DEEP DIVE INTO LINE-HEIGHT

Why should you care about line-height?

Because line-height is **an integral part** of CSS-based layouts.

It can help to make our content **easier to read and comprehend**.

It can be used to **control the vertical rhythm** of multiple column layouts.

It can be used to **centre inline content vertically**.

But also because it is **one of the fundamentals of CSS**, like font-sizing, the cascade, inheritance and selectors.

But before we begin, let's go back in time and look at the term **“leading”**.

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**.

The quick brown fox jumps over the lazy dog

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

The term “leading” is **still used today in print typography**, where it refers to the distance between the baselines of successive lines of type.

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

However, as we will soon see, **“leading”** is still used in association with CSS line-height.

Syntax

The CSS line-height **syntax** looks like this:

https://www.w3.org/TR/CSS2/visudet.html#line-height

```
<'line-height'> = normal | <number> | <length> |
<percentage> | inherit
```

This means that line-height can be specified **using one of the following** methods:

Option 1: Line-height can be specified as **“normal”** which is the initial value. By default, browsers use between 1.0 - 1.2 line-height.

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

Option 2: Line-height can be specified as **“inherit”** which will inherit the line-height from the parent.

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

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

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

Option 4: Line-height can be specified using a **length value**.

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

A **wide range of different types** of length values can be used such as:

```
/* FONT RELATATIVE LENGTHS */


/* font size of the element */
body { line-height: 1em; }
/* x-height of the element's font */
body { line-height: 1ex; }
/* width of the "0" in the element's font */
body { line-height: 1ch; }
/* font size of the root element */
body { line-height: 1rem; }
```

```
/* VIEWPORT PERCENTAGE LENGTHS */

/* 1% of viewport's width */
body { line-height: 1vw; }
/* 1% of viewport's height */
body { line-height: 1vh; }
/* 1% of viewport's smaller dimension */
body { line-height: 1vmin; }
/* 1% of viewport's larger dimension */
body { line-height: 1vmax; }
```

```
/* ABSOLUTE LENGTHS */

/* pixels */
body { line-height: 1px; }
/* millimeters */
body { line-height: 1mm; }
/* quarter-millimeters */
body { line-height: 1q; }
/* centimeters */
body { line-height: 1cm; }
```

```
/* inches */
body { line-height: 1in; }
/* points */
body { line-height: 1pt; }
/* picas */
body { line-height: 1pc; }
```

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

```
body { line-height: 1; }
```

Number values can be specified in a range of different ways, as long as they are **positive values**.

```
/* Valid number values for line-height */
body { line-height: 3; }
body { line-height: 3.01; }
body { line-height: .30; }
body { line-height: .3; }
body { line-height: 0; }
body { line-height: 0.0; }
body { line-height: -0.0; }
body { line-height: +0.0; }
```

Shorthand

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 - separated by a slash:

<font-size>/<line-height>

https://www.w3.org/TR/CSS2/fonts.html#font-shorthand

```
<'font'> = [ [ <'font-style'> || <'font-variant'> || <'font-weight'> ]? <'font-size'> [ / <'line-height'> ]? <'font-family'> ] | caption | icon | menu | message-box | small-caption | status-bar | inherit
```

```
body {
   font: 1em/normal arial, helvetica, sans-serif;
}
```

```
body {
   font: 1em/inherit arial, helvetica, sans-serif;
}
```

```
body {
   font: 1em/20px arial, helvetica, sans-serif;
}
```

```
body {
   font: 1em/120% arial, helvetica, sans-serif;
}
```

```
body {
   font: 1em/1.2 arial, helvetica, sans-serif;
}
```

Inheritance

Some CSS properties are **inherited** - which means that their values are passed down to descendant elements.

For the line-height property, inheritance is **a little more complicated** than many other properties.

To see how line-height inheritance works, we will use four examples where the line-height is set **on the body only**.

Percentage line-height

In the following example, the line-height for the body element has been **set with 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 and the body element's font size are used to **create a calculated value** (16px x 120% = 19.2px).

This calculated value is **inherited** by descendant elements.

body	16px	120%	16 x 120% = 19.2px
h1	32px	inherits calculated value	19.2px
p	16px	inherits calculated value	19.2px
footer	12px	inherits calculated value	19.2px

This results in a line-height which is acceptable for paragraph content, but **too tight for headings** and **too open for the footer text**.

Example 1: Page title that wraps over two lines to show line-height

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde. Incidunt quam ipsam magni aperiam autem. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nostrum ex quibusdam amet eum fuga aliquam, veniam, illum dolores ea vero dolorum adipisci cum culpa veritatis quis earum tempora magni eveniet!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsam modi commodi quo, ea id, rerum praesentium rem accusamus quia deleniti quam saepe nihil nobis iure quaerat placeat reprehenderit maiores tenetur. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iste omnis, obcaecati quis architecto id accusantium iusto dolore. Tempore praesentium culpa quis, aperiam sed id nisi illo, veritatis totam veniam cumque.

Length line-height

In the following example, the line-height for the body element has been **set with 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** directly by descendant elements.

body	16px	20px	20px
h1	32px	inherits 20px	20px
p	16px	inherits 20px	20px
footer	12px	inherits 20px	20px

Again, this results in a line-height which is acceptable for paragraph content, but **too tight for headings** and **too open for the footer text**.

Example 2: Page title that wraps over two lines to show line-height

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde. Incidunt quam ipsam magni aperiam autem. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nostrum ex quibusdam amet eum fuga aliquam, veniam, illum dolores ea vero dolorum adipisci cum culpa veritatis quis earum tempora magni eveniet!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsam modi commodi quo, ea id, rerum praesentium rem accusamus quia deleniti quam saepe nihil nobis iure quaerat placeat reprehenderit maiores tenetur. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iste omnis, obcaecati quis architecto id accusantium iusto dolore. Tempore praesentium culpa quis, aperiam sed id nisi illo, veritatis totam veniam cumque.

Normal line-height

In the following example, the line-height for the body element has been **set with the “normal” value**.

```
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.

body	16px	normal	16 x 1.2 (approx.) = 19.2px (approx.)
h1	32px	normal	32 x 1.2 (approx.) = 38.4px (approx.)
p	16px	normal	16 x 1.2 (approx.) = 19.2px (approx.)
footer	12px	normal	12 x 1.2 (approx.) = 14.4px (approx.)

This method scales the line-height to suit each element. This results in a line-height which is **acceptable** for the paragraph, heading and footer content.

Example 3: Page title that wraps over two lines to show line-height

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde. Incidunt quam ipsam magni aperiam autem. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nostrum ex quibusdam amet eum fuga aliquam, veniam, illum dolores ea vero dolorum adipisci cum culpa veritatis quis earum tempora magni eveniet!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsam modi commodi quo, ea id, rerum praesentium rem accusamus quia deleniti quam saepe nihil nobis iure quaerat placeat reprehenderit maiores tenetur. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iste omnis, obcaecati quis architecto id accusantium iusto dolore. Tempore praesentium culpa quis, aperiam sed id nisi illo, veritatis totam veniam cumque.

Number line-height

In the following example, the line-height for the body element has been **set with a number value** (1.2).

```
body {
    font-size: 16px;
    line-height: 1.2;
}

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

In this case, the **factor** (1.2) rather than a calculated value is inherited by descendant elements.

body	16px	1.2	16 x 1.2 = 19.2px
h1	32px	factor of 1.2	32 x 1.2 = 38.4px
p	16px	factor of 1.2	16 x 1.2 = 19.2px
footer	12px	factor of 1.2	12 x 1.2 = 14.4px

Like the normal value, this method scales to suit each element and results in a line-height which is **acceptable** for the paragraph, heading and footer content.

Example 4: Page title that wraps over two lines to show line-height

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde. Incidunt quam ipsam magni aperiam autem. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nostrum ex quibusdam amet eum fuga aliquam, veniam, illum dolores ea vero dolorum adipisci cum culpa veritatis quis earum tempora magni eveniet!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsam modi commodi quo, ea id, rerum praesentium rem accusamus quia deleniti quam saepe nihil nobis iure quaerat placeat reprehenderit maiores tenetur. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iste omnis, obcaecati quis architecto id accusantium iusto dolore. Tempore praesentium culpa quis, aperiam sed id nisi illo, veritatis totam veniam cumque.

Which method is best?

Number values are the preferred method as they work well when inherited.

Unlike the “normal” keyword, number values allows us to set **specific line-heights for different types of elements**.

Inline boxes and line-height

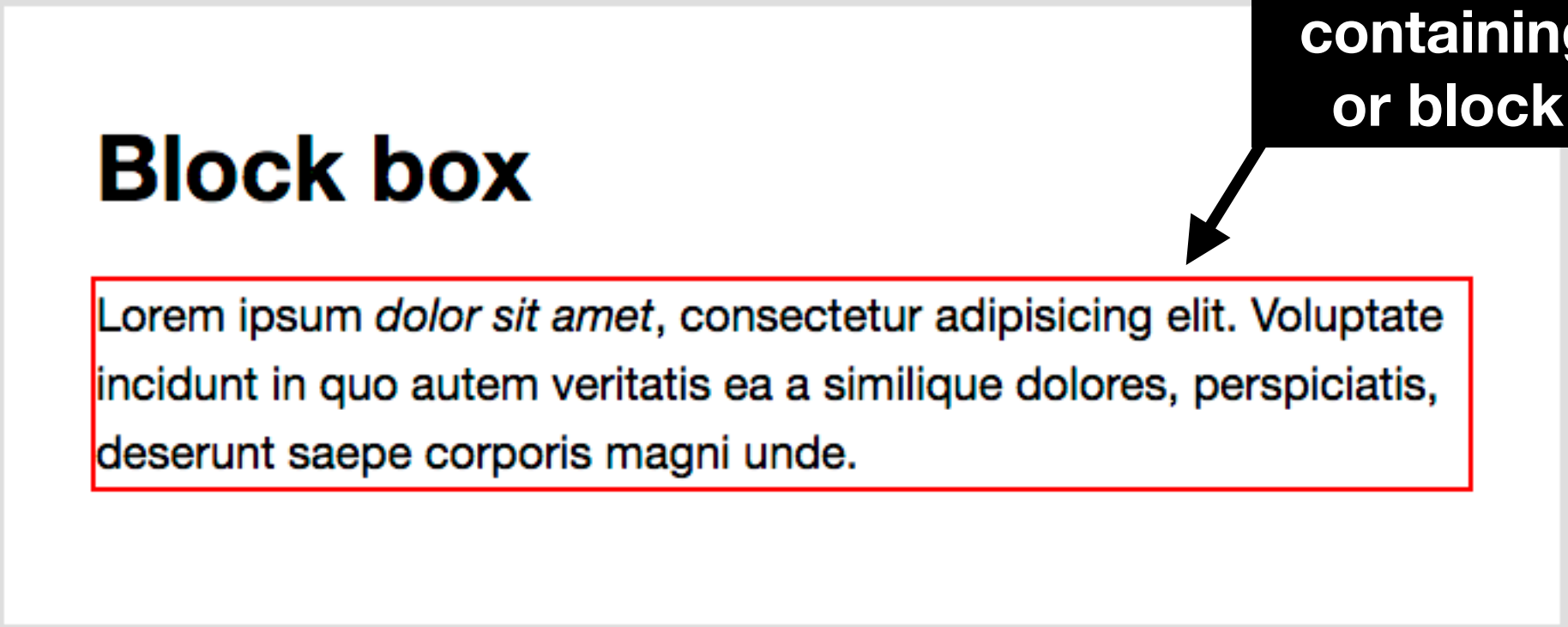
Types of boxes

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

If we look at a **simple paragraph of text**, there are a range of possible boxes that are relevant.

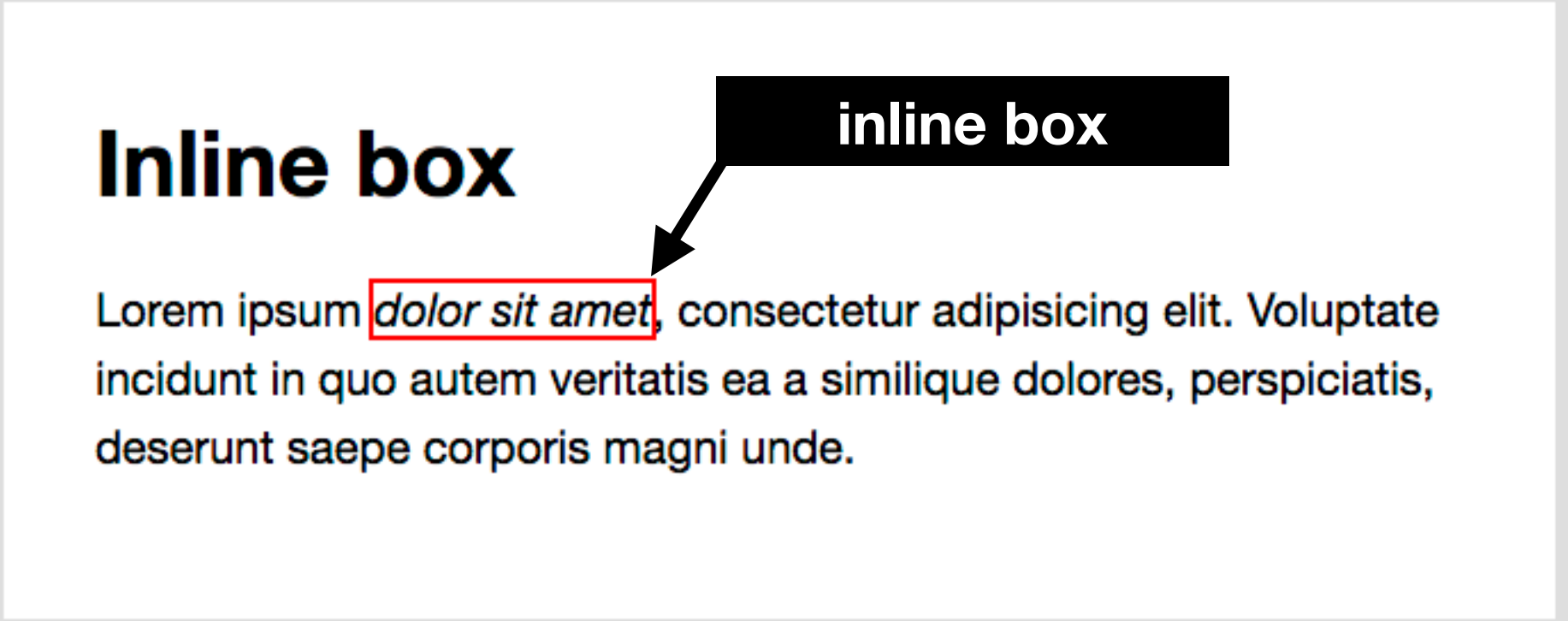
The paragraph is referred to as a **containing box** in this case - as it contains other boxes.

The paragraph can also be referred to as a **block box** as it displays as a block - with whitespace before and after.


containing box
or block box
Block box
Lorem ipsum *dolor sit amet*, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde.

Inside the paragraph, there may be any number of **inline boxes**. These are boxes that do not form new lines like block boxes.

In our example, the **italic element** is an inline box.


inline box
Inline box
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde.

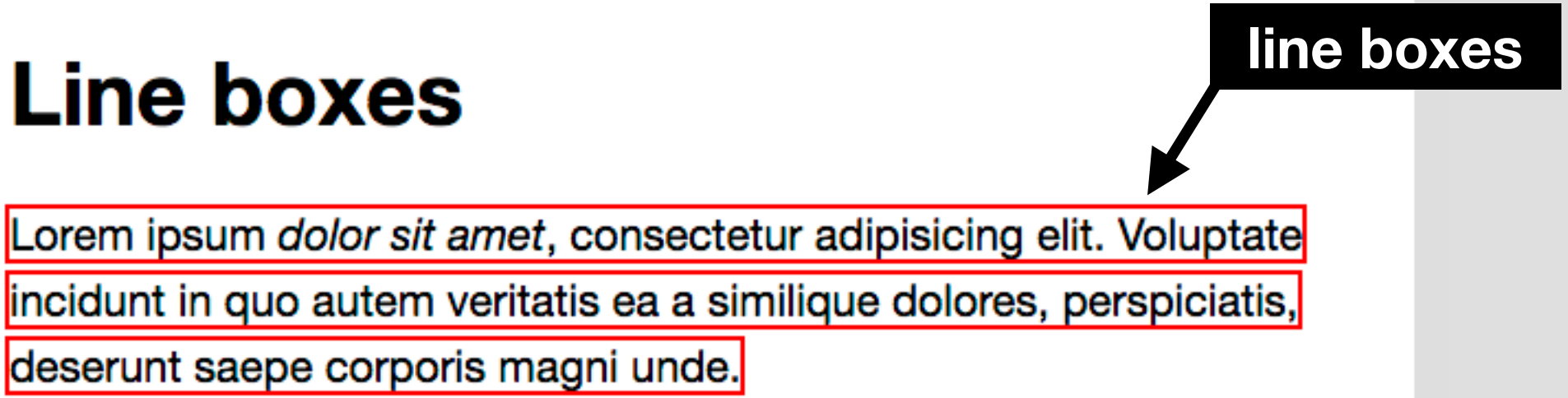
Other inline boxes without specific markup are referred to as **anonymous inline boxes**.

Anonymous boxes

Lorem ipsum *dolor sit amet*, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde.

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

Line boxes



Lorem ipsum *dolor sit amet*, consectetur adipisicing elit. Voluptate incidunt in quo autem veritatis ea a similique dolores, perspiciatis, deserunt saepe corporis magni unde.

We'll be looking at line boxes **in more detail** later.

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


inline box
content area
ÙAbcdefghijkl

How line-height affects inline boxes

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

Step 1.

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

line-height - font-size = leading

20px - 16px = 4px

Step 2.

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

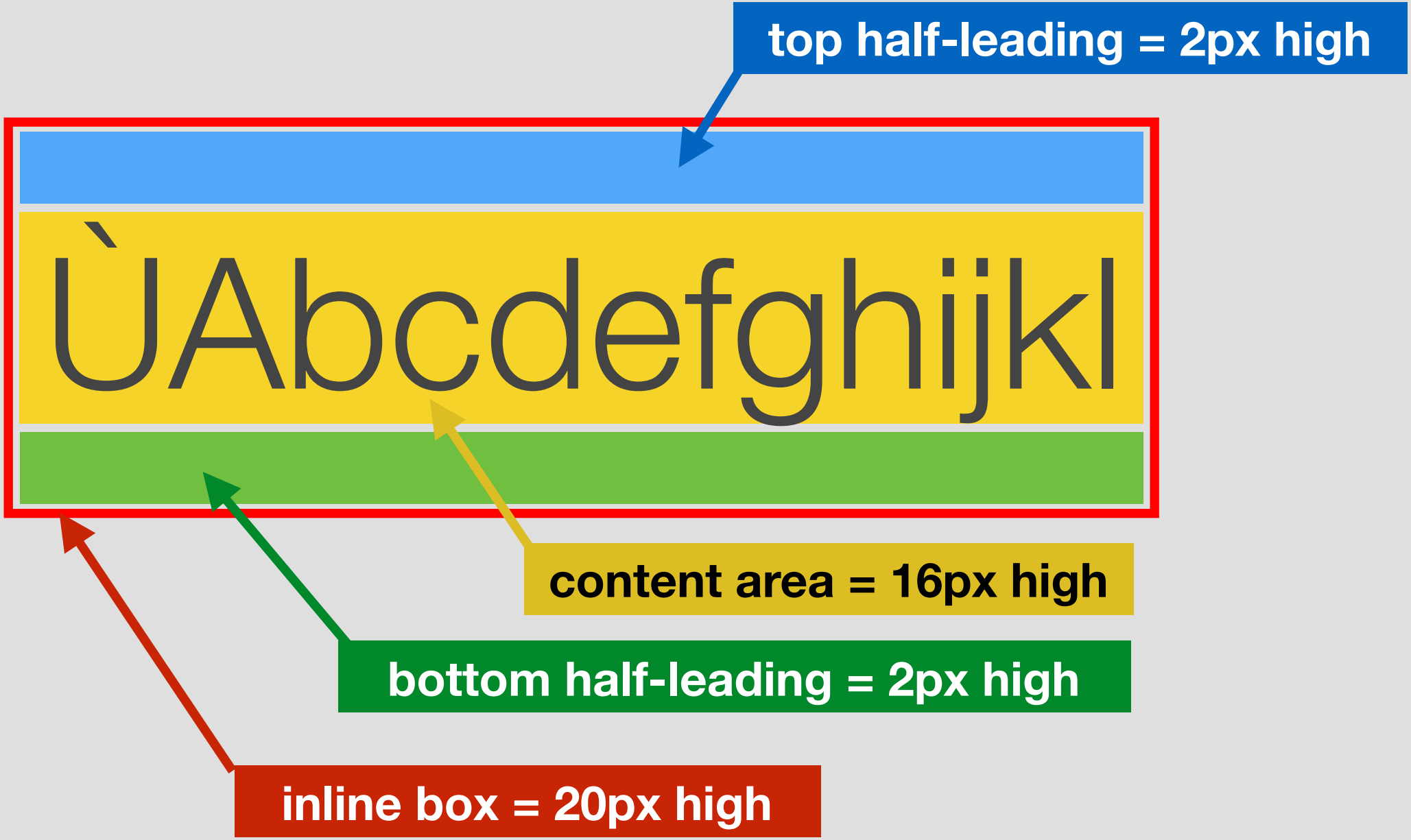
leading / 2 = half-leading

4px / 2 = 2px (half-leading)

Step 3.

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

Top half-leading: 2px
Content area: 16px
Bottom half-leading: 2px
Total height: 20px


top half-leading = 2px high
ÙAbcdefghijkl
content area = 16px high
bottom half-leading = 2px high
inline box = 20px high

However, if the line-height is smaller than the font size, the inline box will be **the height of the line height only**.

This means the content area **will poke out the top and bottom** of the inline box.

line-height - font-size = leading

12px - 16px = -4px (leading)

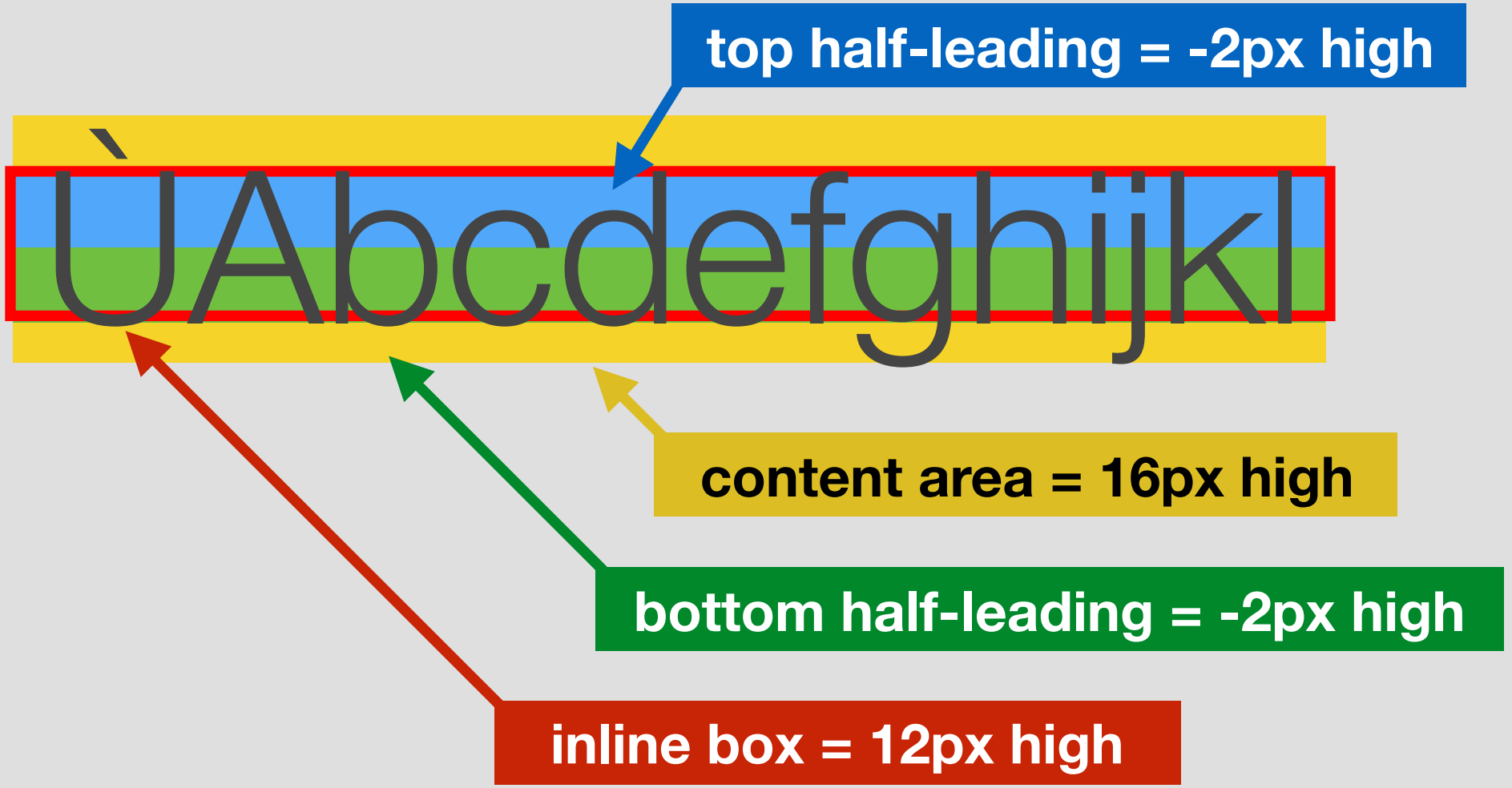
-4px / 2 = -2px (half-leading)

Top half-leading: -2px

Content area: 16px

Bottom half-leading: -2px

Total height: 12px


top half-leading = -2px high
ÙAbcdefghijkl
content area = 16px high
bottom half-leading = -2px high
inline box = 12px high

Finally, you can also **set the line-height to “0”** which means the inline element will have no height at all.

line-height - font-size = leading

0 - 16px = -16px (leading)

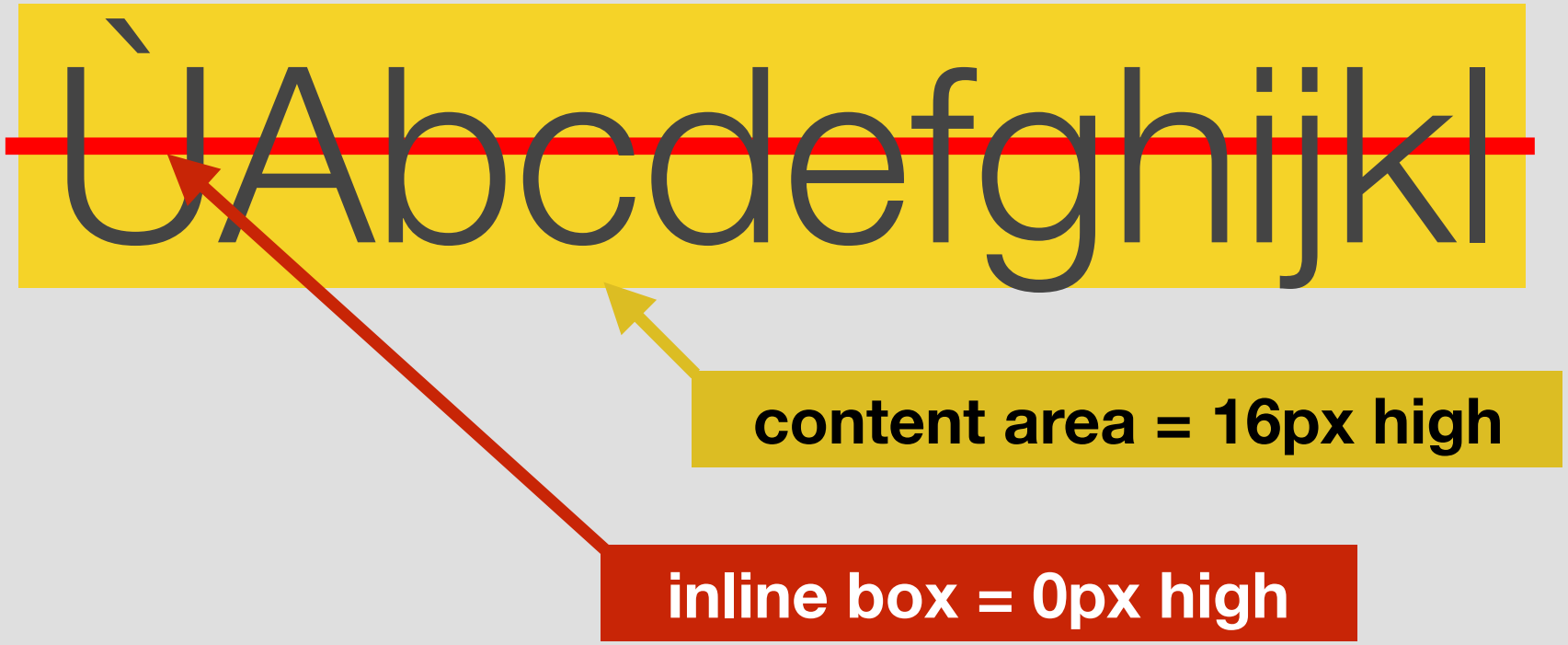
-16px / 2 = -8px (half-leading)

Top half-leading: -8px

Content area: 16px

Bottom half-leading: -8px

Total height: 0


ÙAbcdefghijkl
content area = 16px high
inline box = 0px high

Using line-height to vertically align content

Line-height can be used to vertically align content inside a parent container **as long as the content is one line only**.

For example:

Let’s take a small piece of text with font-size 16px and we want it to to be vertically aligned inside a parent that is 200px high.

We can **set the line-height to 200px** and this text will automatically sit in the vertical centre of the parent.

line-height - font-size = leading

200px - 16px = 184px (leading)

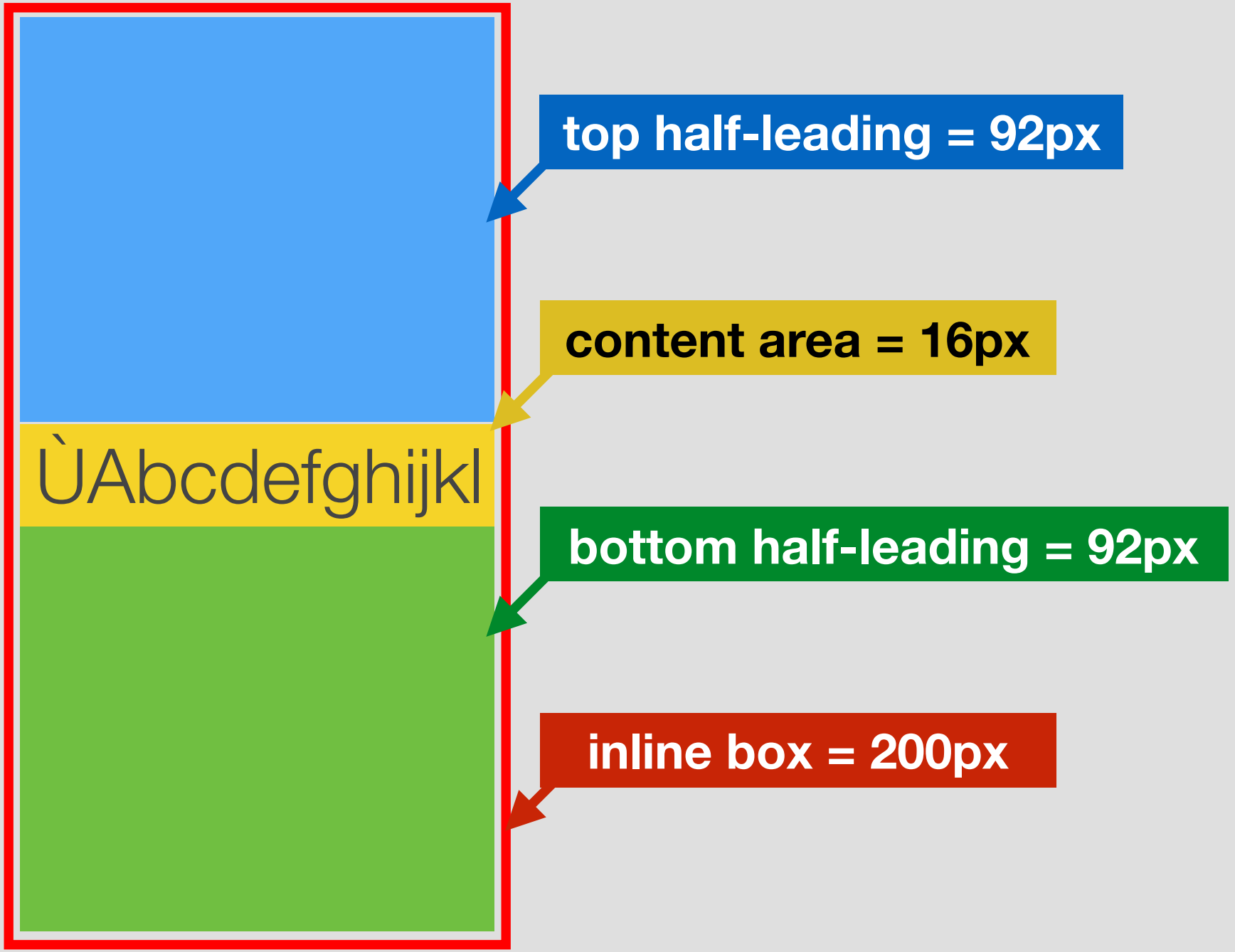
184px / 2 = 92px (half-leading)

Top half-leading: 92px

Content area: 16px

Bottom half-leading: 92px

Total height: 200px

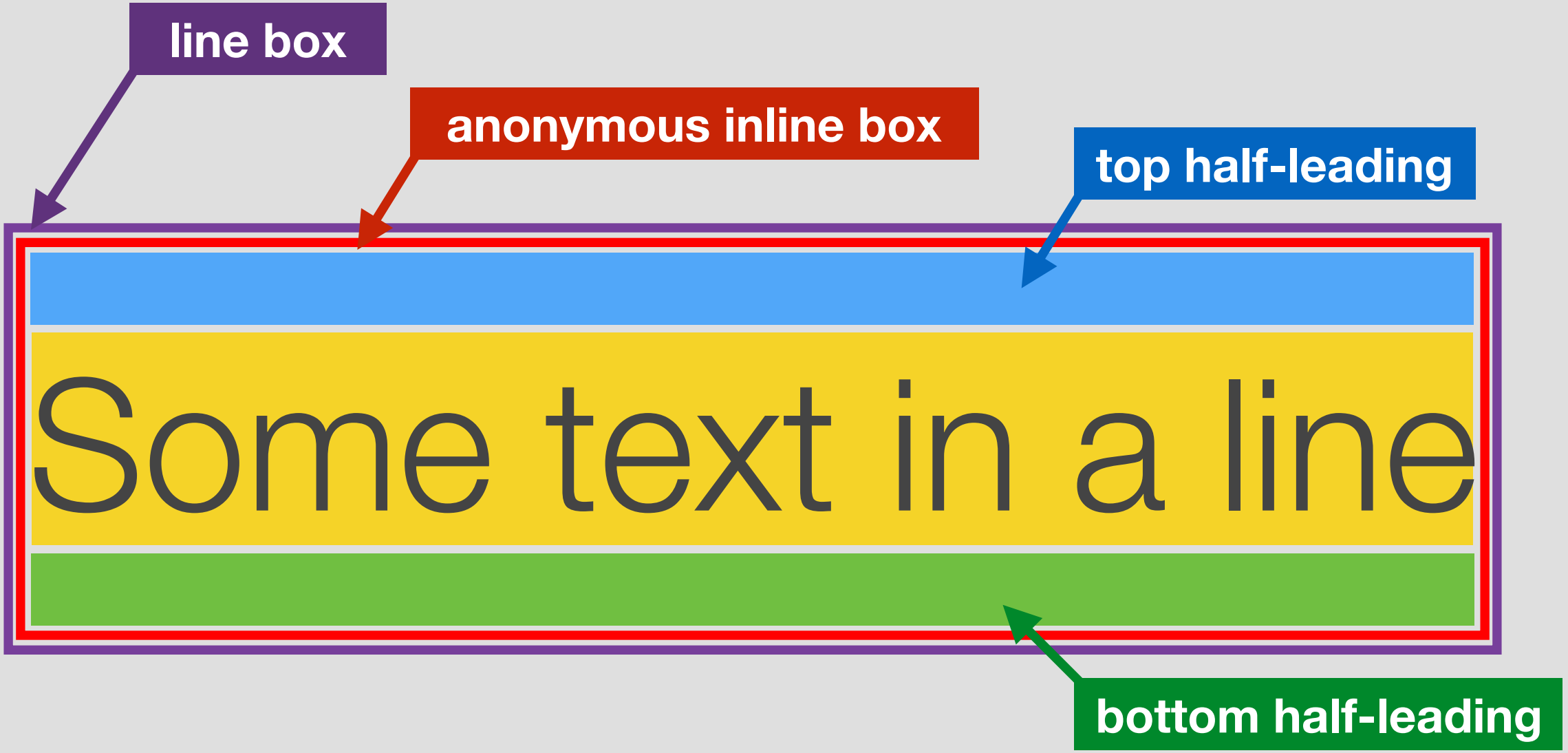

top half-leading = 92px
content area = 16px
ÙAbcdefghijkl
bottom half-leading = 92px
inline box = 200px

Line boxes

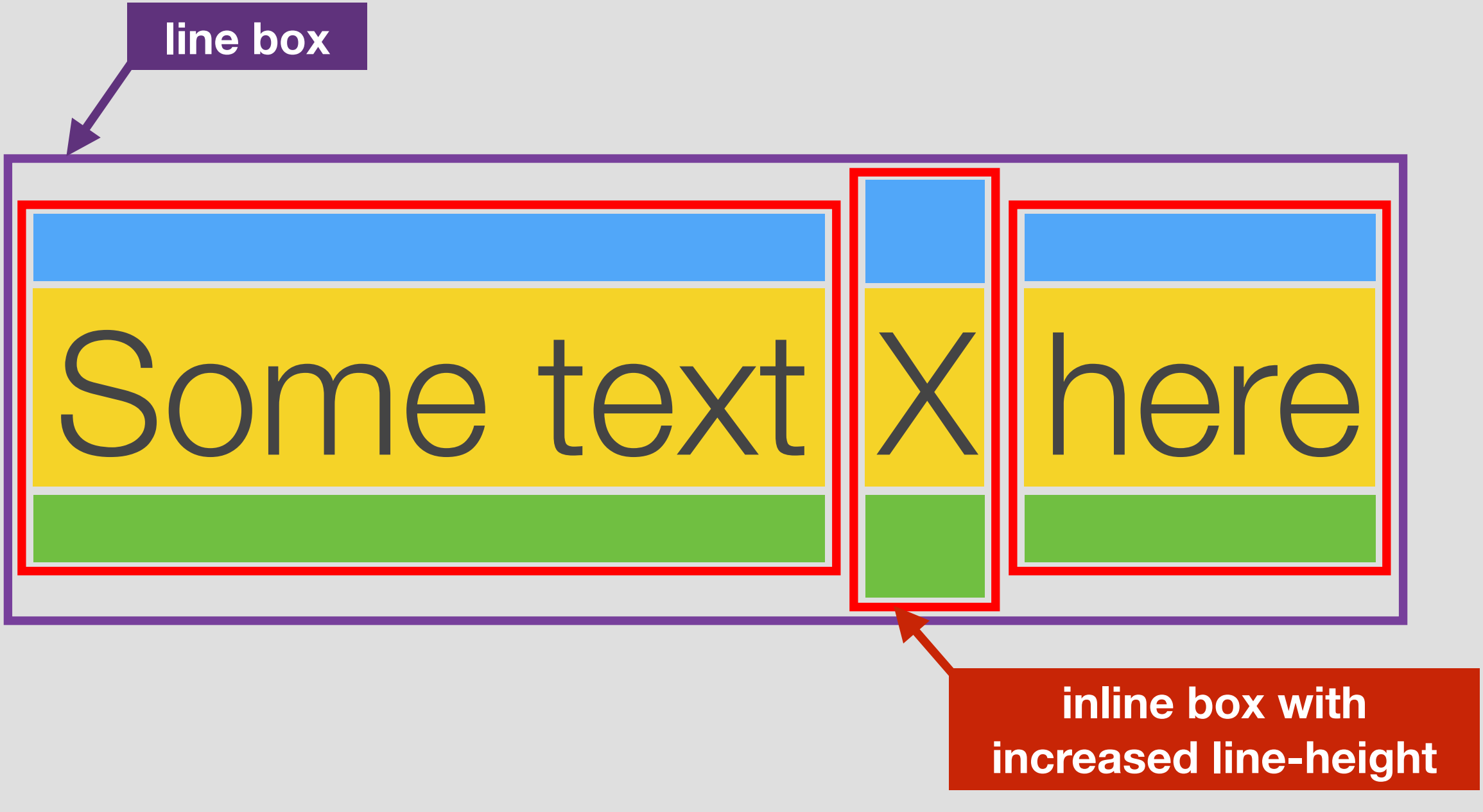
How inline boxes affect line boxes

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

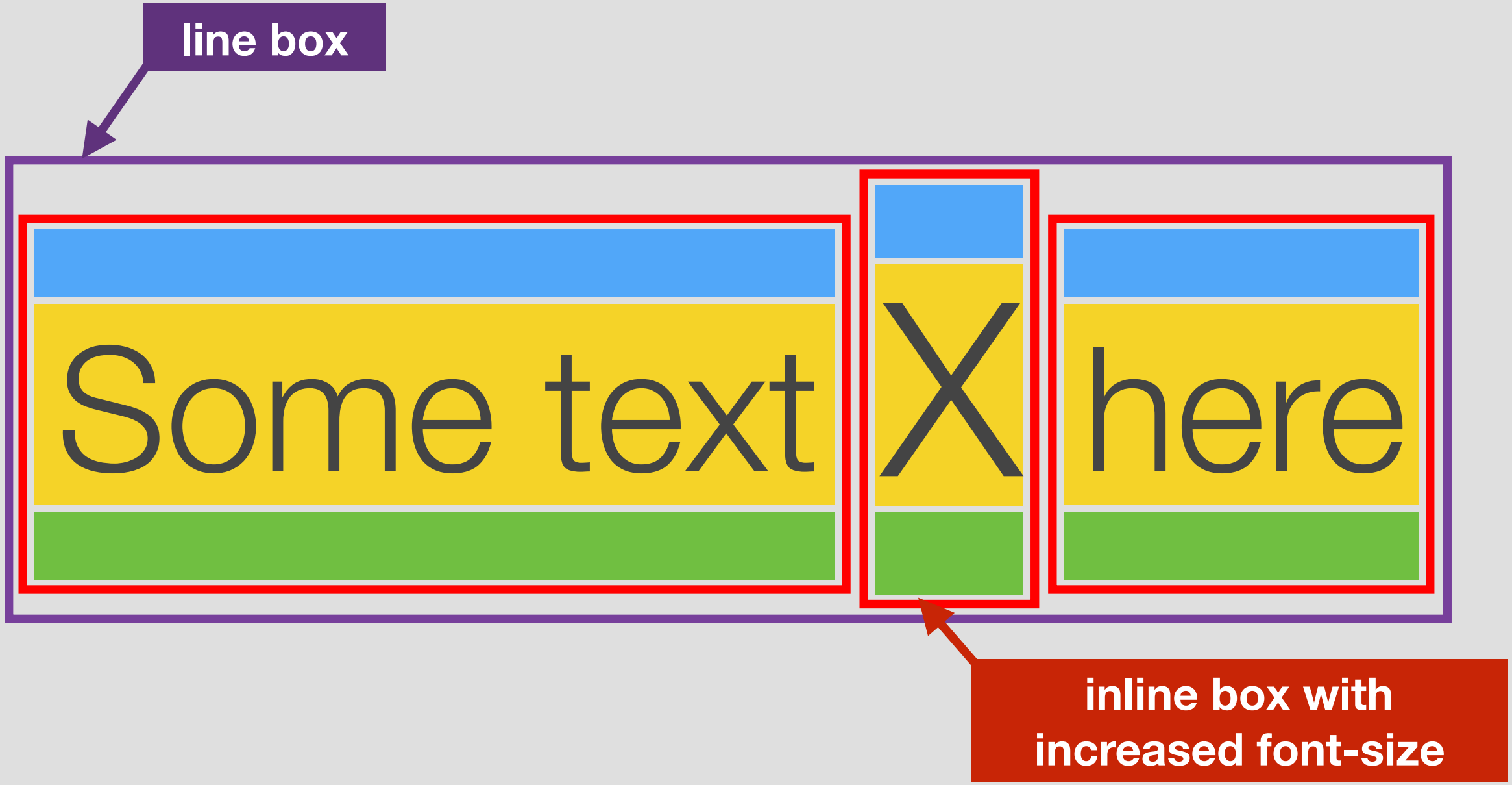
The tallest inline box could be an **anonymous inline box**.


line box
anonymous inline box
top half-leading
Some text in a line
bottom half-leading

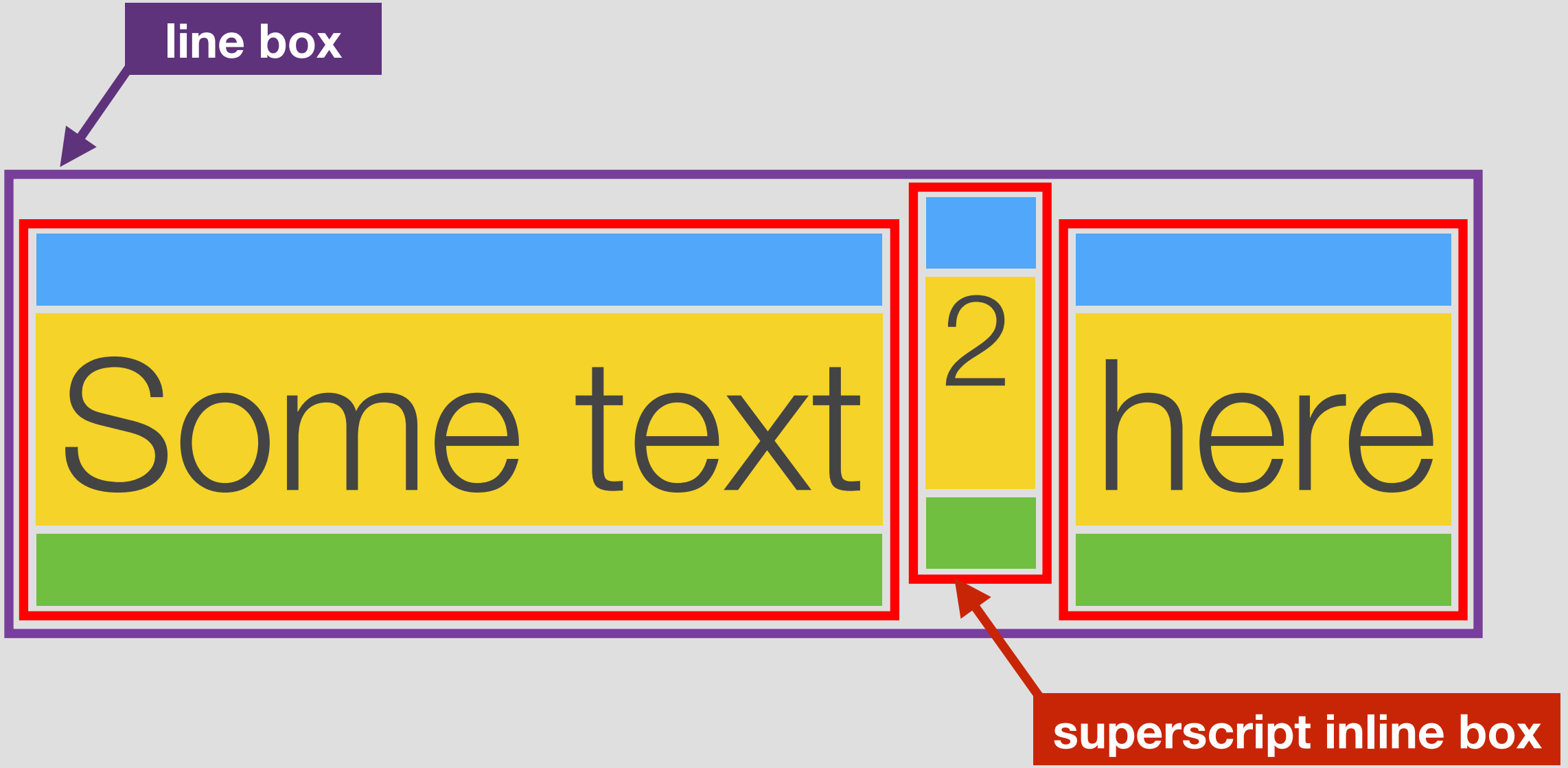
It could be an inline box **with increased line-height** (which makes this inline box taller than other inline boxes).


line box
Some text X here
inline box with
increased line-height

It could be an inline box **with a larger font-size** (which makes this inline box taller than other inline boxes).


line box
Some text
X
here
inline box with
increased font-size

Depending on the browser, it could be the presence of a **superscript or subscript**. (Some browsers render superscript elements in a way that affects line boxes)

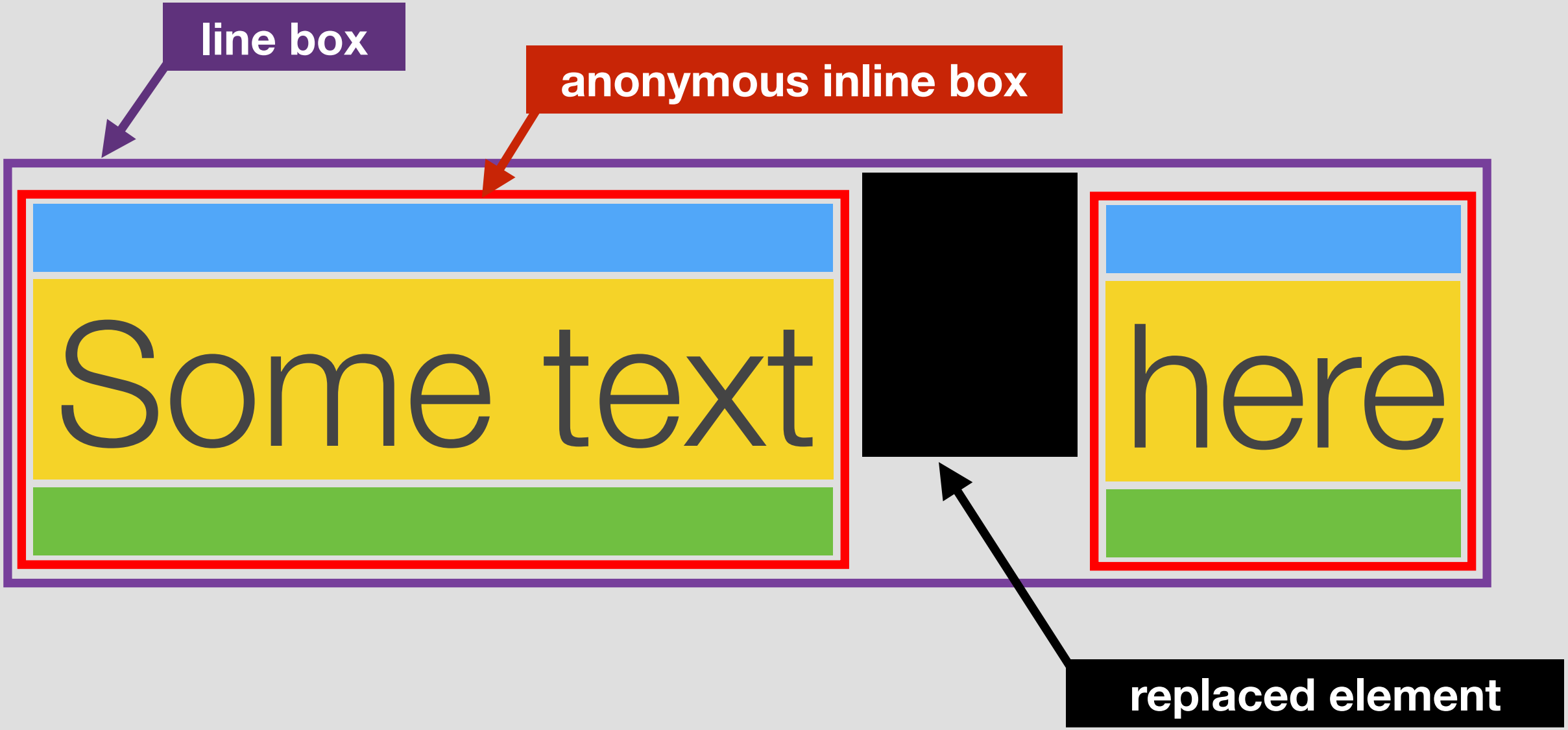

line box
Some text
2
here
superscript inline box

Side note:

We can solve this by setting the sup and sub elements with line-height set to “0”.

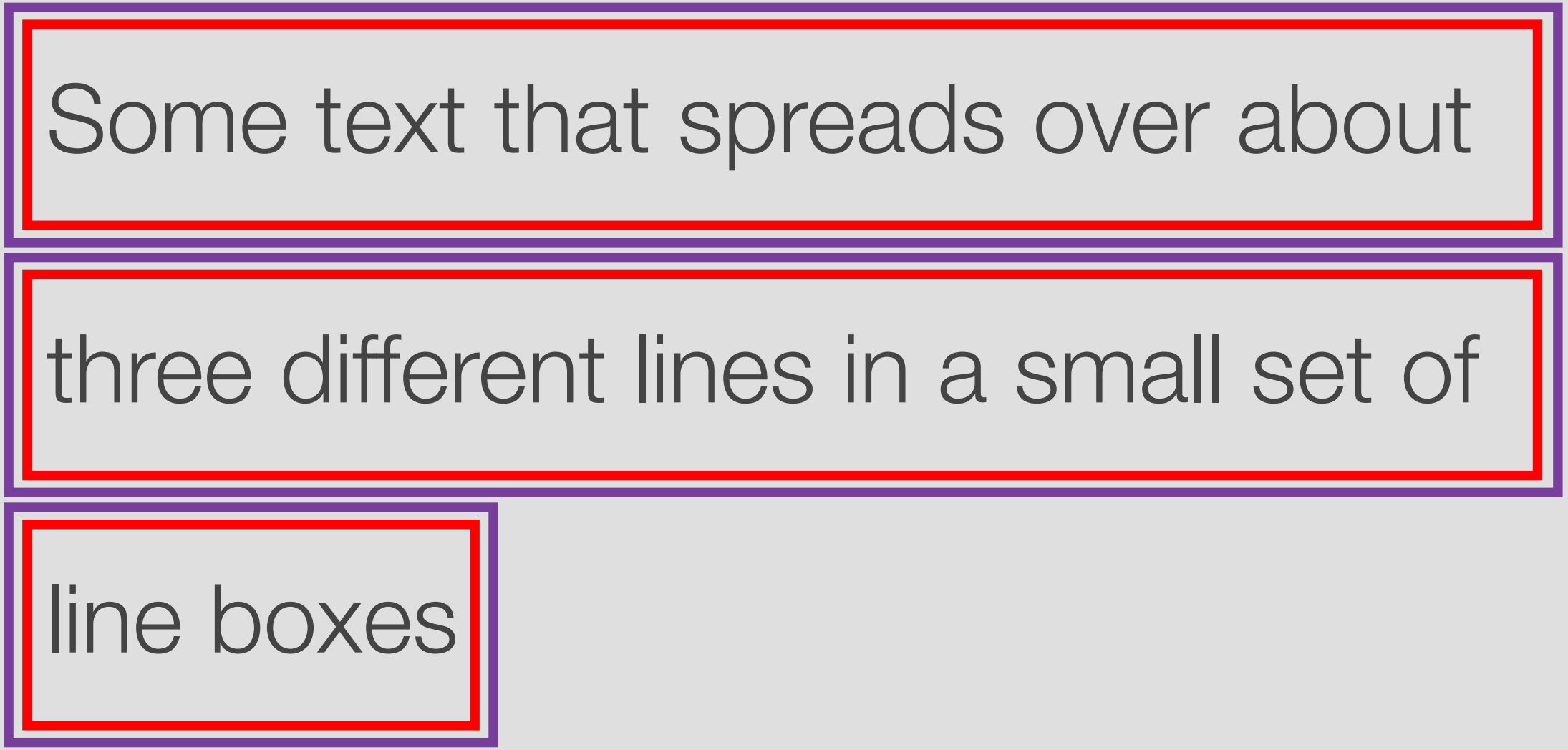
```
sub,
sup {
 line-height: 0;
}
```

Or even the presence of a **replaced element** that is larger than the text around it, such as an image.

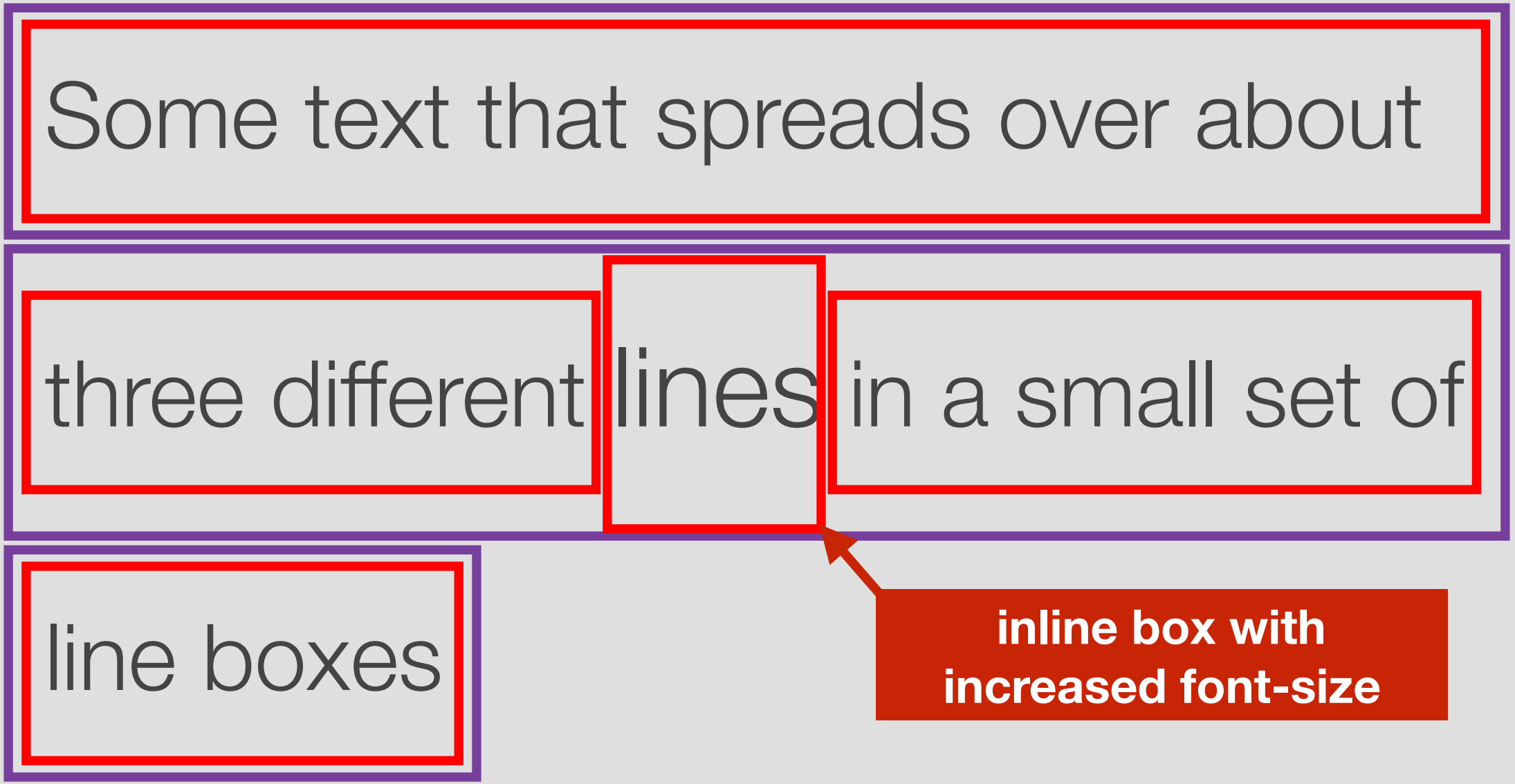

line box
anonymous inline box
Some text
here
replaced element

Inline boxes poking out of line boxes?

Line boxes are **laid out one after the other**, spreading to the width of the containing box.


Some text that spreads over about
three different lines in a small set of
line boxes

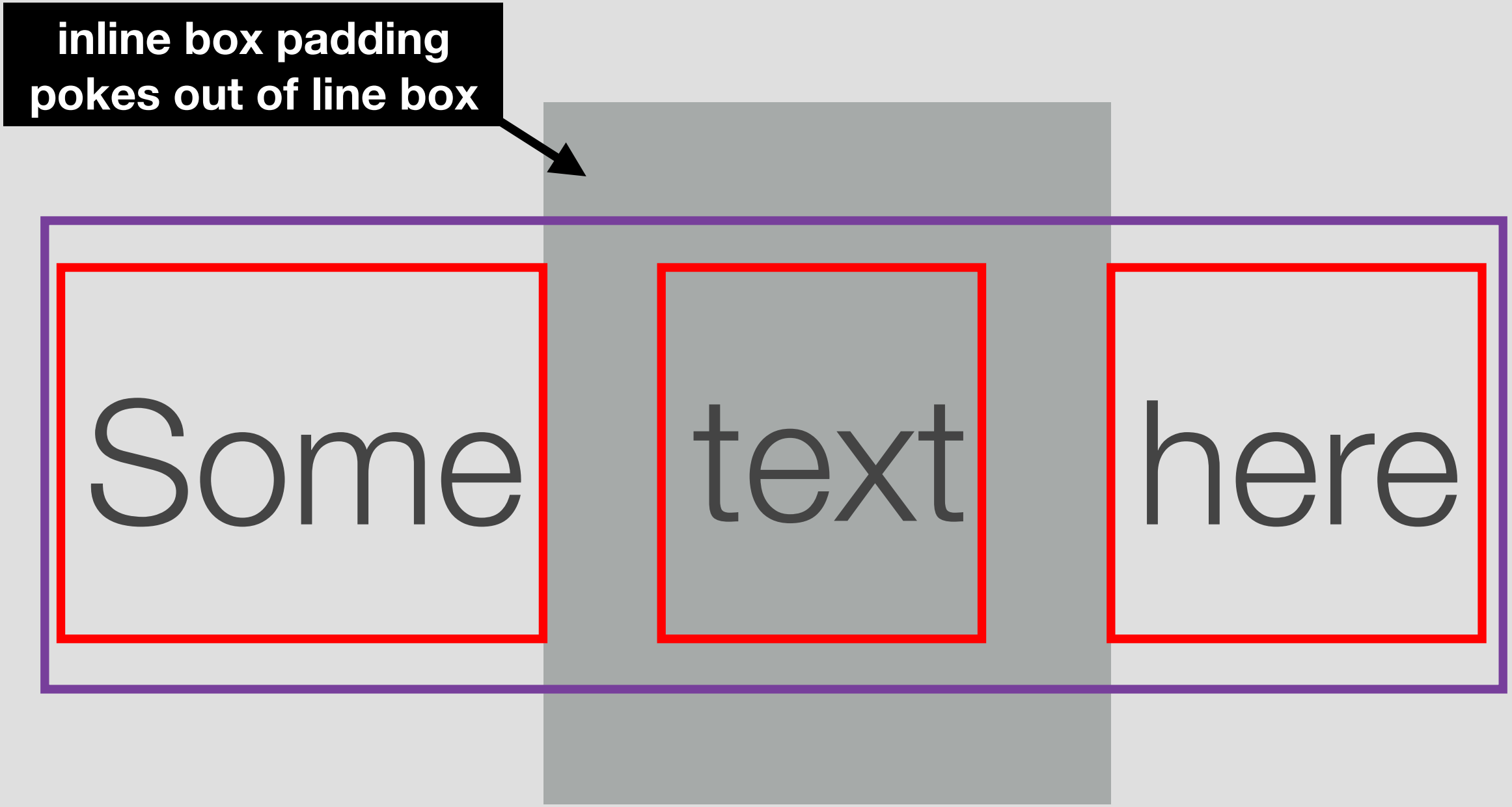
As we have seen, line boxes will **grow to the height of inline boxes** inside.


Some text that spreads over about
three different lines in a small set of
line boxes
inline box with
increased font-size

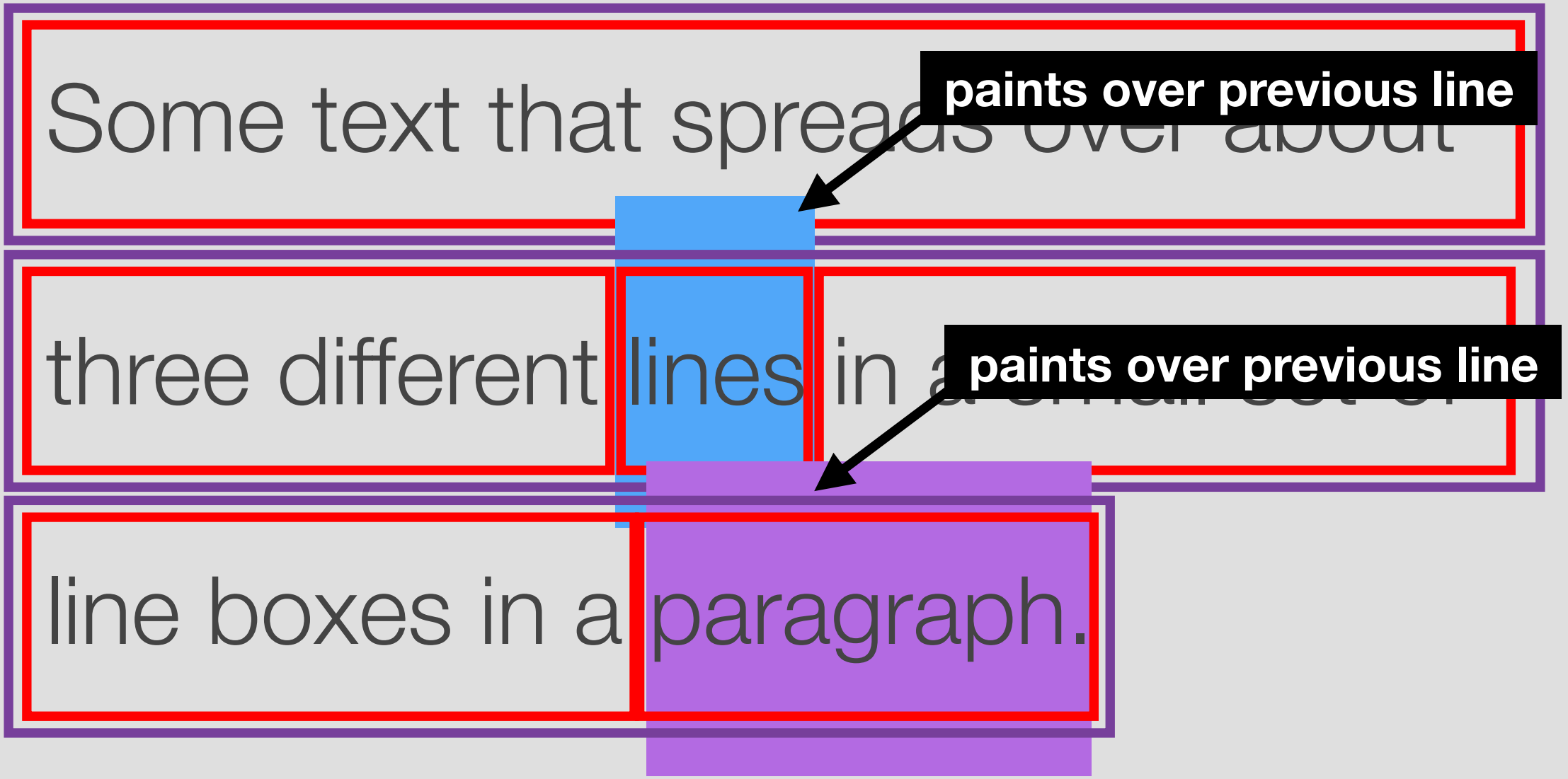
However, there are times when aspects of inline boxes **will poke out of the top and/or bottom of line boxes**.

An example is an inline box with **padding, margin or border**.

Because **inline boxes cannot be given height**, padding, margin and border can be present above and below the element, but they do not affect the line box.


inline box padding
pokes out of line box
Some
text
here

Browsers will render the line boxes in document order. So, borders on subsequent lines **may paint over the borders and text of previous lines**.


Some text that spreads over about
paints over previous line
three different lines in a small box of
paints over previous line
line boxes in a paragraph.

Ideal line-height?

The concept of **“ideal line-height”** depends on a wide range of factors including the element, the type of content, and the column width.

For this reason, I’m only going to touch on suggested line-hight for **a small set of elements**, in specific circumstances.

Research has shown that line-height that is too small **can make content harder to read** as users have to work harder to move from line to line.

Similarly, line-height that is too large can force **users eyes have to travel further** to read content, which can become tiring.

The **WCAG 2.0 guidelines** state that: “line spacing is at least space-and-a-half within paragraphs”.

https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-visual-presentation.html

This means that general content such as paragraphs **should be set to a line-height of 1.5**.

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

The same rules should apply to **ordered and unordered lists** which have a lot of content inside each list item.

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

However, content-heavy list items could then bleed into each other, so you might want to **add additional space after list items**.

```
li {
  line-height: 1.5;
  margin-botton: .5em;
}
```

On the other hand, headings often **look strange when there is too much line-height**, so I generally set headings to 1.1 or 1.2 - much tighter than paragraphs.

```
h1,h2,h3,h4,h5,h6 {
  line-height: 1.1;
}
```

Responsive line-height?

Several years ago, I was involved in user testing a content-heavy website where we wanted the content to be **“as readable as possible”** at all screen sizes.

We **tested a range of different factors** including font-family, font-size, color and line-height.

As well as testing specific tasks, and recording times for these tasks, we also **asked users directly about these factors** after each test was concluded.

It turned out that users were reasonably comfortable with paragraphs and lists that were **anywhere from 1.4 - 1.6 at large and mid screen sizes**.

However, users were more comfortable with **slightly less line-height (between 1.3 - 1.5) at small screen size**, as the lines were much shorter.

As long as line-height is set using number values, it is **very easy to tweak line-heights for the different screen sizes**.

```
p,li { line-height: 1.4; }

@media(min-width: 320px) {
  p,li { line-height: 1.5; }
}
```

Baseline grids

“Vertical rhythm” in multiple column layouts is where you to **establish a baseline grid** that aligns across multiple columns

Heading Level 1

Lorem ipsum dolor sit amet, has id discere platonem ocurreret, ut duo audire senserit maiestatis, per ex assum instructior. Quo assum facete deleniti ne. Ei pri nisl voluptatum.

Amet laboramus sententiae te usu. Et cum quis amet veniam, mel case omittam id, ei vis atqui.

Te munere audire sit, cu sea vidisse probatus, munere molestie voluptatibus id ius. Paulo intellegebat has id. Nam no graecis fastidii perfecto, nec ut atomorum salutatus. Usu at rebum zril principes, hinc esse id cum. Nam ridens ullamcorper et.

No numquam interpretaris duo. Ei pri nullam sanctus, sea ornatus probatus pertinax an. Saepe persius delectus cum eu, ea vim numquam electram aliquando. Et eos erat dolorem abhorreant, quem stet vidit te per. Inermis nonumes mei no, et has ornatus antiopam cotidieque.

Subheading

Ut sit paulo consulatu, mea nonumy appareat conceptam et. Dicat consulatu hendrerit duo at, ei sea tation antiopam accommodare. Eam ad perfecto imperdiet, novum solet eu mei. Eu eam aliquam consulatu instructior, vel vocibus oportere intellegebat ex.

Using Desktop Publishing software, this can easily be achieved by simply checking a **“snap to baseline grids”** button.

However, it’s **much harder using CSS**. Here are some steps to achieve a simple baseline grid.

Step 1.

Set a line-height which will become the baseline grid.

16px / 24px

```
$baseline: 24px;
```

Step 2.

Set headings, paragraphs and lists with this line-height.

```
$baseline: 24px;

h1 {
  line-height: $basefont*2;
}

p {
  line-height: $baseline;
}
```

Step 3.

Turn off margin-top on all of these elements, and set the margin-bottom to match the line-height. This will set consistent one full line gaps after each element.

Heading Level 1

Lorem ipsum dolor sit amet, has id discere platonem ocurreret, ut duo audire senserit maiestatis, per ex assum instructior. Quo assum facete deleniti ne. Ei pri nisl voluptatum.

Amet laboramus sententiae te usu. Et cum quis amet veniam, mel case omittam id, ei vis atqui.

Te munere audire sit, cu sea vidisse probatus, munere molestie voluptatibus id ius. Paulo intellegebat has id. Nam no graecis fastidii perfecto, nec ut atomorum salutatus. Usu at rebum zril principes, hinc esse id cum. Nam ridens ullamcorper et.

No numquam interpretaris duo. Ei pri nullam sanctus, sea ornatus probatus pertinax an. Saepe persius delectus cum eu, ea vim numquam electram aliquando. Et eos erat dolorem abhorreant, quem stet vidit te per. Inermis nonumes mei no, et has ornatus antiopam cotidieque.

Subheading

Ut sit paulo consulatu, mea nonumy appareat conceptam et. Dicat consulatu hendrerit duo at, ei sea tation antiopam accommodare. Eam ad perfecto imperdiet, novum solet eu mei. Eu eam aliquam consulatu instructior, vel vocibus oportere intellegebat ex.

```
$baseline: 24px;

h1 {
  line-height: $basefont*2;
  margin-bottom: $baseline;
}
p {
  line-height: $baseline;
  margin-bottom: $baseline;
}
```

Step 4.

You may need to set font-sizes to the same ratios.

```
$basefont: 16px;
$baseline: 24px;

h1 {
  line-height: $basefont*2;
  margin-bottom: $baseline;
}
p {
  font-size: $basefont;
  line-height: $baseline;
  margin-bottom: $baseline;
}
```

However, **nothing is ever that simple**. As soon as you introduce pull-quotes, different headings, special content and images, things can quickly break down.

https://www.smashingmagazine.com/2012/12/css-baseline-the-good-the-bad-and-the-ugly/

http://webdesign.tutsplus.com/articles/setting-web-type-to-a-baseline-grid--webdesign-3414

http://alistapart.com/article/settingtypeontheweb

http://stephanecurzi.me/baselinecss.2009/grid.html

Conclusion

Line-height is **everywhere in our layouts**. It's in our headings, in our nav items, our form controls, our buttons.

Understanding how line-height works will **make your job a lot easier**.

We’re done.

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley