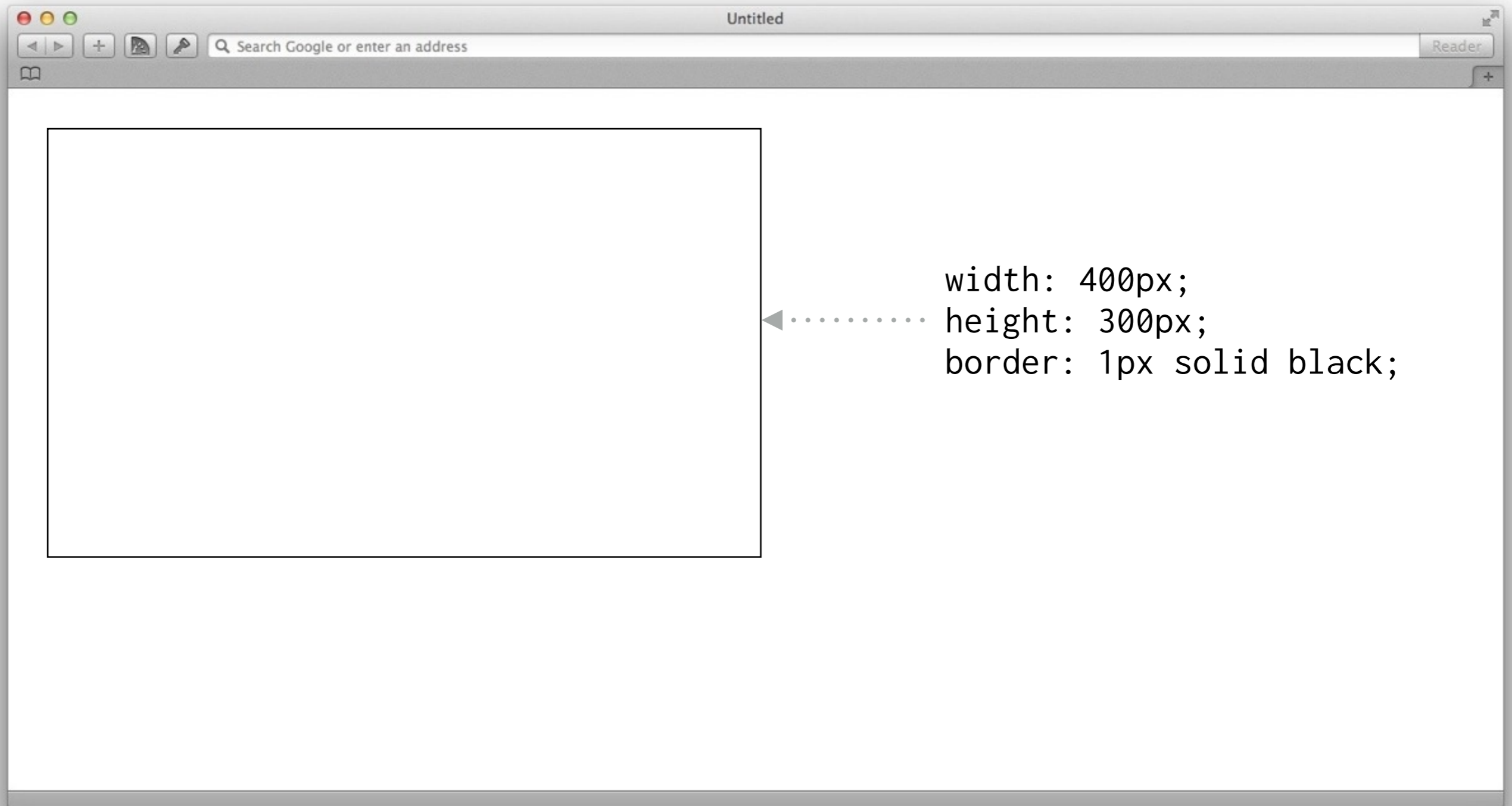
Remove completely from the document flow, as if it never existed.

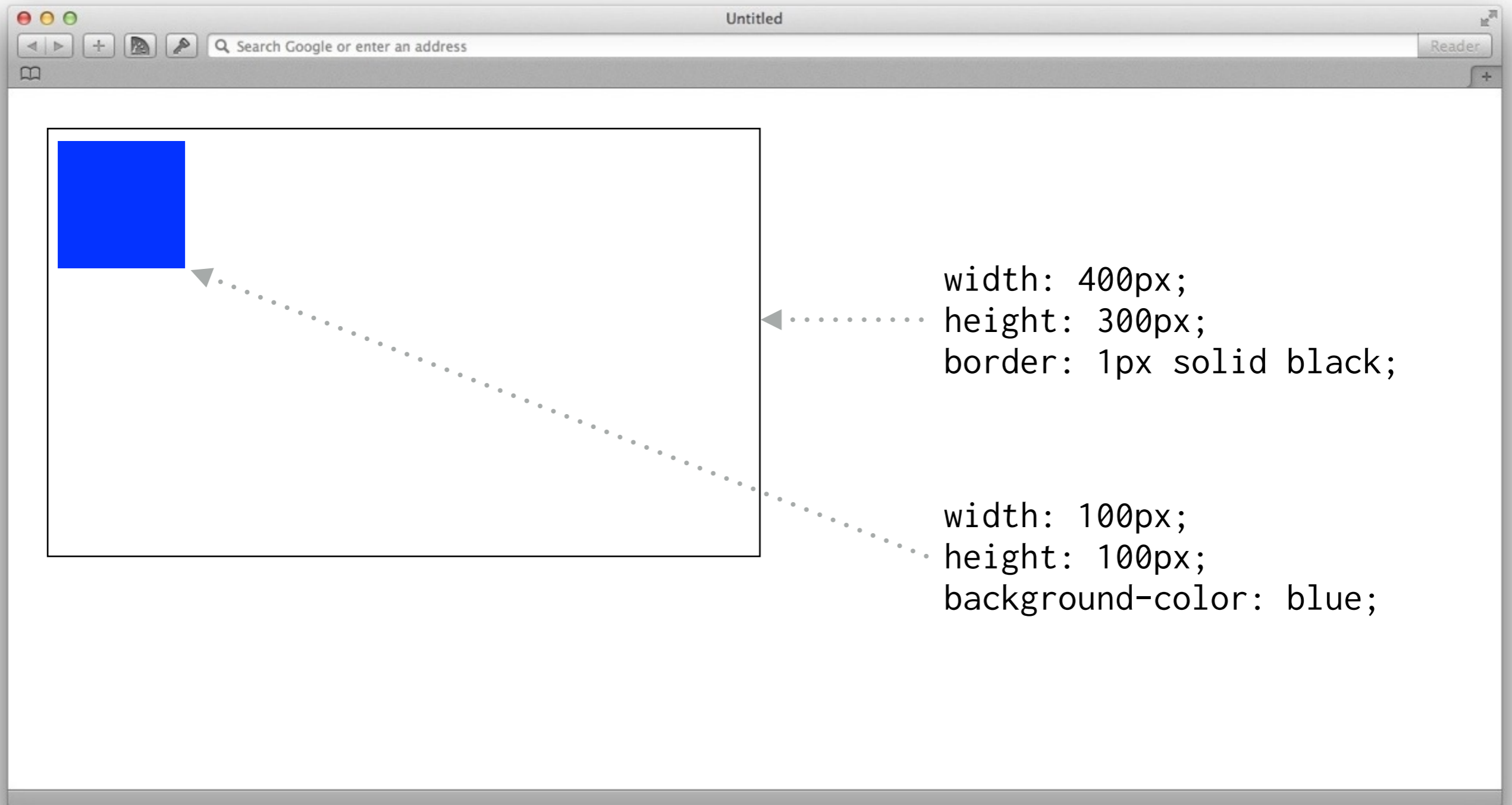
`visibility: hidden;`

Make it hidden, but it's still there and affects elements around it.

# Positioning

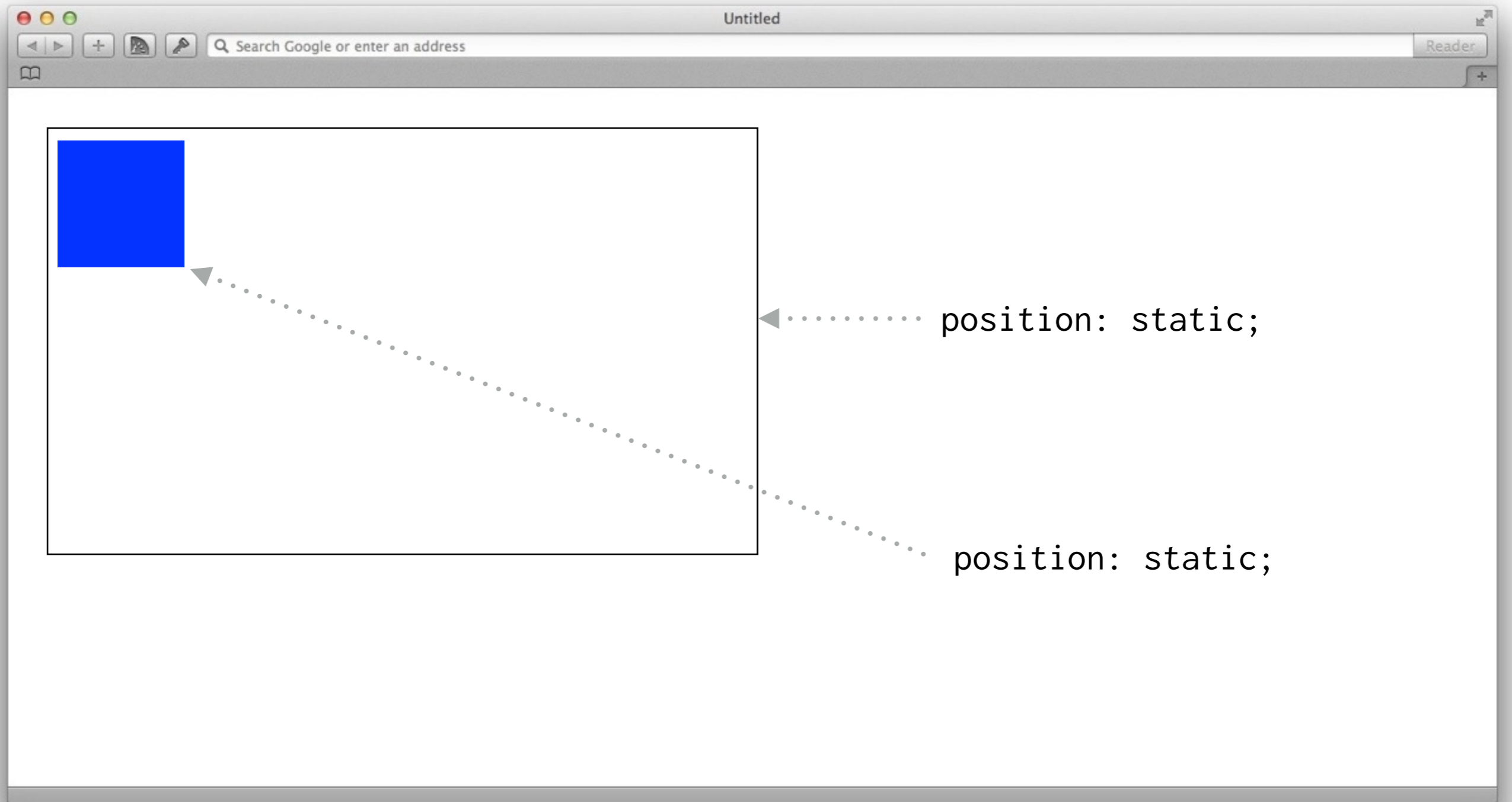
- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.





```
width: 400px;  
height: 300px;  
border: 1px solid black;
```

```
width: 100px;  
height: 100px;  
background-color: blue;
```



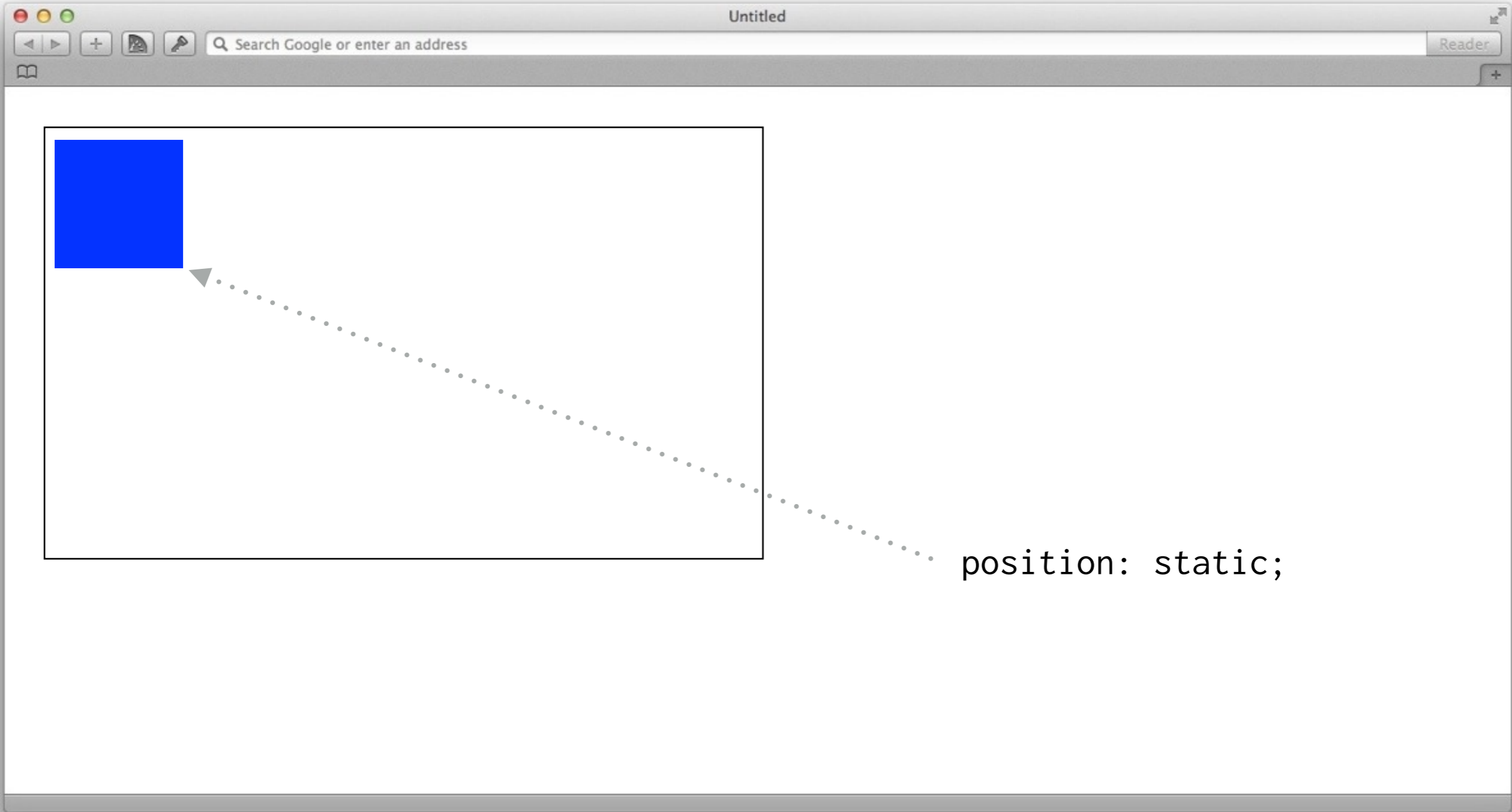


# Positioning

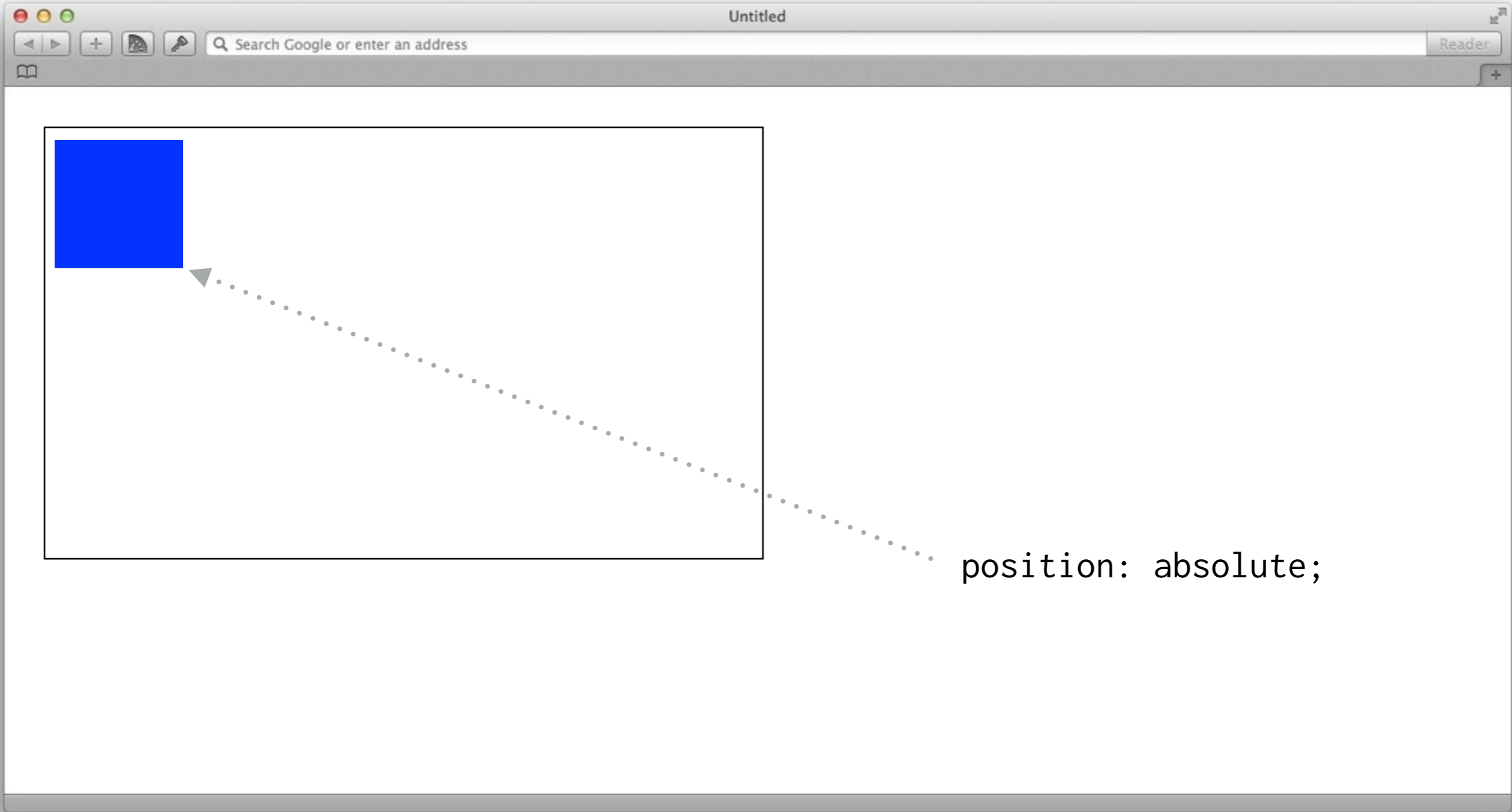
- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.

# Positioning

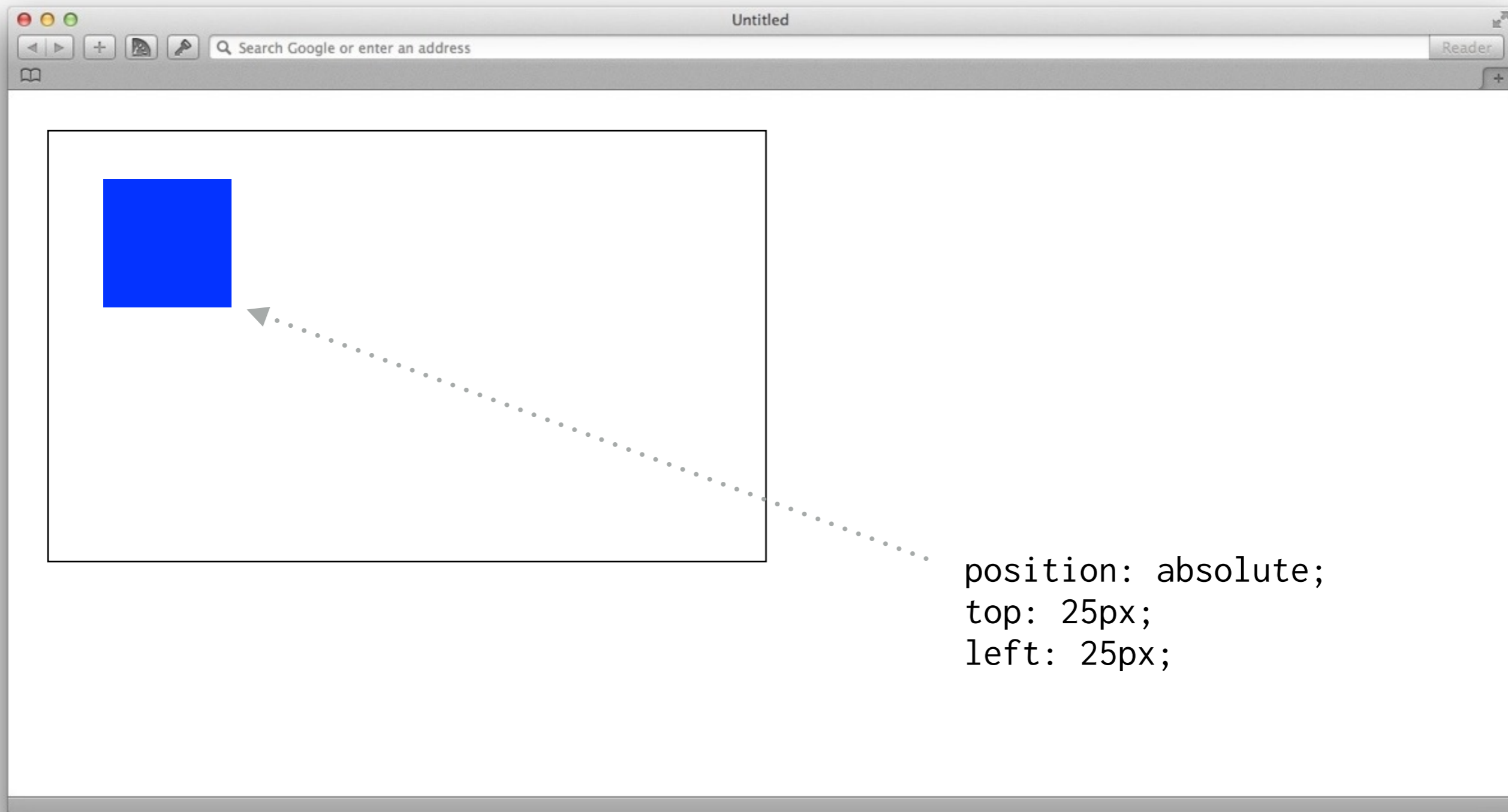
- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.

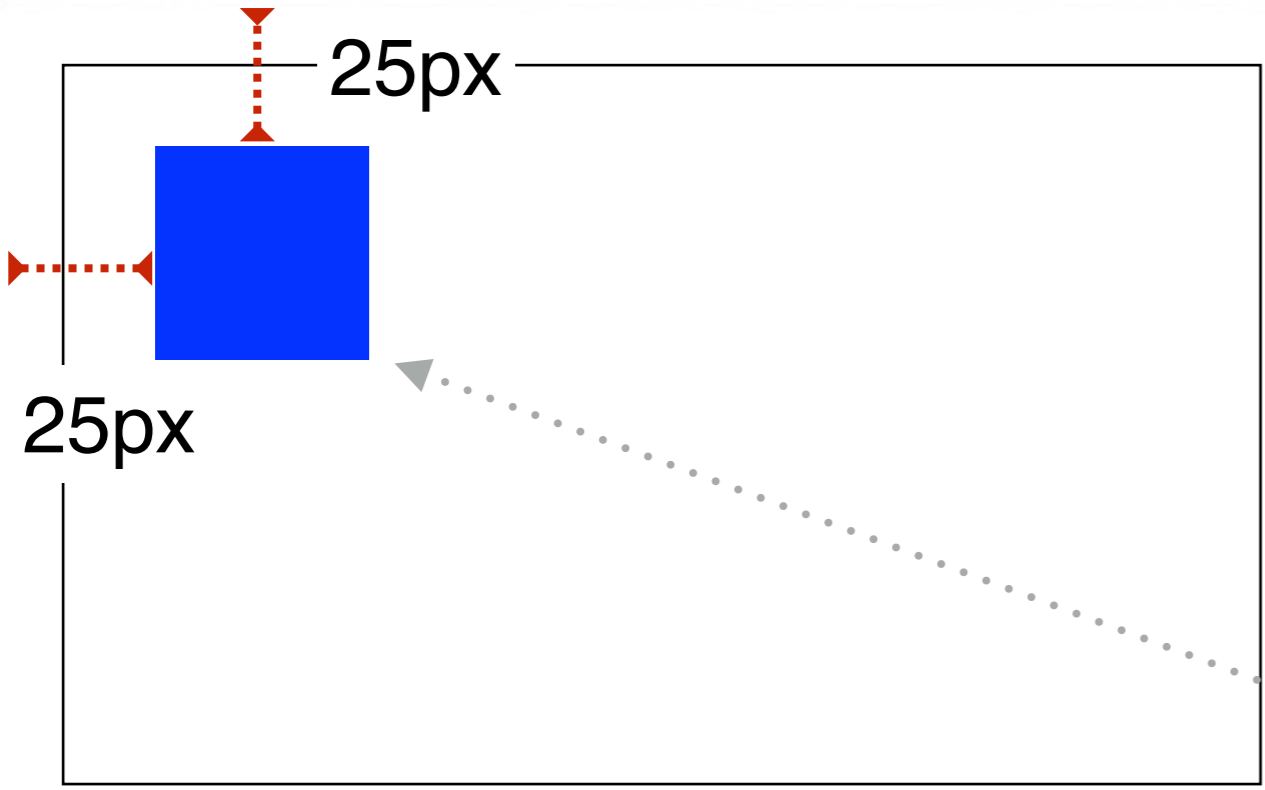


position: static;



position: absolute;





25px

25px

```
position: absolute;  
top: 25px;  
left: 25px;
```

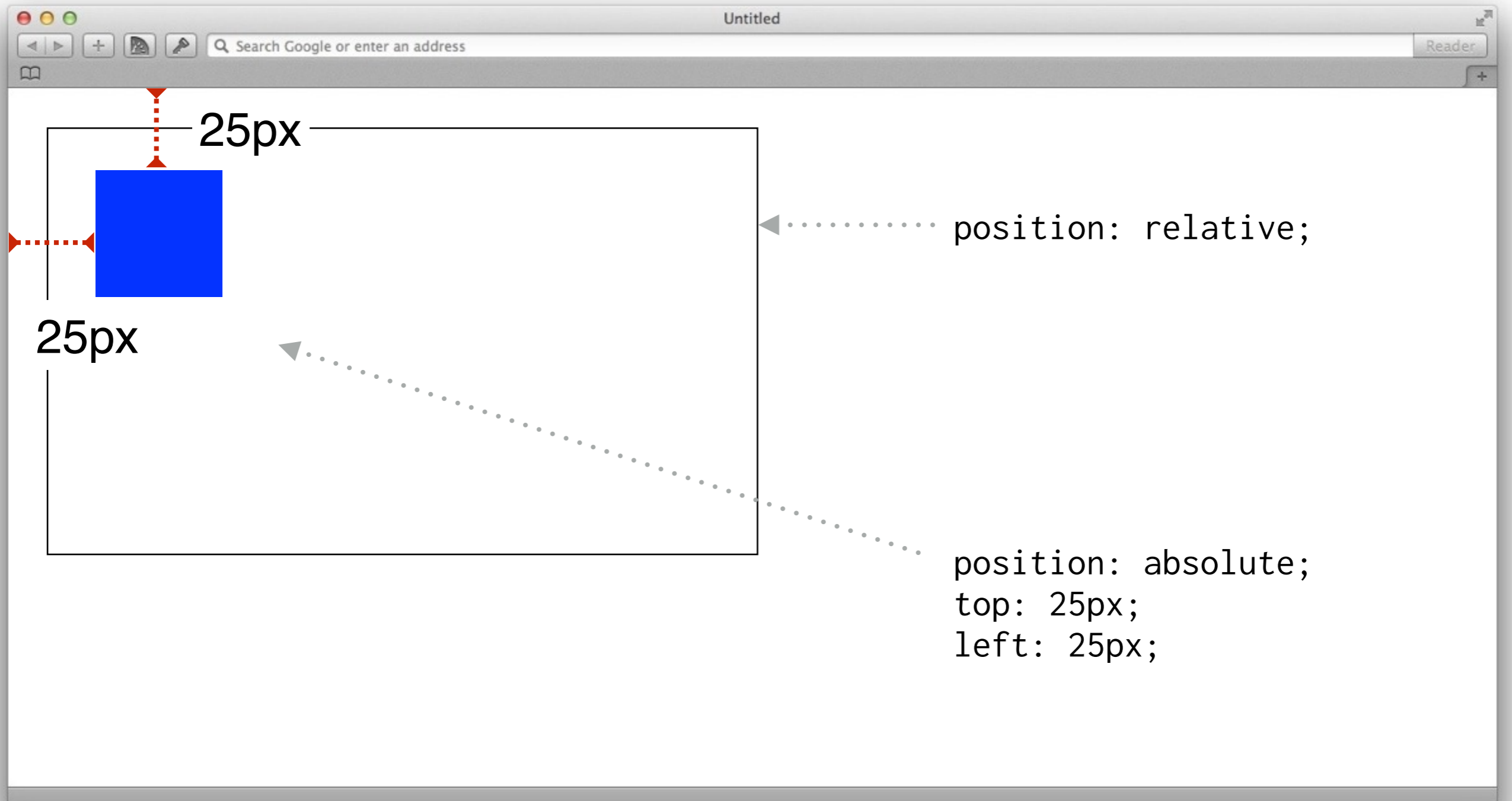
# Positioning

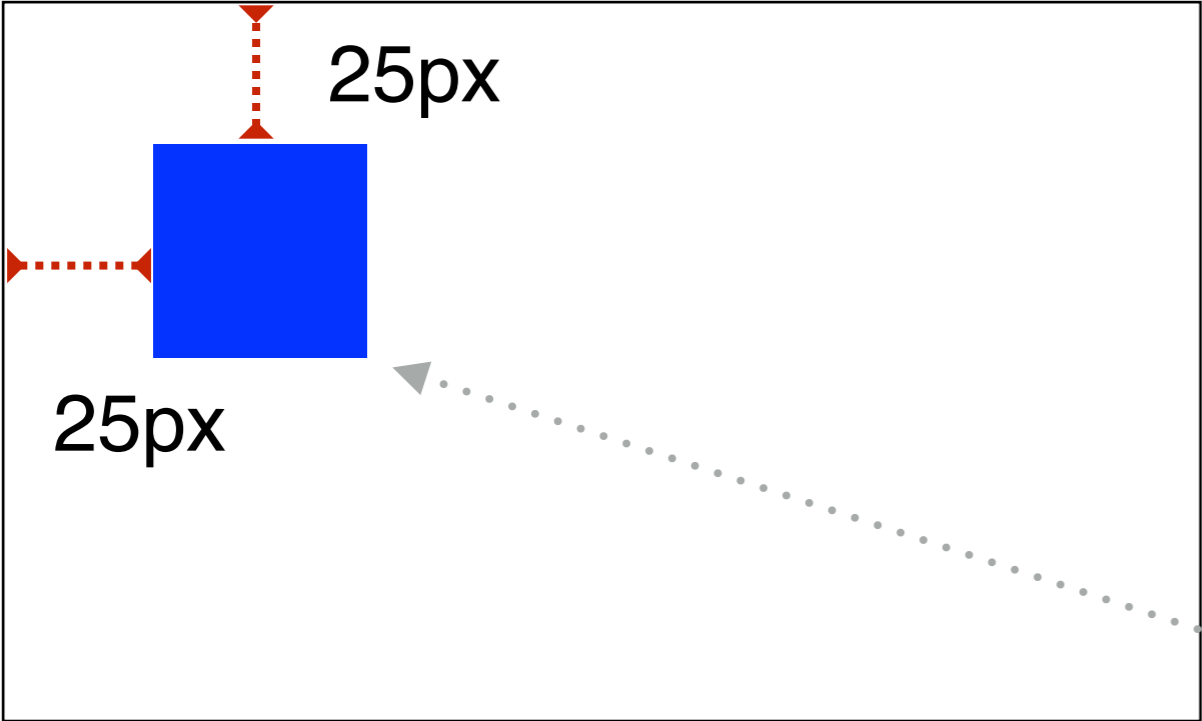
- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.

# Positioning

- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.







position: relative;

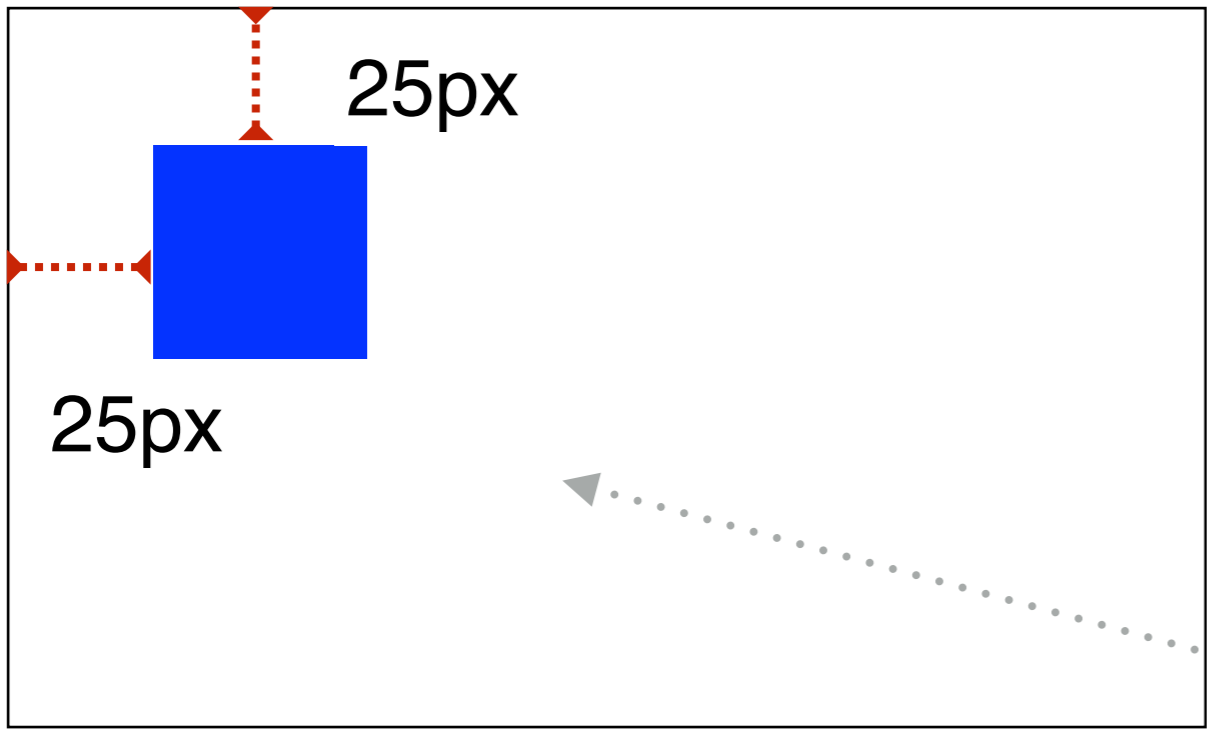
position: absolute;  
top: 25px;  
left: 25px;

# Positioning

- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.

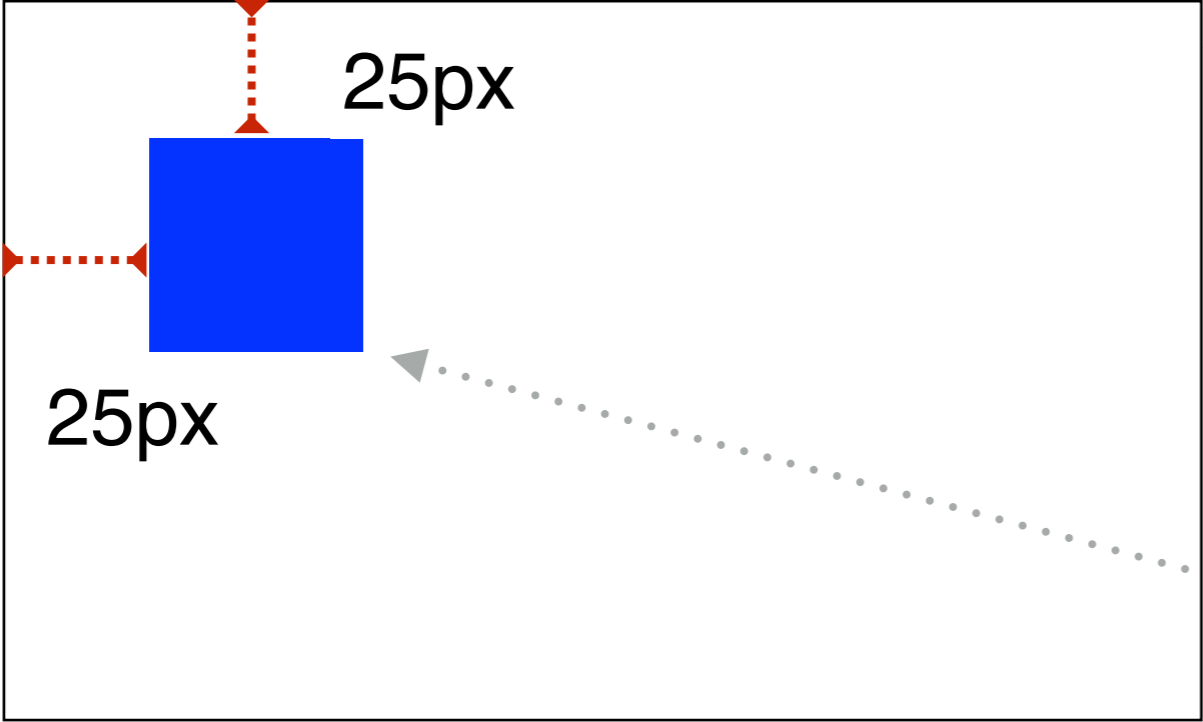
# Positioning

- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.



```
position: absolute;  
top: 50px;  
left: 50px;
```

```
position: absolute;  
top: 25px;  
left: 25px;
```



```
position: absolute;  
top: 50px;  
left: 50px;
```

```
position: absolute;  
top: 25px;  
left: 25px;
```

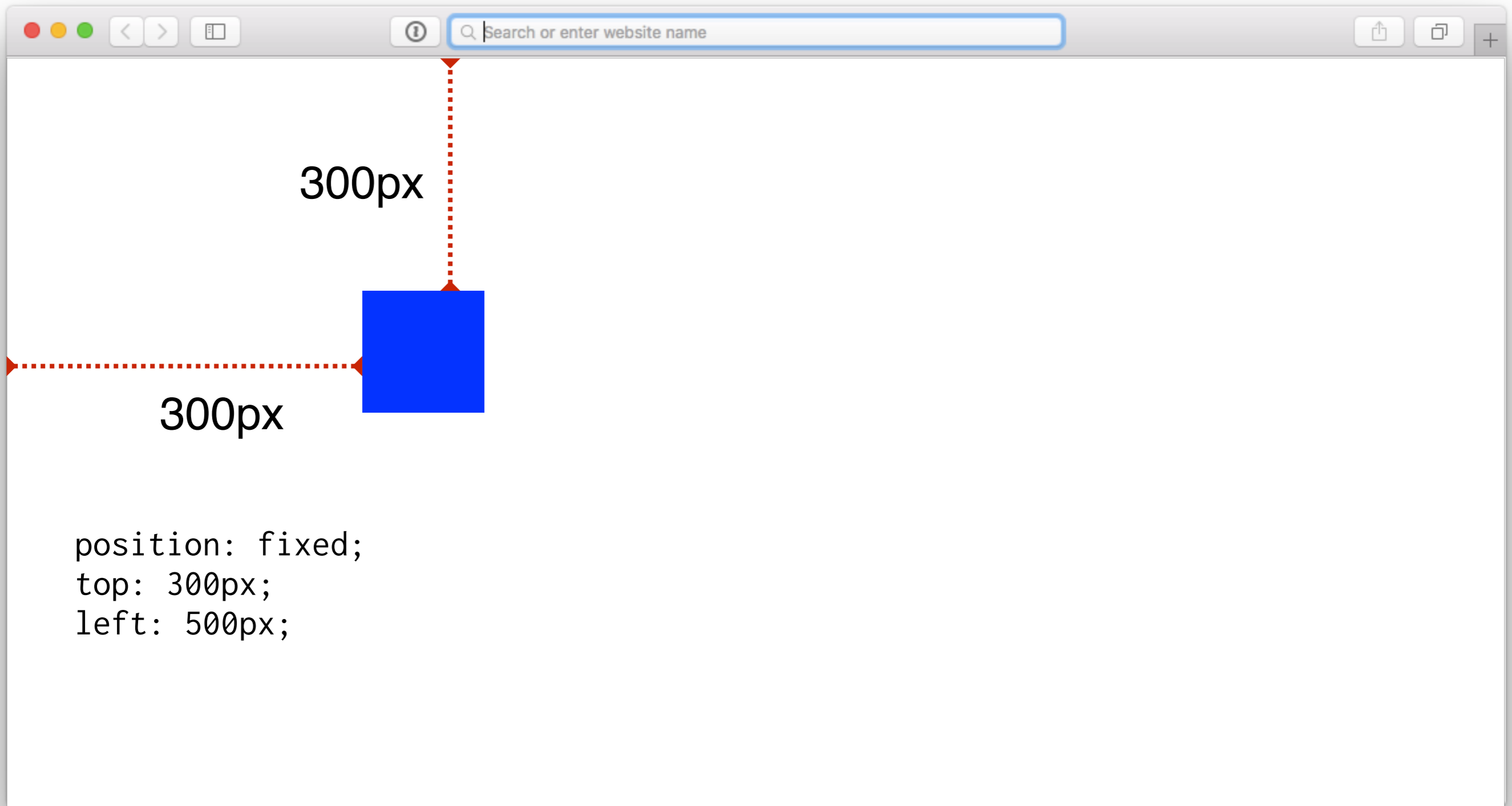
# Positioning

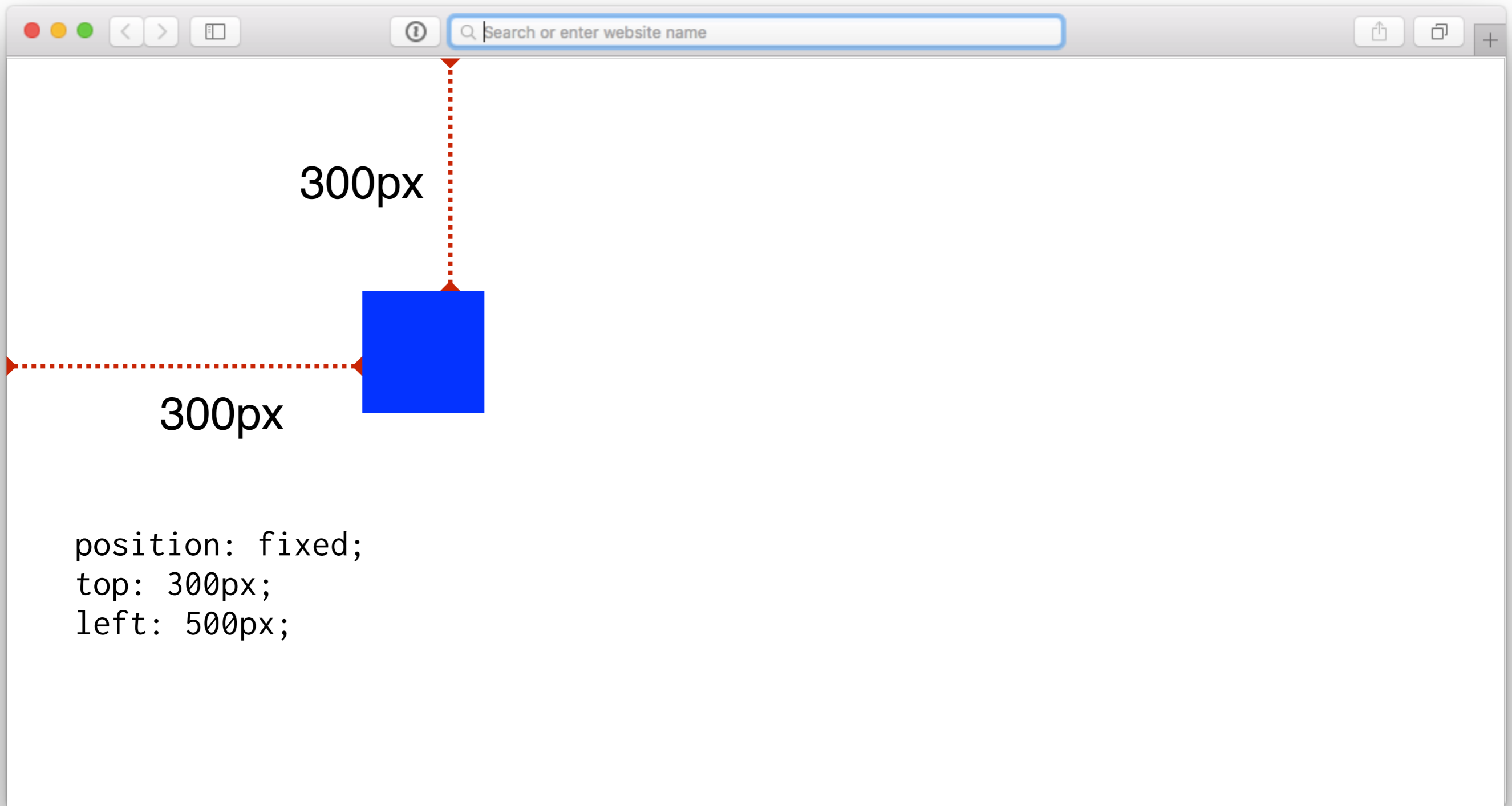
- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.

# Positioning

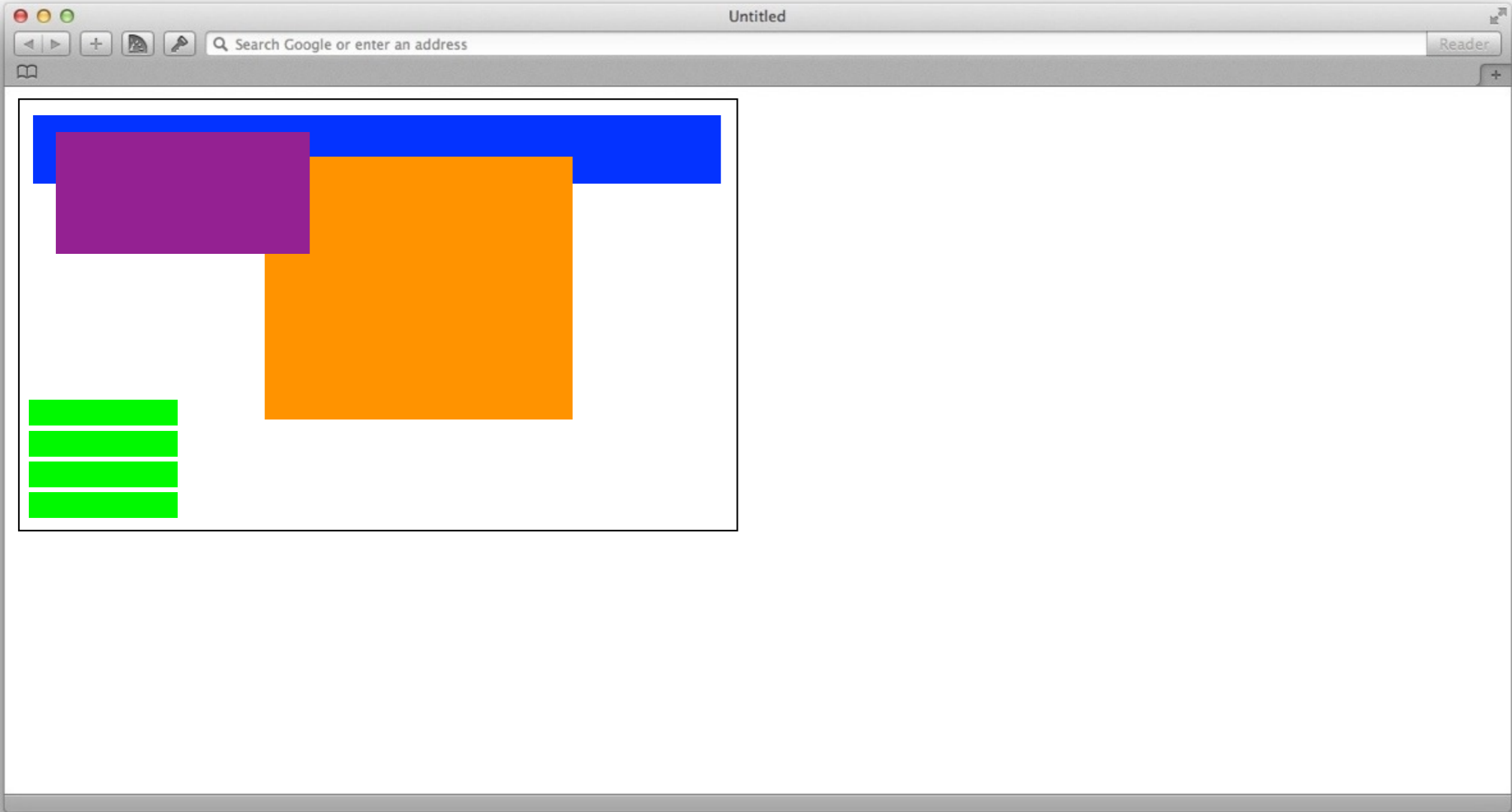
- **Static** - This is the default positioning. Elements are arranged according to the normal document flow.
- **Relative** - This is identical to static, but causes elements inside this tag to use it as a frame of reference.
- **Absolute** - Elements are positioned separate from the document flow. Items will be located relative to the first parent element that has any positioning other than static.
- **Fixed** - Position elements separate from the document flow, but relative to the browser. Stays in the same spot even when scrolled.







**z-index**



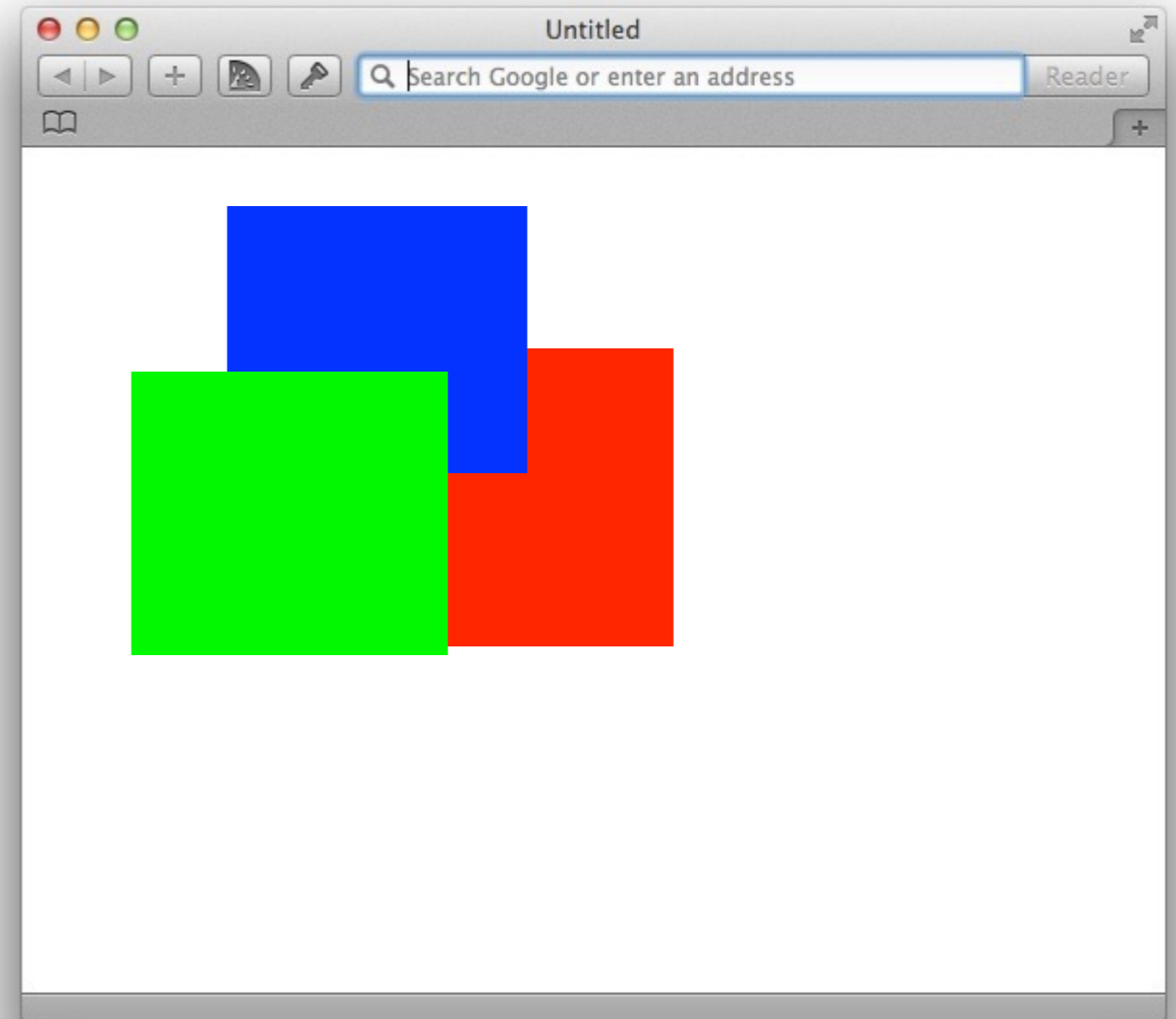
- **z-index** property is an arbitrary number that determines the *stacking order*.
- A higher **z-index** number will indicate those elements should be on top. A lower number means they should appear underneath other elements.
- **z-index** property can only be applied to elements that are positioned with *relative*, *absolute* or *fixed*, but **not** the default *static*.

# HTML:

```
<div class="container">
  <div class="redbox"></div>
  <div class="bluebox"></div>
  <div class="greenbox"></div>
</div>
```

# CSS:

```
.redbox{
  position:absolute;
  top: 100px; left:300px;
}
.bluebox{
  position:absolute;
  top: 10px; left: 200px;
}
.greenbox{
  position:absolute;
  top:110px; left: 50px;
}
```

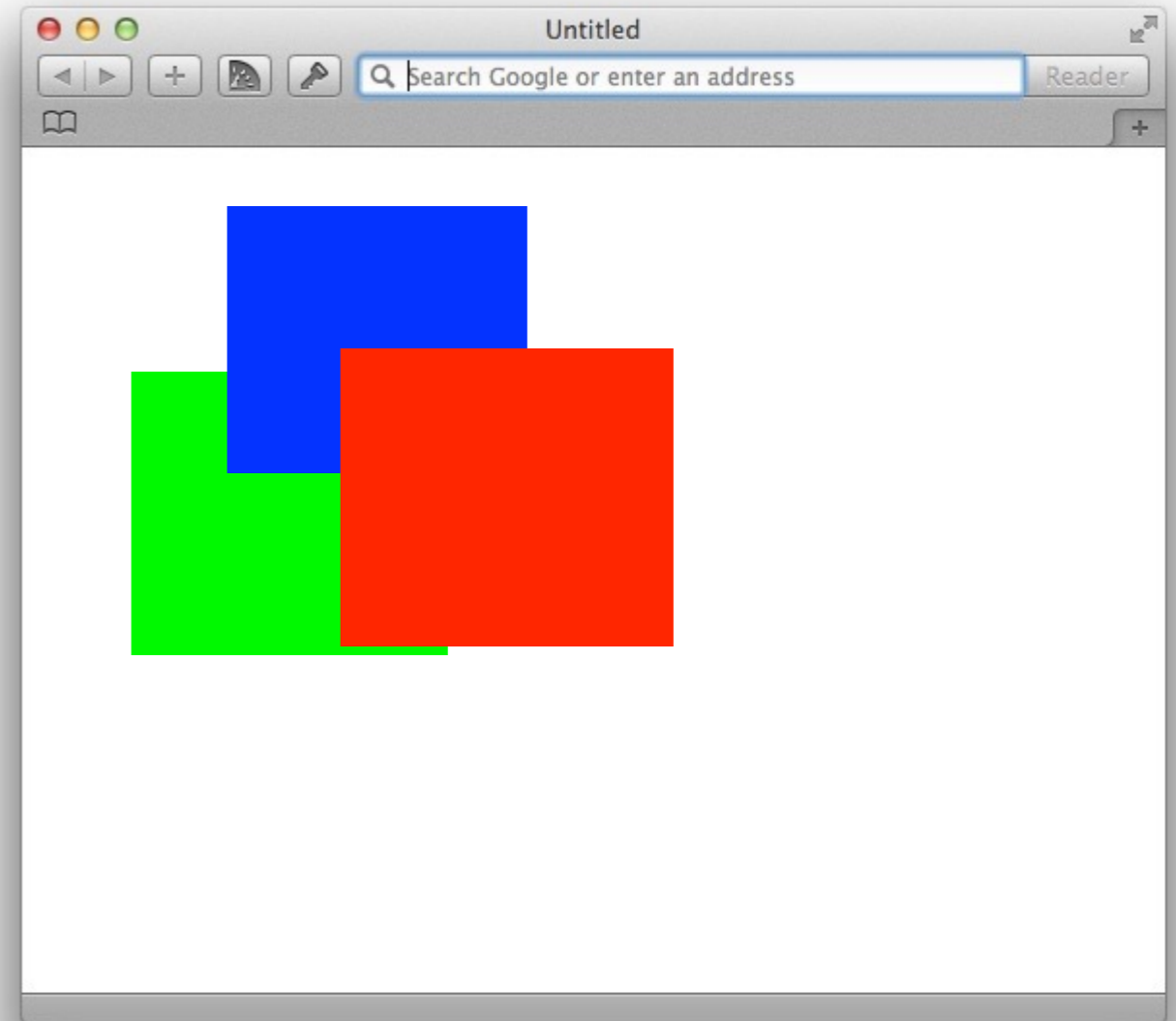


# HTML:

```
<div class="container">
  <div class="redbox"></div>
  <div class="bluebox"></div>
  <div class="greenbox"></div>
</div>
```

# CSS:

```
.redbox{
  position:absolute;
  z-index: 100;
}
.bluebox{
  position:absolute;
  z-index: 50;
}
.greenbox{
  position:absolute;
  z-index: 1;
}
```

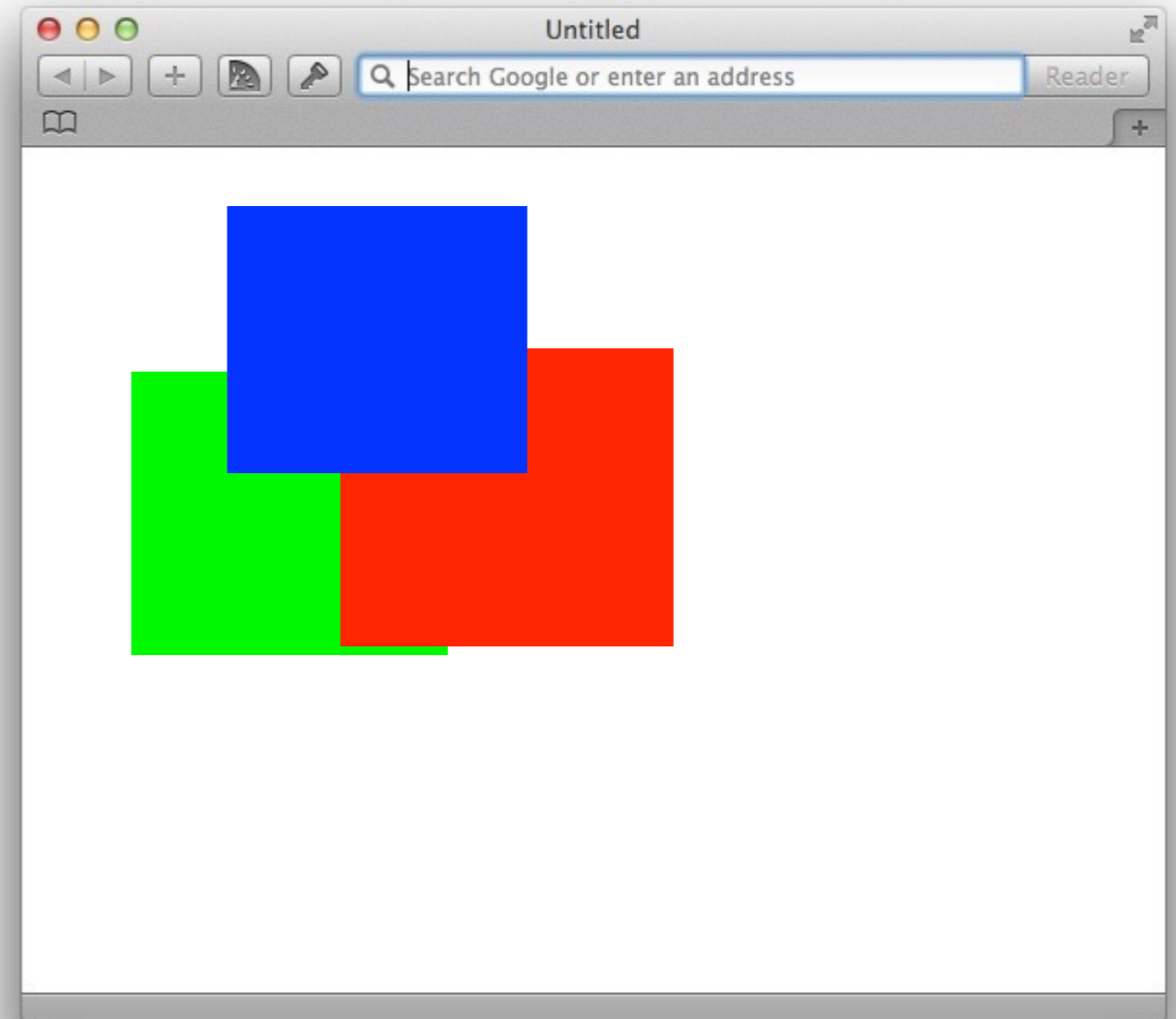


# HTML:

```
<div class="container">
  <div class="redbox"></div>
  <div class="bluebox"></div>
  <div class="greenbox"></div>
</div>
```

# CSS:

```
.redbox{
  position:absolute;
  z-index: 500000000000;
}
.bluebox{
  position:absolute;
  z-index: 500000000001;
}
.greenbox{
  position:absolute;
  z-index: -100;
}
```



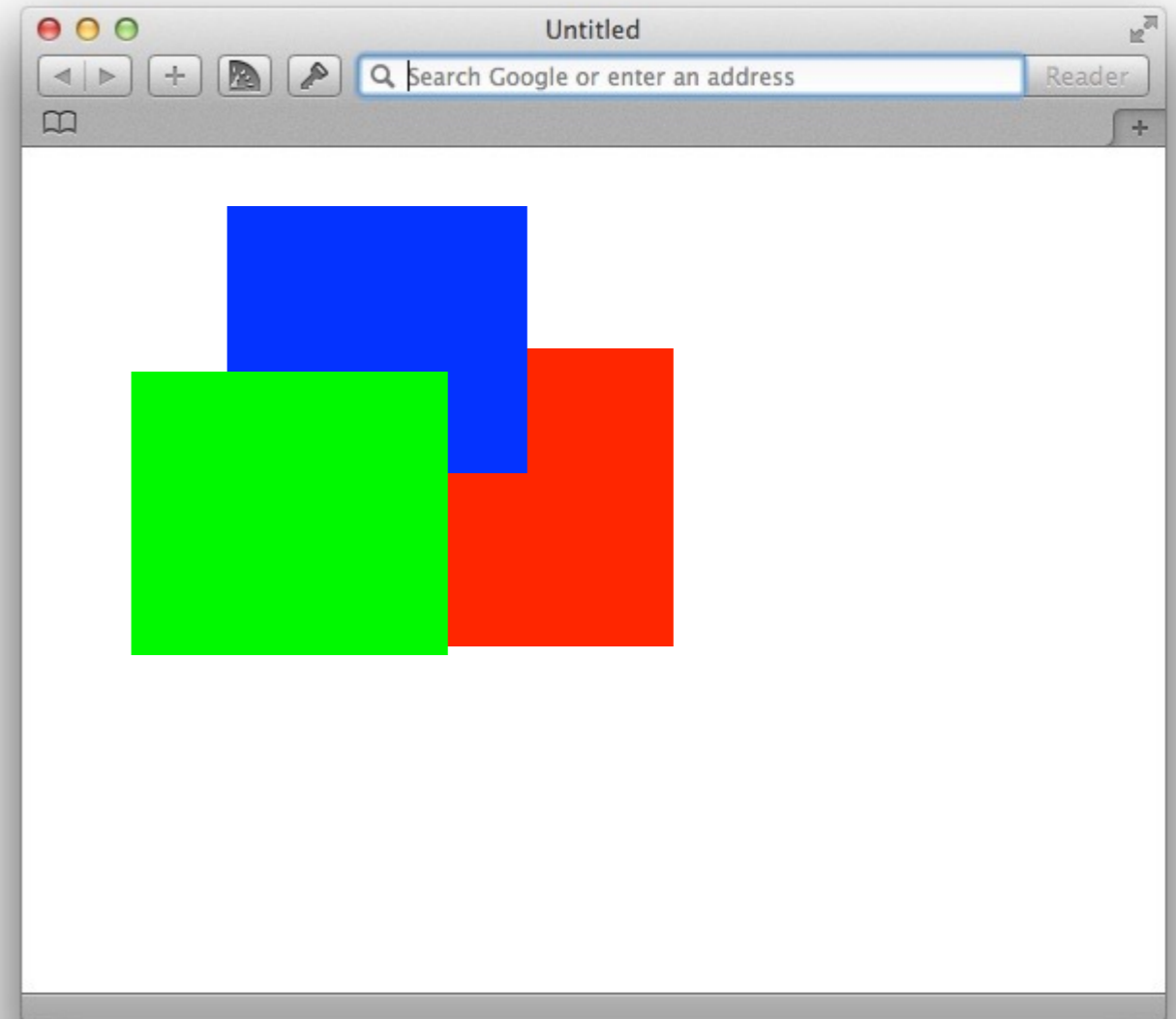


# HTML:

```
<div class="container">
  <div class="redbox"></div>
  <div class="bluebox"></div>
  <div class="greenbox"></div>
</div>
```

# CSS:

```
.redbox{
  position:absolute;
  z-index: 10;
}
.bluebox{
  position:absolute;
  z-index: 10;
}
.greenbox{
  position:absolute;
  z-index: 11;
}
```



**transforms**

- Transforms are based on an element's **current position.**
- Transforms are x value first (horizontal) and y value second (vertical)

# Transforms

```
transform: translate(10px, 5px);
```

```
transform: translateX(10px);
```

```
transform: translateY(5px);
```

# Transform rotation, scale

```
transform: rotate(7deg);  
transform: scale(1.2);
```

# Multiple transforms

```
transform: translateX(5px) rotate(7deg) scale(1.2);
```