

css

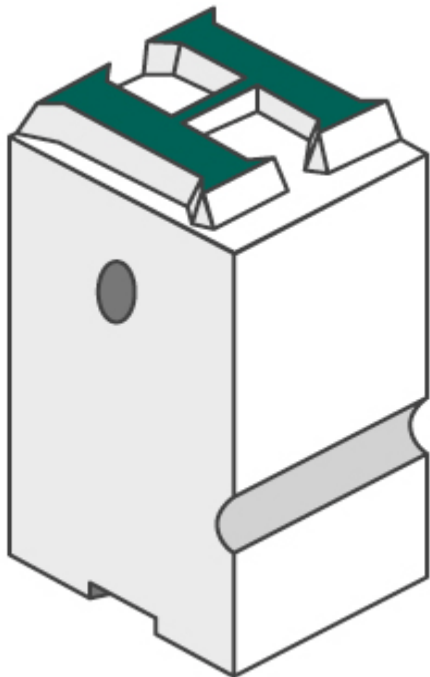
LINE-HEIGHT

**What is  
leading?**

Back in the “good old days” type  
was set by hand using **printing  
presses.**



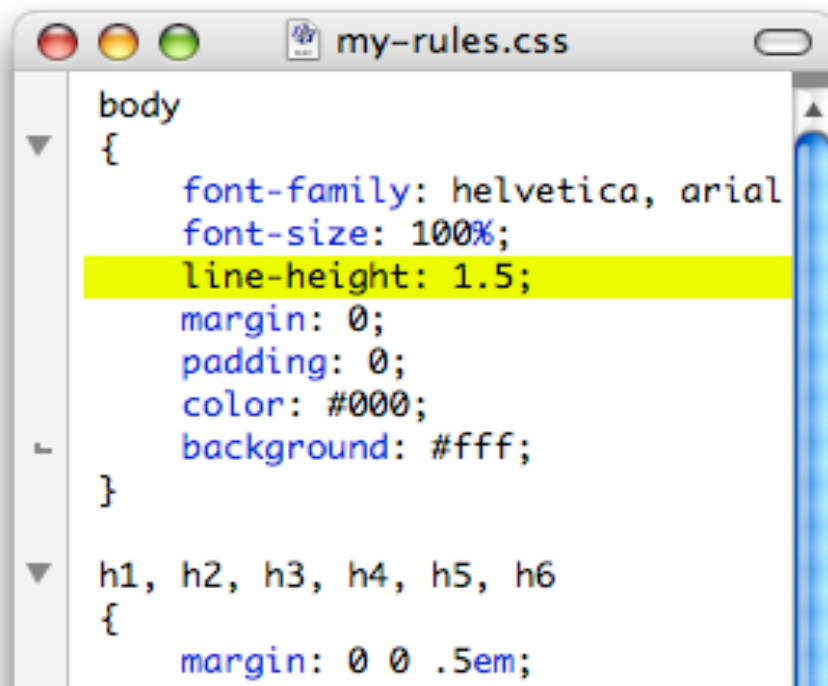
Printed material was created by setting out letters in rows. Each letter was created on an **individual block.**



**Leading**, or lead strips were added between the lines of letters when additional vertical space was required.



In CSS, **line-height** is used to control the vertical space between lines.



```
my-rules.css
body
{
  font-family: helvetica, arial
  font-size: 100%;
  line-height: 1.5;
  margin: 0;
  padding: 0;
  color: #000;
  background: #fff;
}
h1, h2, h3, h4, h5, h6
{
  margin: 0 0 .5em;
```

However, the terms **leading** and **half-leading** are still used in association with CSS line-height.



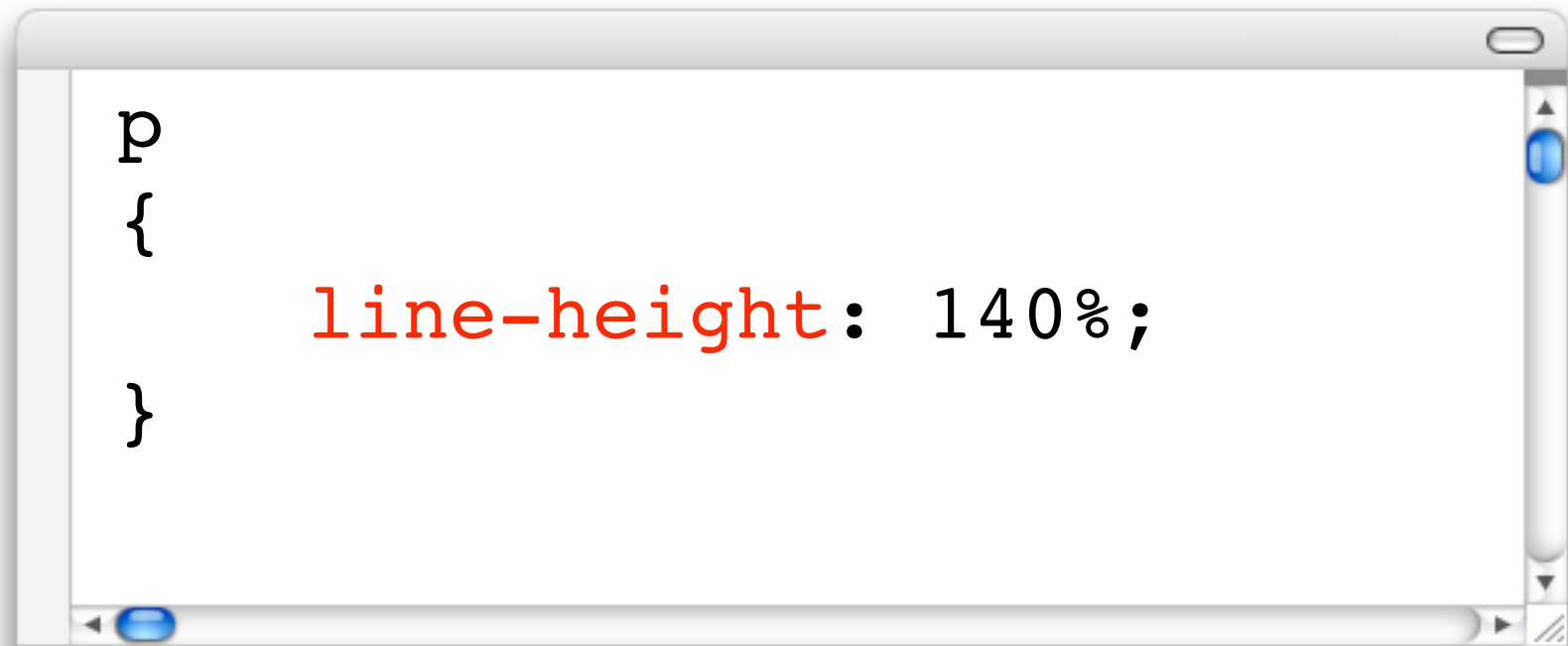
**So how do you  
apply  
line-height?**



By default, browsers use between **1.0 - 1.2** line-height. This is referred to as an initial value.



You can override this default line-height using the CSS **line-height property**.

A code editor window with a white background and a grey border. The code is as follows:

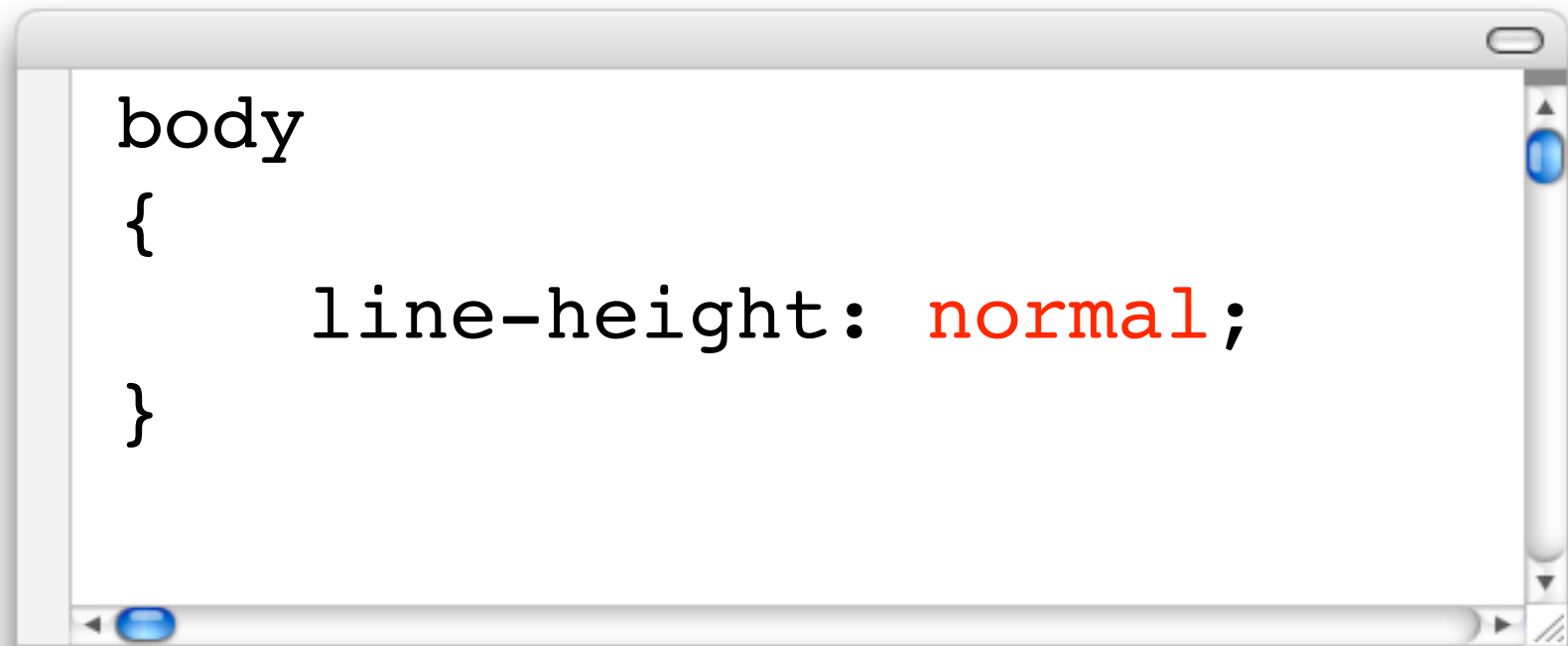
```
p  
{  
    line-height: 140%;  
}
```

The text 'line-height: 140%;' is highlighted in red. The editor has a scrollbar on the right and a scroll wheel at the bottom.

Line-height can be specified  
with **five different types of unit.**



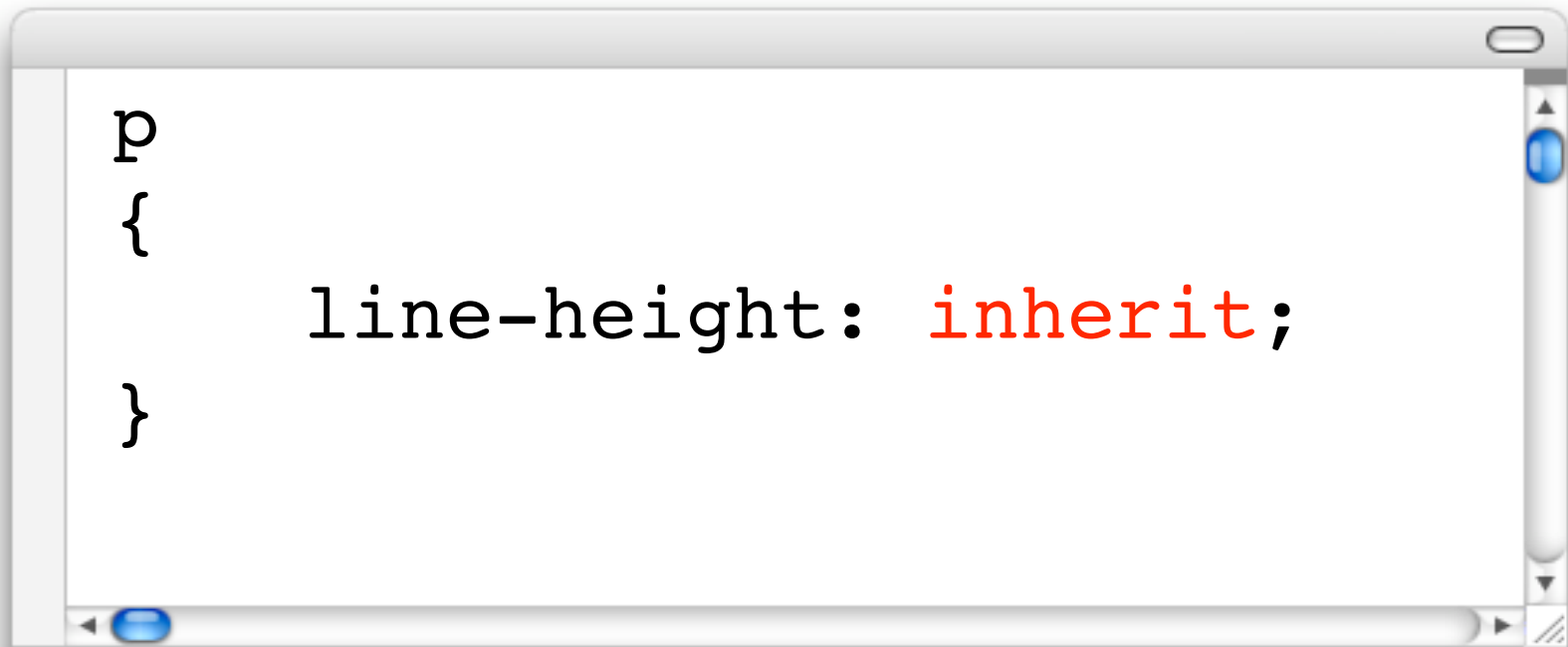
1. Line-height can be specified as **normal**.

A code editor window with a white background and a grey border. The code is as follows:

```
body
{
    line-height: normal;
}
```

The word "normal" in the CSS rule is highlighted in red. The editor has a scrollbar on the right and a scroll bar at the bottom.

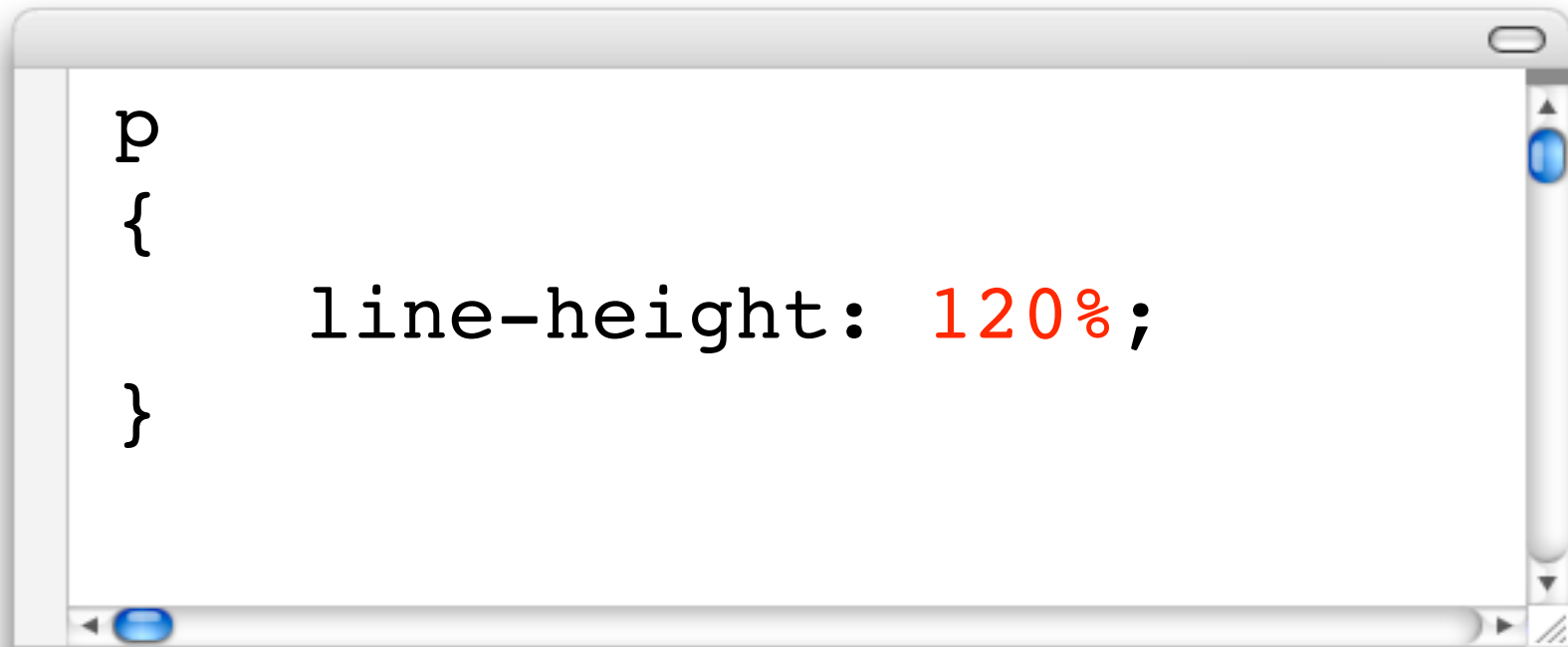
2. Line-height can be specified as **inherit**.

A code editor window with a white background and a grey border. It contains CSS code for a paragraph element. The code is: 

```
p  
{  
    line-height: inherit;  
}
```

 The word 'inherit' is highlighted in red. The editor has a scrollbar on the right and a scroll bar at the bottom.

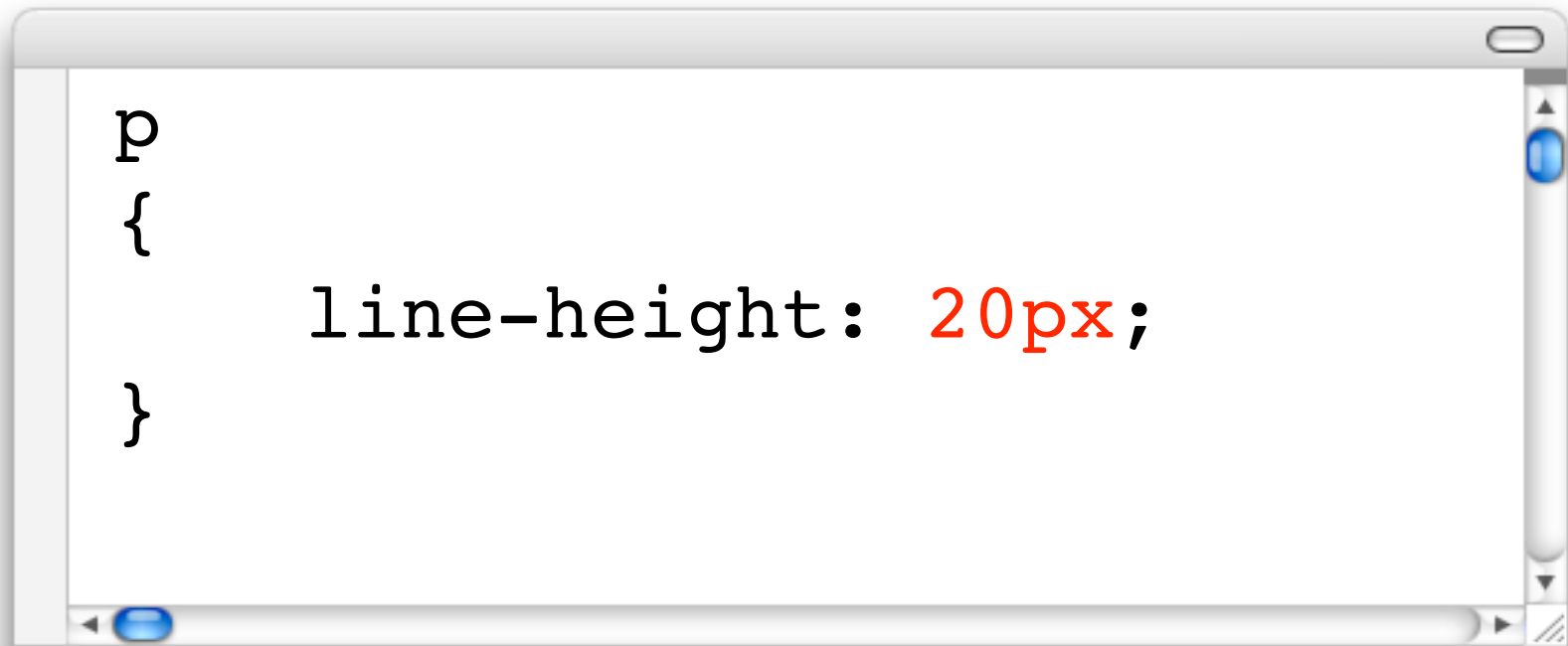
3. Line-height can be specified using a **percentage value**.

A code editor window with a white background and a grey border. The code is as follows:

```
p  
{  
    line-height: 120%;  
}
```

The text '120%' is highlighted in red. The window has a scrollbar on the right and a scroll wheel at the bottom.

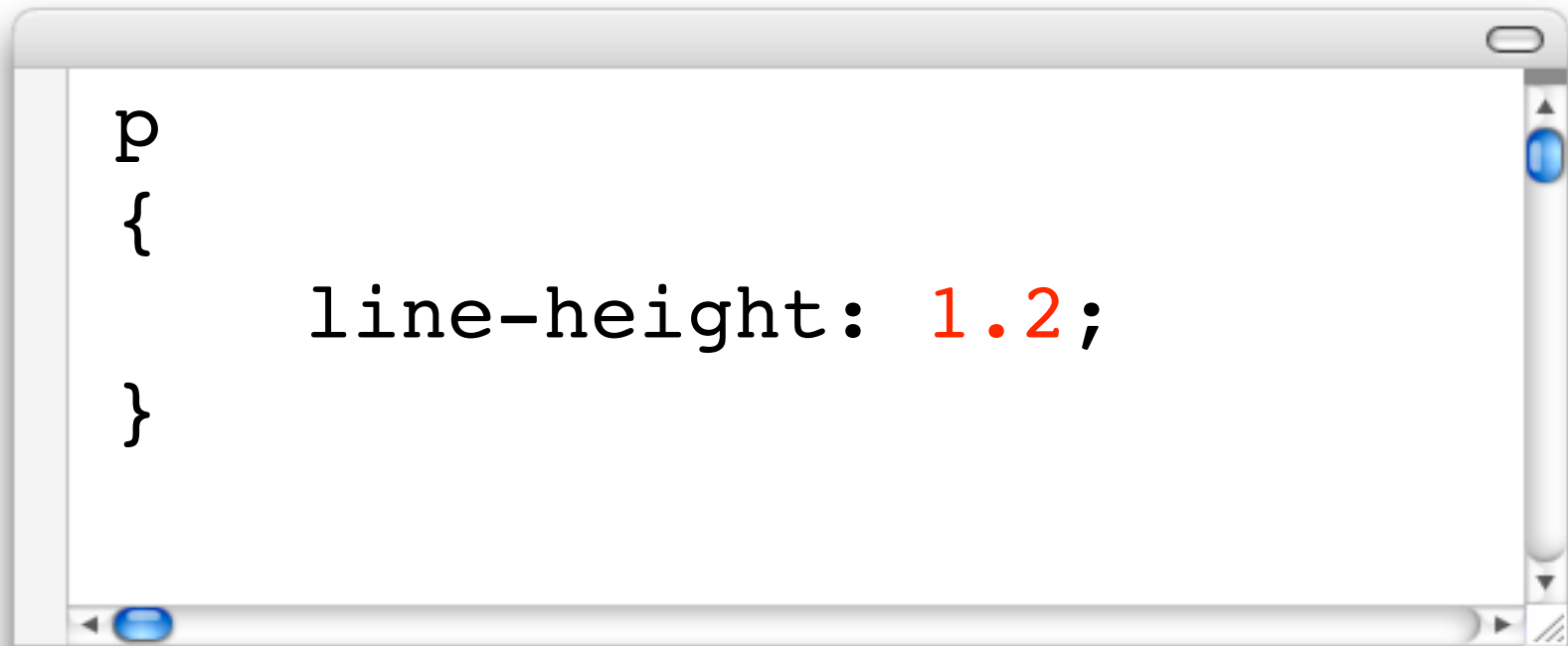
4. Line-height can be specified using a **length value** (using px, em etc).

A code editor window with a white background and a grey border. It contains CSS code for a paragraph element. The code is: 

```
p  
{  
    line-height: 20px;  
}
```

 The text '20px;' is highlighted in red. The editor has a scrollbar on the right and a scroll bar at the bottom.

5. Line-height can also be specified using a **number value** (a unit-less value).

A code editor window with a white background and a grey border. It contains CSS code for a paragraph element. The code is: 

```
p  
{  
    line-height: 1.2;  
}
```

 The value '1.2' is highlighted in red. The window has a scrollbar on the right and a scroll button at the bottom left.



**Shorthand  
line-height**

These five line-height values can also be specified using the **font shorthand property**.



The line-height value is written in conjunction with the font-size value. The values are separated by a slash:

**<font-size>/<line-height>**

For example...

# Normal value

```
body
{
    font: 100%/normal arial,
    helvetica, sans-serif;
}
```

# Percentage value

```
body
{
    font: 100%/120% arial,
    helvetica, sans-serif;
}
```

# Length value

```
body
{
    font: 100%/20px arial,
    helvetica, sans-serif;
}
```

# Number value

```
body
{
    font: 100%/1.2 arial,
    helvetica, sans-serif;
}
```

# Calculating line-height



Some CSS properties are **inherited** (passed down to descendant elements).



This is designed to **make it easier for authors** - so they do not have to specify properties for all descendants.



EG. the **color property** is inherited. If a color is applied to the body element, it will be passed down to all other elements on the page.



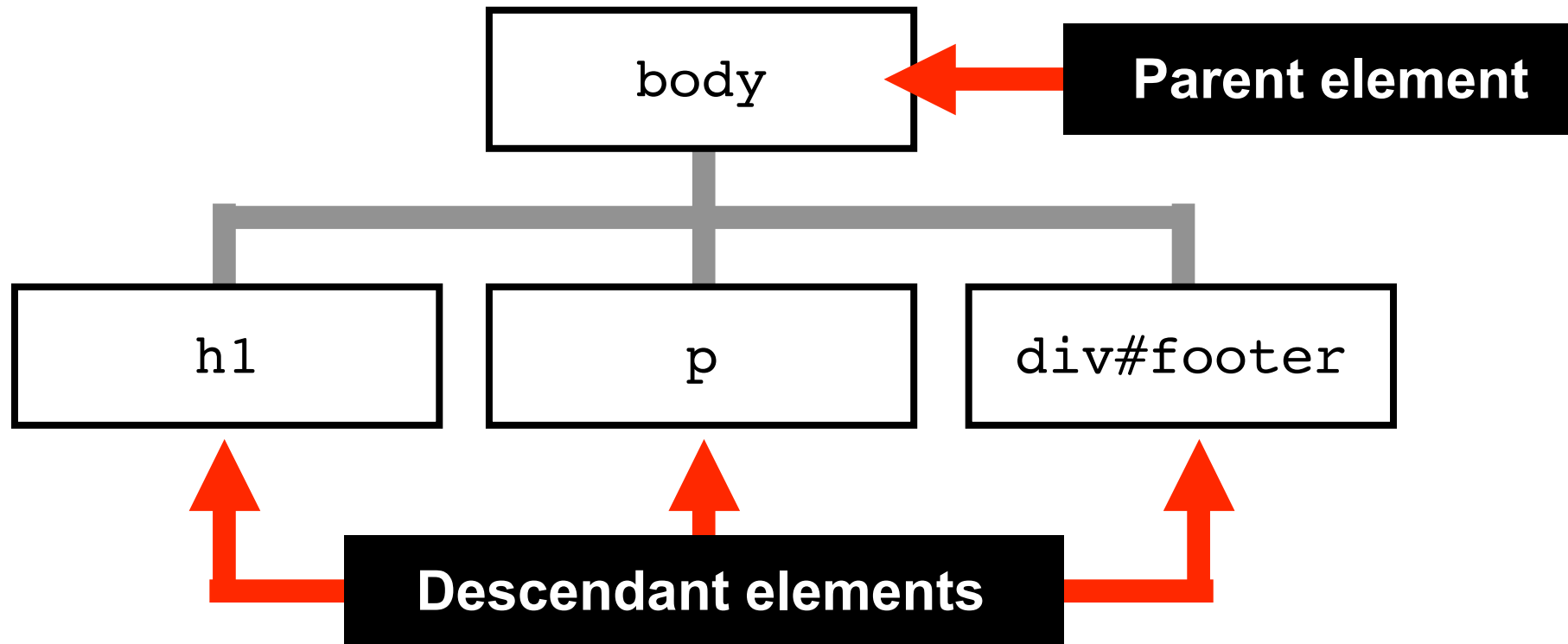
For line-height, inheritance is a  
little **more complicated...**



To see the various line-height options in action, we will use the following **HTML code**:

```
<h1>
    consect etuer adipi scing eli
</h1>
<p>
    Lorem ipsum dolor sit amet co
</p>
<div id="footer">
    Duis autem vel eum iriure dol
</div>
```

This produces the following **document tree**:



# We will also use the following **CSS**

(pixels used for font-size to simplify - though not recommended!):

```
body
{
    font-size: 16px;
    line-height: XXX;
}

h1 { font-size: 32px; }
p { font-size: 16px; }
#footer { font-size: 12px; }
```

# **Example 1**

## **The percentage value**



The line-height has been set to a **percentage value** (120%).

```
body
{
    font-size: 16px;
    line-height: 120%;
}

h1 { font-size: 32px; }
p { font-size: 16px; }
#footer { font-size: 12px; }
```

The percentage value (120%) and the body element's font size (16px) are used to create a **calculated value** ( $16\text{px} \times 120\% = 19.2\text{px}$ ). This calculated value is inherited by descendant elements.



All descendants, regardless of their size, receive the **same calculated value line-height**.

element	font-size	line height	calculated line-height
body	16px	120%	16px x 120% = 19.2px
h1	32px	inherits calculated value - 19.2px	19.2px
p	16px	inherits calculated value - 19.2px	19.2px
#footer	12px	inherits calculated value - 19.2px	19.2px

The line-heights **do not scale**  
**with** the relevant font size.



# **Example 2**

## **the length value**

The line-height has been set using a **length value** (20px).

```
body
{
    font-size: 16px;
    line-height: 20px;
}

h1 { font-size: 32px; }
p { font-size: 16px; }
#footer { font-size: 12px; }
```

The **length value (20px)** is inherited by descendant elements.



All elements, regardless of their font-size, receive the **same line-height**.

element	font-size	line height	calculated line-height
body	16px	20px	20px
h1	32px	inherits 20px	20px
p	16px	inherits 20px	20px
#footer	12px	inherits 20px	20px



Again, the line-heights **do not scale with** the relevant font size.



# **Example 3**

## **the normal value**

The line-height has been set to **normal** (which is approx 1.2).

```
body
{
    font-size: 16px;
    line-height: normal;
}

h1 { font-size: 32px; }
p { font-size: 16px; }
#footer { font-size: 12px; }
```

In this case, the **normal value** rather than a calculated value is inherited by descendant elements. Browsers may interpret the actual normal value in slightly different ways.



All elements now have line-heights that are **relative to their font-size.**

element	font-size	line height	calculated line-height
body	16px	normal	16px x approx 1.2 = approx 19.2px
h1	32px	normal	32px x aprox 1.2 = approx 38.4px
p	16px	normal	16px x approx 1.2 = approx 19.2px
#footer	12px	normal	11.2px x approx 1.2 = approx 13.44px

Now, the line-heights **scale with**  
the relevant font size.



But what if you want the flexibility  
of the normal value, but to be  
able to specify the factor used?  
This is where **number values**  
come in.



# **Example 4**

## **the number value**



The line-height has been set to a **number value** (1.5).

```
body
{
    font-size: 16px;
    line-height: 1.5;
}

h1 { font-size: 32px; }
p { font-size: 16px; }
#footer { font-size: 12px; }
```

In this case, the **factor (1.5)** rather than a calculated value is inherited by descendant elements.



All elements now have line-heights that are **relative to their font-size.**

element	font-size	line height	calculated line-height
body	16px	1.5	16px x 1.5 = 24px
h1	32px	factor of 1.5	32px x 1.5 = 48px
p	16px	factor of 1.5	16px x 1.5 = 24px
#footer	12px	factor of 1.5	12px x 1.5 = 18px

Again, the line-heights **scale with**  
the relevant font size.



**consect etuer adipi scing elit**

Too loose?

**sed diam nonummy**

Lorem ipsum dolor sit amet consectetur adipiscing elit sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volut. Ut wisi enim ad minim veniam.

OK

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros.

OK

**So, which is the  
best method?**

Generally, a **number value** is the best method for setting line-height as line-heights will then always scale with the relevant font-size.

It is hard to determine a “**perfect line-height**” as each situation is different. However, it is safe to assume that headings can have less relative line-height than paragraphs of text.

For example, all content could be set to **1.5**, and then headings redefined to **1.2**.

```
body
{ line-height: 1.5; }

h1, h2, h3, h4, h5, h6
{ line-height: 1.2; }
```



The Web Content Accessibility Guidelines (WCAG) 2.0 state:  
**“line spacing is at least space-and-a-half within paragraphs”**.

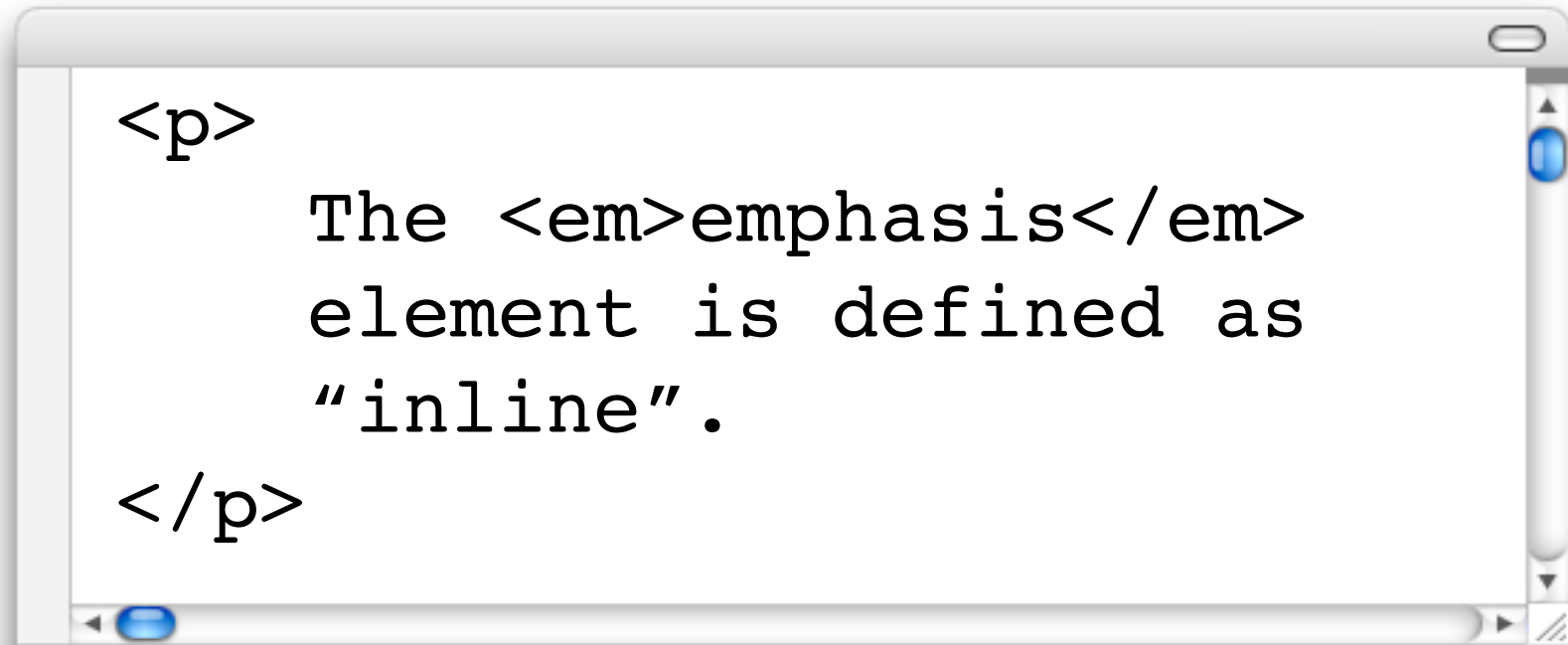
This means that to be AAA compliant, paragraph line-height should be set to 1.5

**Looking deeper  
into line-height**

In order to understand  
line-height more fully, we need to  
look at **various types of CSS  
boxes.**



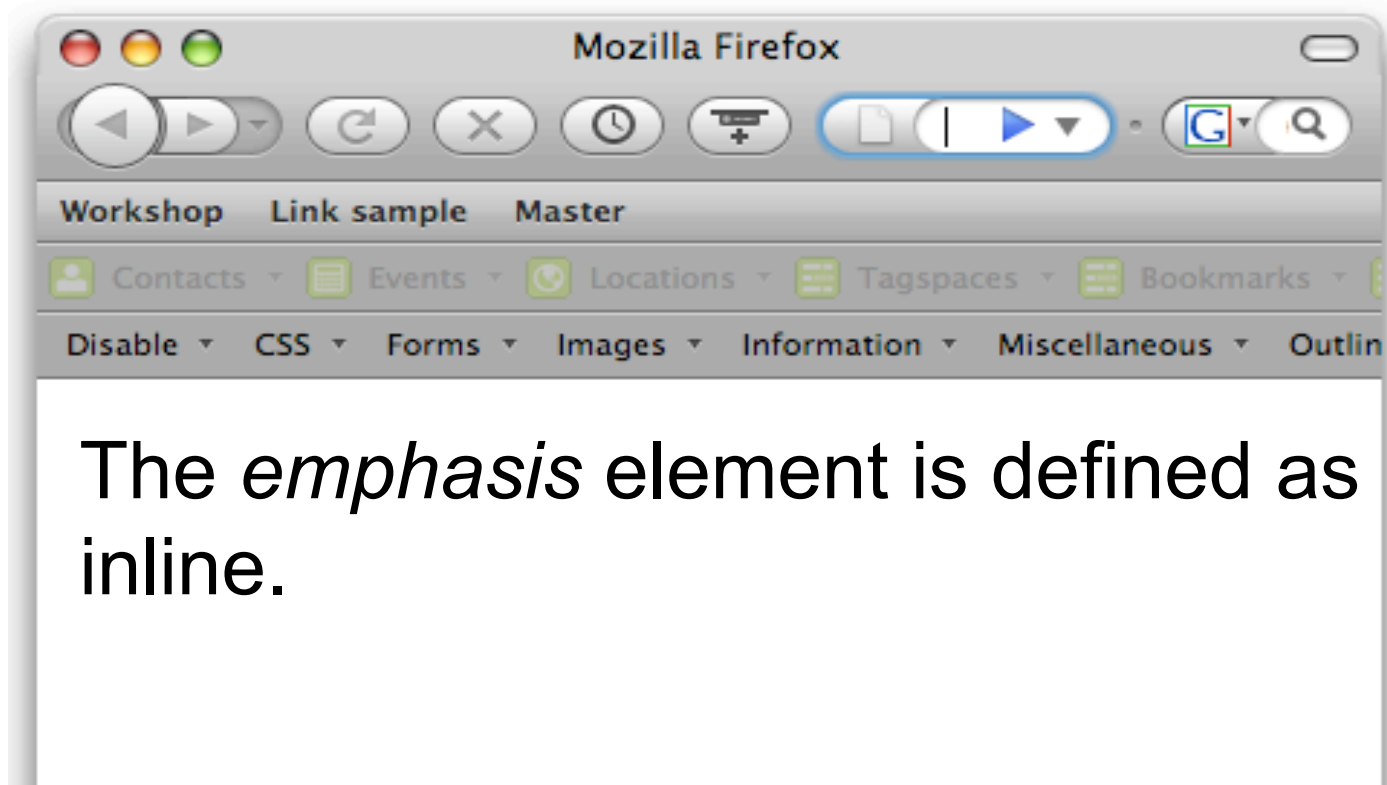
Let's start with a simple piece of  
**HTML code.**

A code editor window with a white background and a grey border. It contains the following HTML code:

```
<p>  
  The <em>emphasis</em>  
  element is defined as  
  "inline".  
</p>
```

The code is displayed in a monospaced font. The window has a scrollbar on the right and a scroll bar at the bottom.

The code should be **rendered** like this in most browsers.

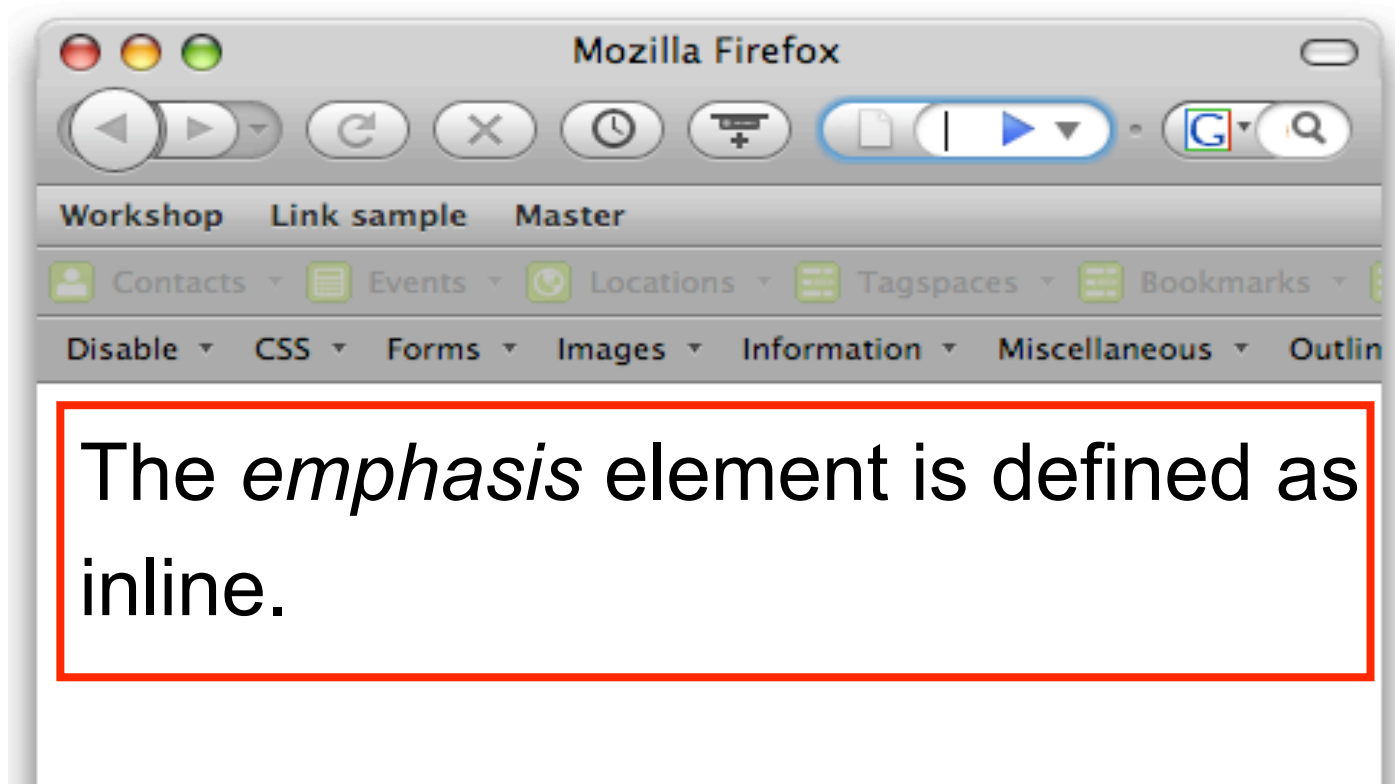


There are **four types of boxes** that are relevant in this sample.



**Box type 1:  
containing  
boxes**

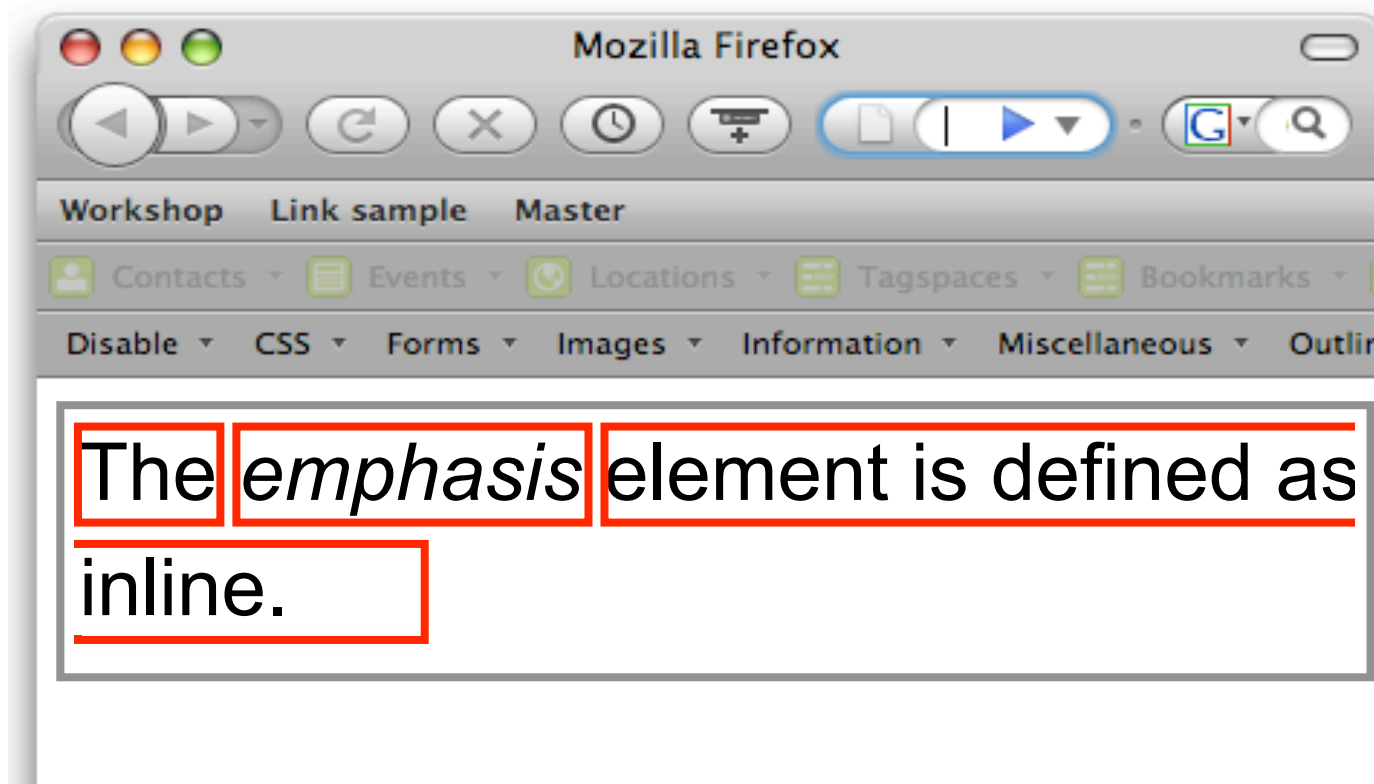
The paragraph is referred to as a **containing box** in this case - as it contains other boxes.





**Box type 2:  
inline boxes**

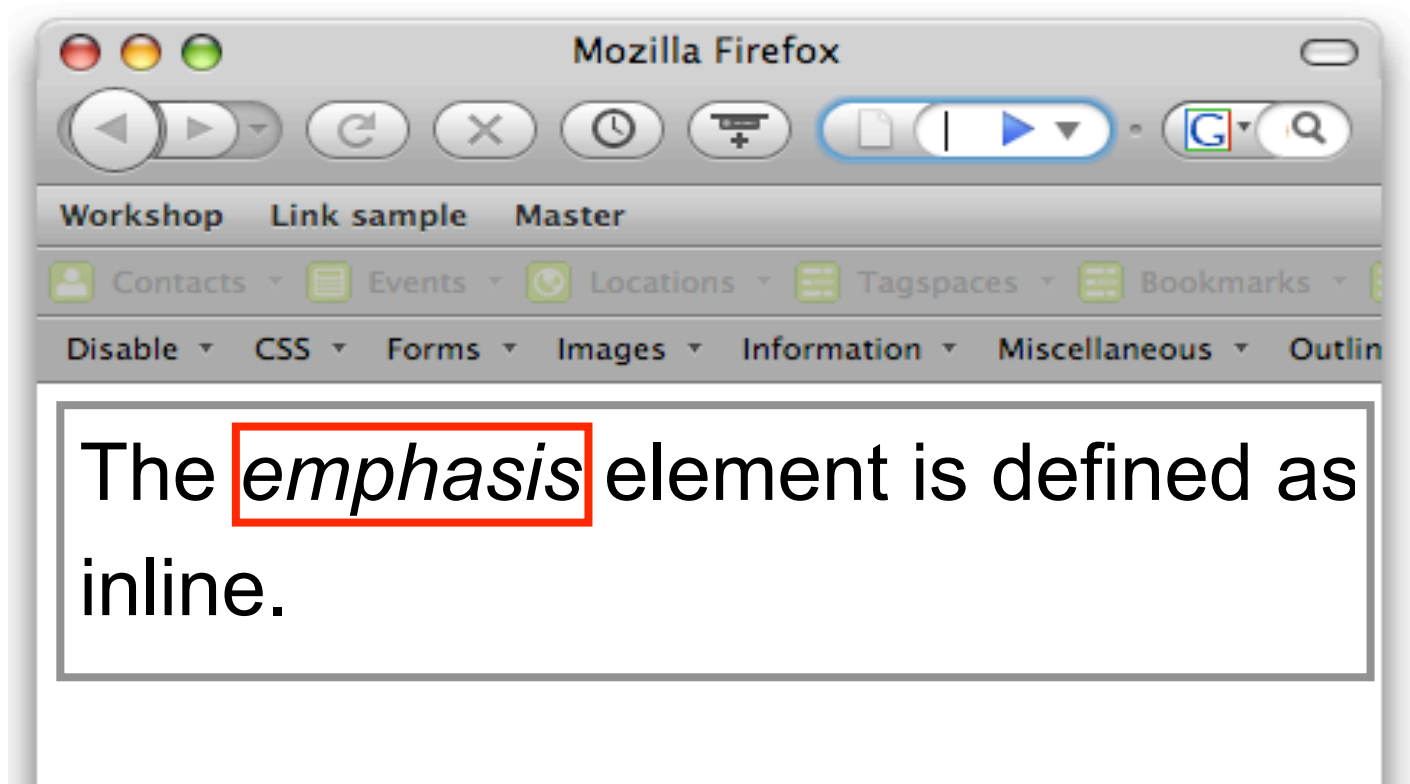
Inside the paragraph are a series of **inline boxes**.



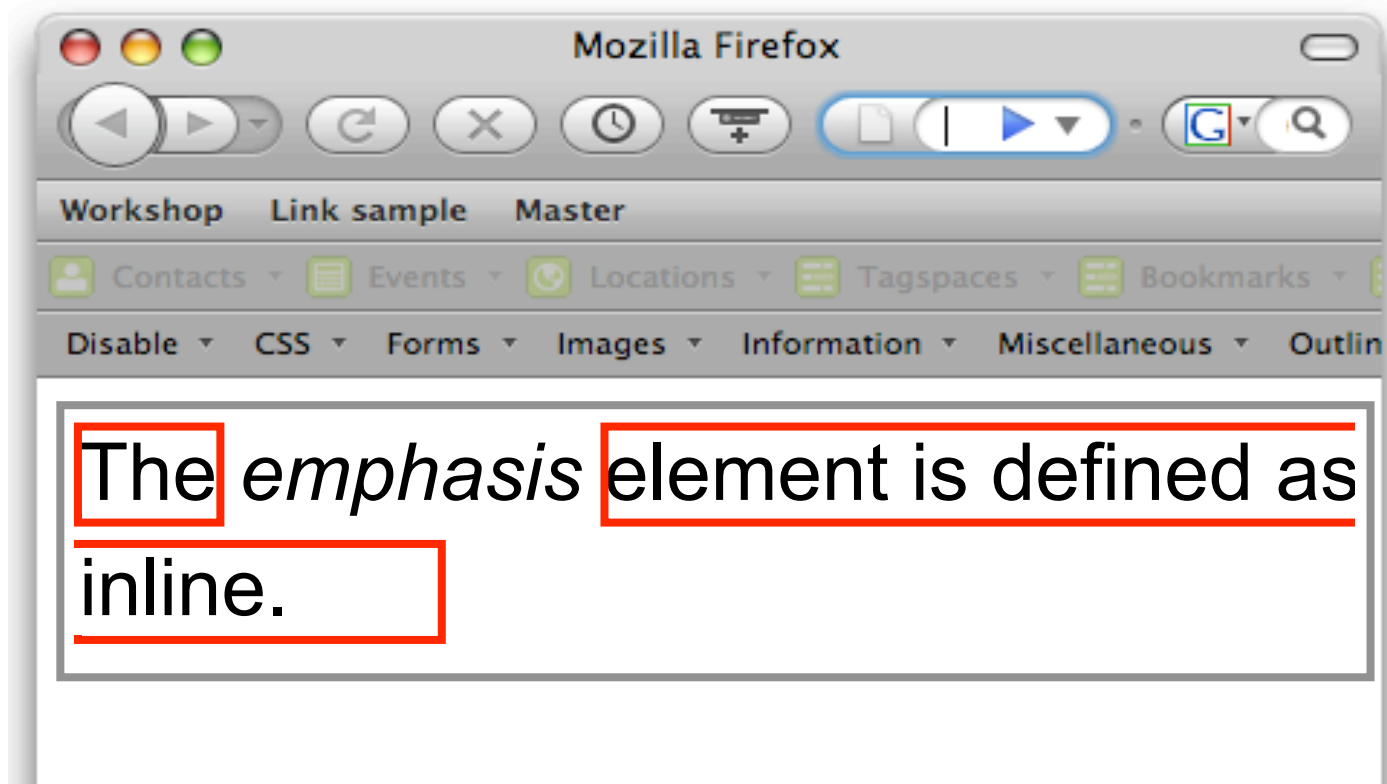
Inline boxes do not form new  
blocks of content; the content is  
**distributed in lines.**



The **emphasis element** is a type of inline box.

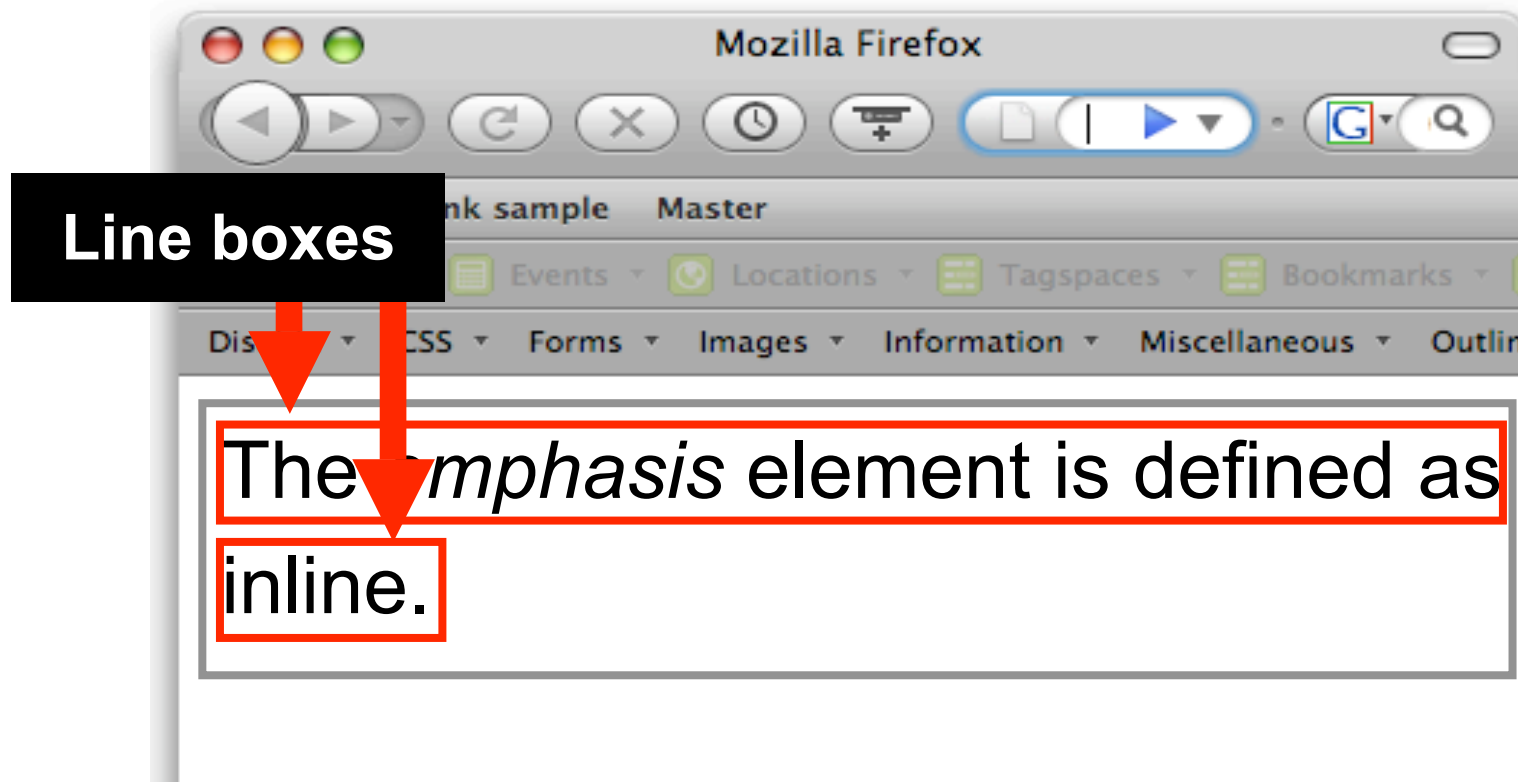


Other boxes without specific markup are referred to as **anonymous inline boxes**.



**Box type 3:  
line boxes**

Inline boxes sit side-by-side within the containing box, forming **line boxes**.



**Box type 4:  
content area**



The **content area** is the invisible box that surrounds the text. Its height is determined by the font-size.



# **Inline boxes and line-height**

Line height is applied to inline boxes using a **simple formula...**



1. Find the **difference** between the font-size and line-height. This will determine the leading.

For example:

Font-size: 16px

Line-height: 20px

Difference: 4px (leading)

2. **Divide** the leading in half to create a “half-leading” value.

4px leading / 2 = 2px half-leading

3. Apply this half-leading value to the **top and bottom** of the content area.



But things sometimes become a  
little **more complicated...**



The **inline box** generally wraps around the content box. Half-leading sits above and below the content box.





However, the inline box can  
**sometimes be smaller** than the  
content area!



For example, if the line-height is **smaller than the font size**. In this case, the inline box will honor the line height.

For example:  
Font-size: 16px  
Line-height: 12px  
Inline box size: 12px

The content area then **pokes out** the top and bottom of the inline box. The half-leading collapses together to form the inline box height.



**Some notes on  
line box height**

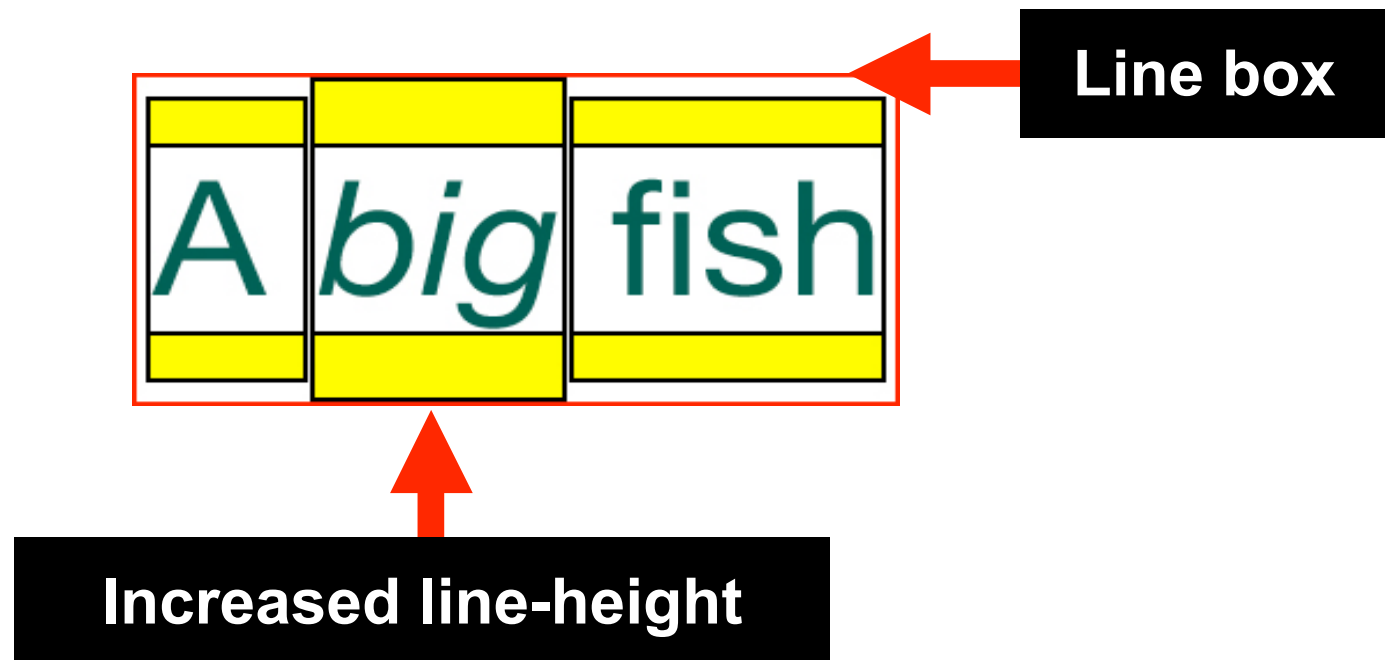
The height of line boxes is determined by the **tallest inline box** (or replaced element) inside.



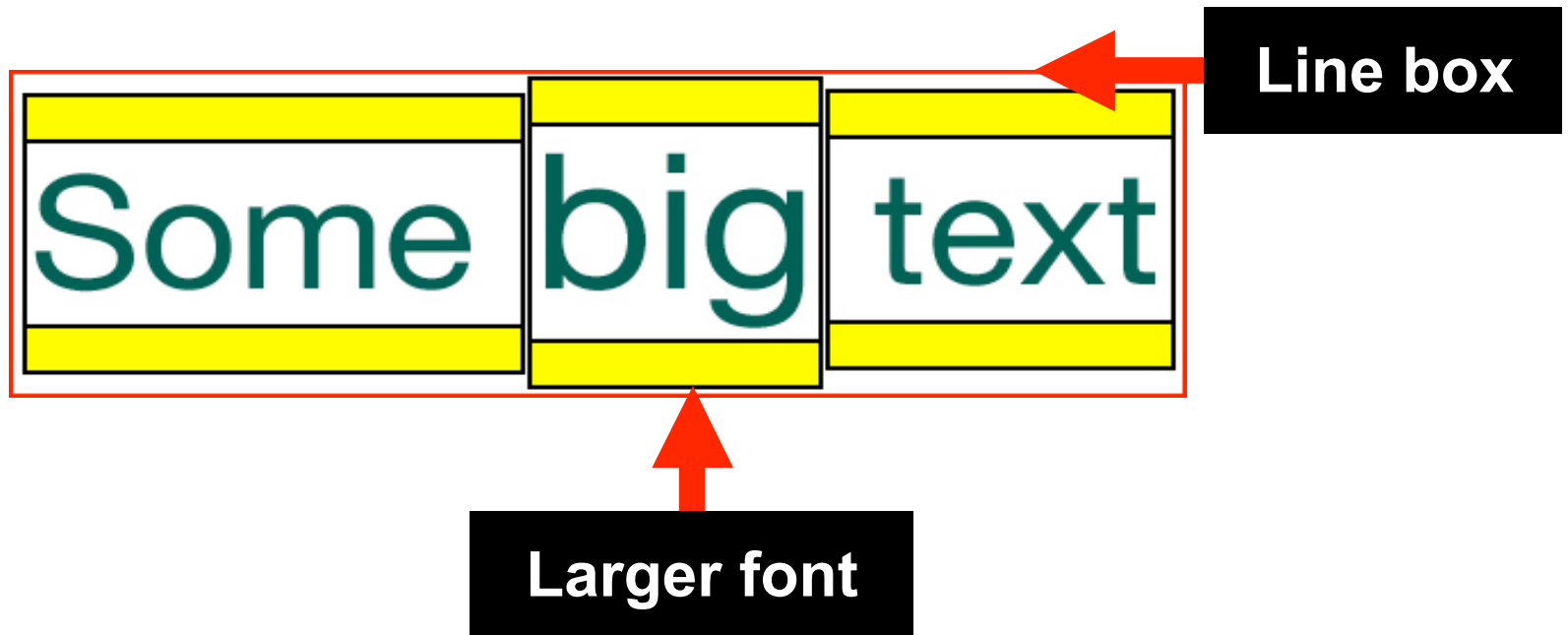
The tallest inline box could be an **anonymous inline box**.



It could be an inline box with **increased line-height.**

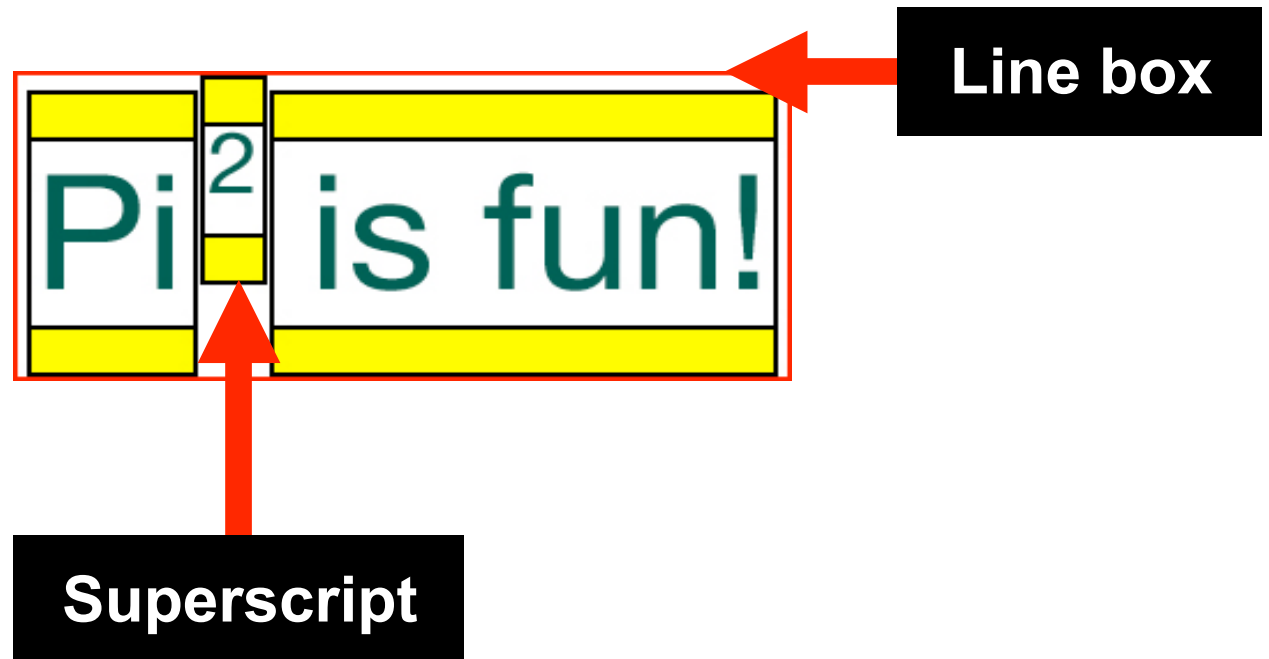


Or an inline box with a  
**larger font-size.**





Or the presence of a **superscript**  
or **subscript**.

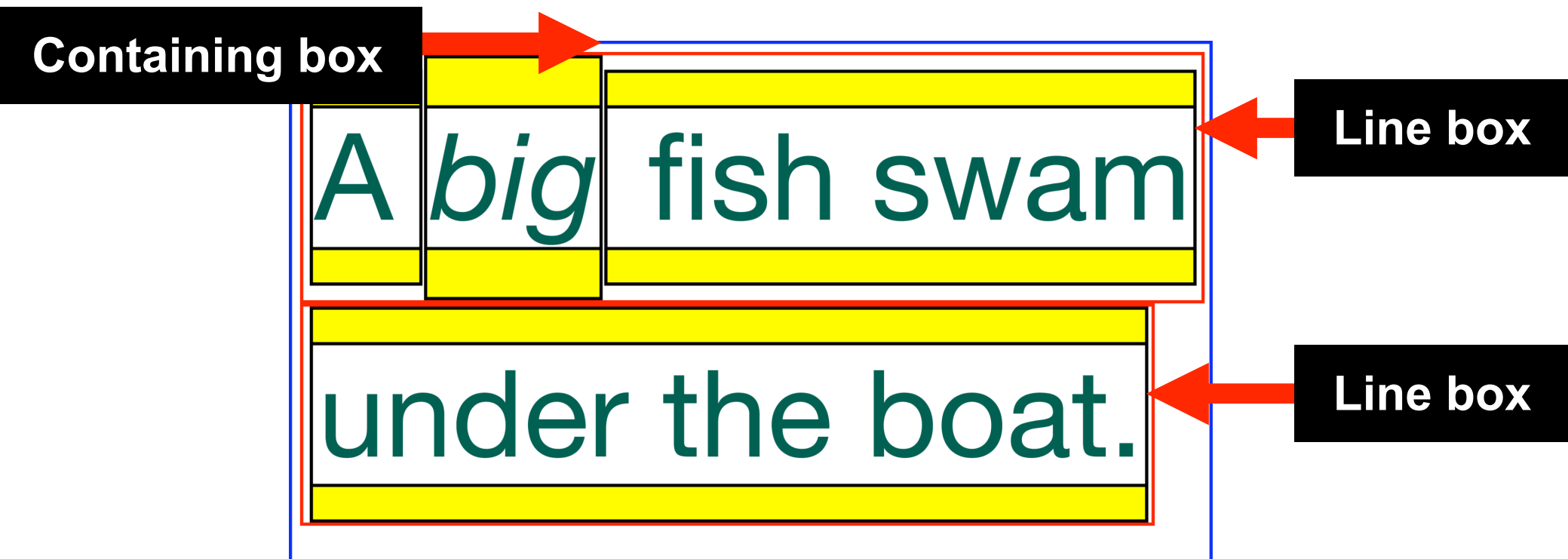


Or even the presence of a **replaced element**, such as an image.



**An inline image aligned to the baseline**

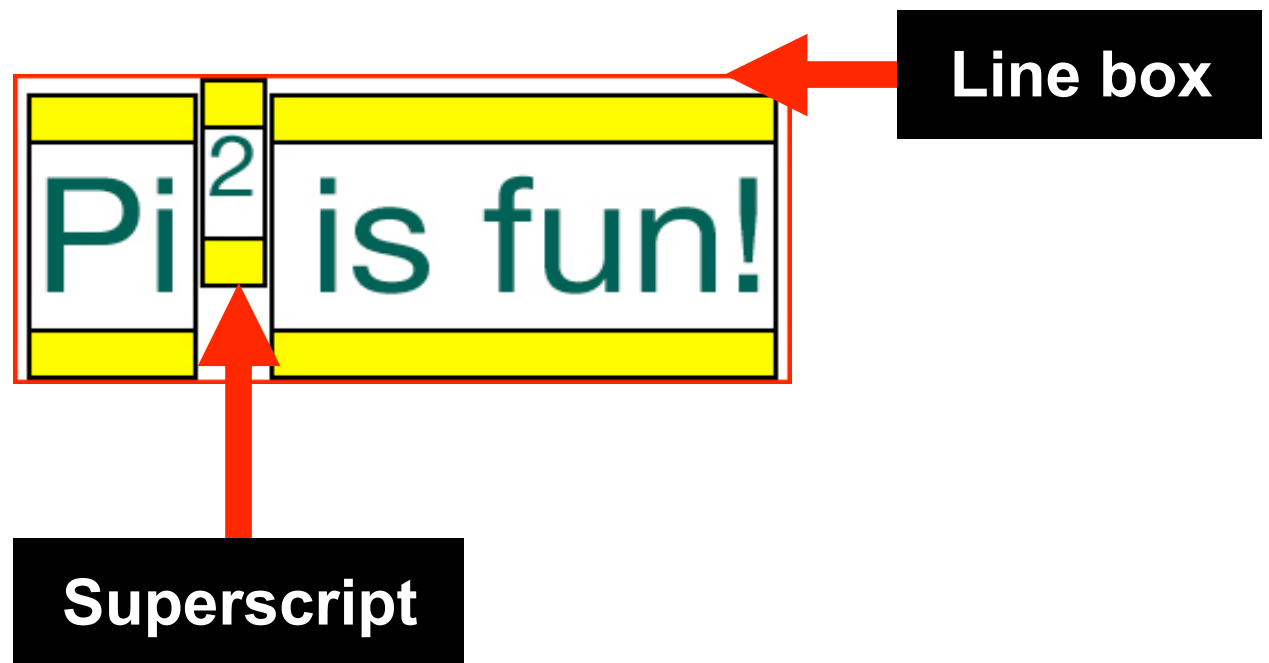
Line boxes then **stack on top of each other** within the width of the containing box.



**Some final  
notes**

# Superscript and subscript

The **superscript** and **subscript** elements can sometimes force the line box to be taller than normal.



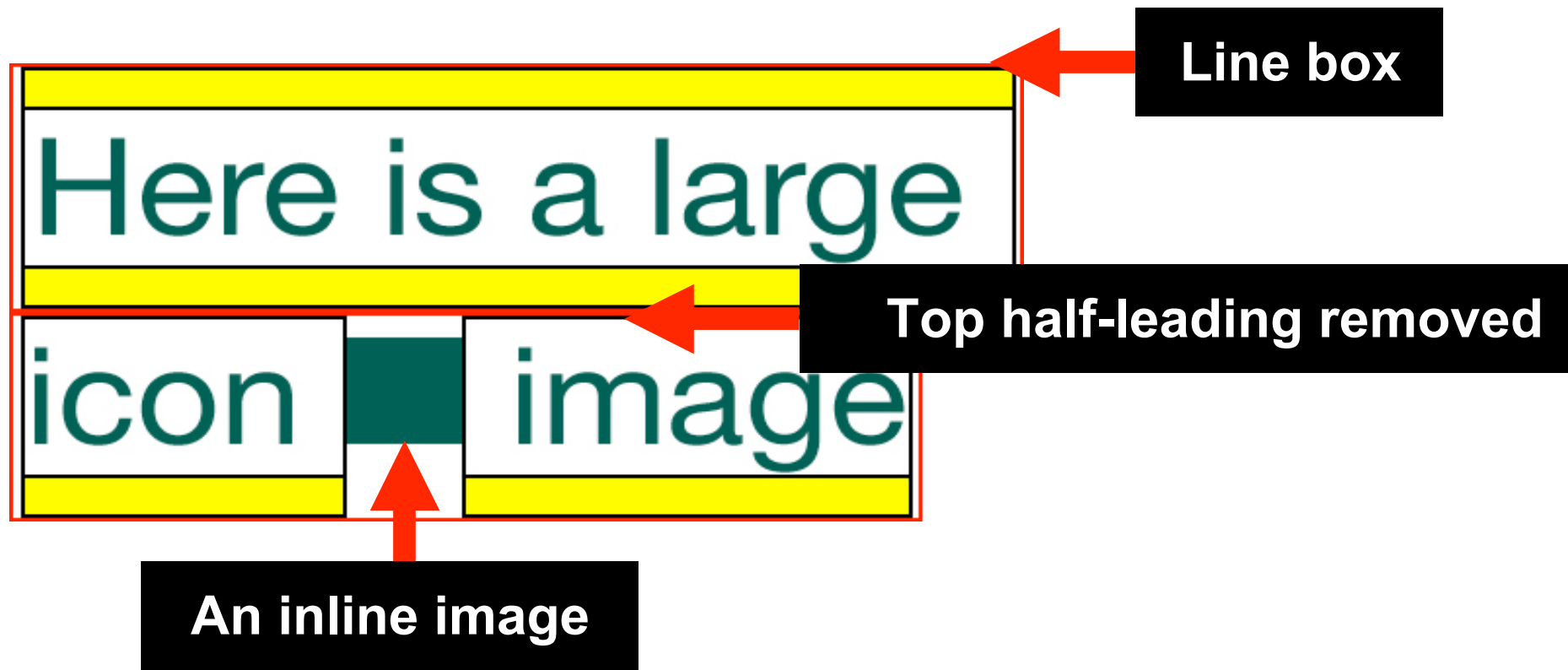
You can fix this by setting these elements to a **line-height of “0”** which will remove all half-leading from the element.

```
sup, sub  
{ line-height: 0; }
```

# **IE6, line-height and inline images**



IE5/6 **incorrectly removes** the top half-leading when an inline image is present.



This is a hard one to fix, but if needed, **margin can be added to the top of the image** to fake the half-leading. This additional margin should only be shown to

**IE5/6**

(using conditional comments)

**We're done!**