$math

Contents

Introduction

This manual is not a systematic discussion about math in ConTEXt but more a collection of wrap-ups. The file also serves as testcase. The content can change over time and can also serve as a trigger for discussions on the mailing list. Suggestions are welcome.

Hans Hagen
Hasselt NL

1 Vertical spacing

The low level way to input inline math in TeX is

```
$ e = mc^2 $
```

while display math can be entered like:

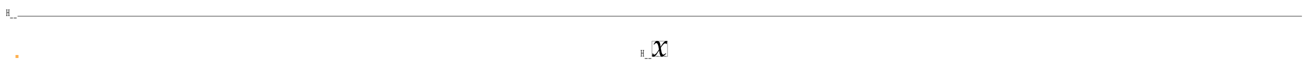
```
$$ e = mc^2 $$
```

The inline method is still valid, but for display math the $$ method should not be used. This has to do with the fact that we want to control spacing in a consistent way. In ConTeXt the vertical spacing model is rather stable although in MkIV the implementation is quite different. It has always been a challenge to let this mechanism work well with space round display formulas. This has to do with the fact that (in the kind of documents that we have to produce) interaction with already present spacing is somewhat tricky.

Of course much can be achieved in TeX but in ConTeXt we need to have control over the many mechanisms that can interact. Given the way TeX handles space around display math there is no real robust solution possible that gives visually consistent space in all cases so that is why we basically disable the existing spacing model. Disabling is easier in LuaTeX and recent versions of MkIV have been adapted to that.

In pure TeX what happens is this:

```
$$ x $$
```



A horizontal box (visualized by the thin rule on its baseline) get added which triggers a baselineskip. Then the formula is put below it. We can get rid of that box with `\noindent`:

```
\noindent $$ x $$
```

In addition (not shown here) vertical space is added before and after the formula and left- and rightskip on the edges. In fact typesetting display math goes like this:

- typeset the formula using display mode and wrap it in a box
- add an equation number, if possible in the same line, otherwise on a line below
- in the process center the formula using the available display width and required display indentation
- add vertical space above and below (depending also in displays being short in relation to the previous line
- at the same time also add penalties that determine the break across pages

Apart from the spacing around the formula and the equation number, typesetting is not different from:

```
\hbox {$ \displaystyle x $}
```

So this is what we will use by default in ConTEXt in order to better control spacing as spacing around math is a sensitive issue. Because math itself can have a narrow band, for instance a lone x, or relative much depth, as with y, or both depth and height as in $(1,2)$ and $x^2 + y_2$ and because a preceding line can have no or little depth and a following line little height, the visual appearance can become inconsistent. The default approach is to force consistent spacing, but when needed we can implement variants.

Spacing around display math is set up with `\setupformulas`:

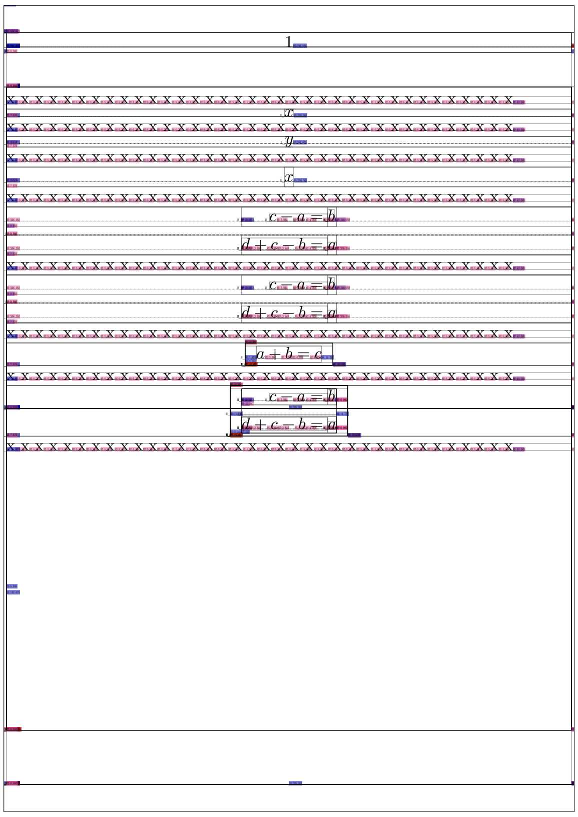
```
\setupformulas
  [spacebefore=big,
   spaceafter=big]
```

When the whitespace is larger that setting wins because as usual the larger of blanks or whitespace wins.

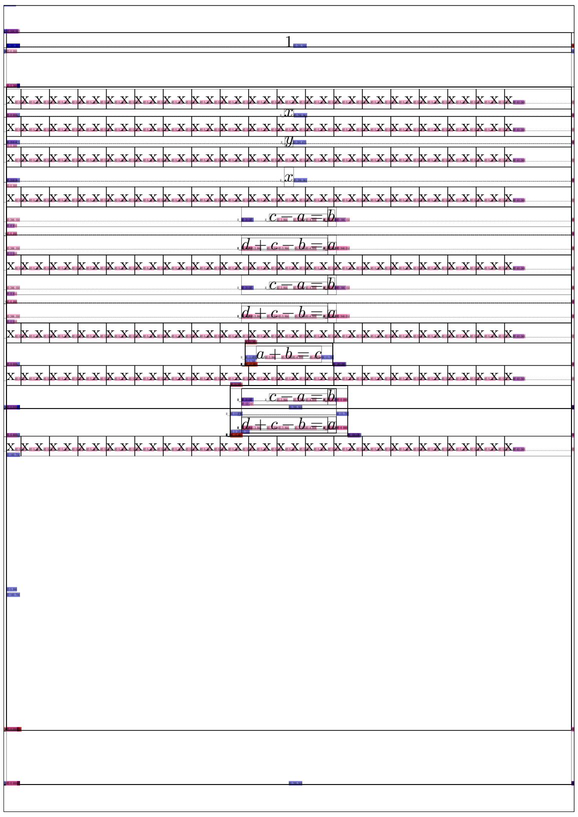
In figures 1.1, figures 1.2 and 1.3 we see how things interact. We show lines with and without maximum line height and depth (enforced by struts) alongside.

Because we want to have control over the placement of the formula number but also want to be able to align the formula with the left or right edge of the text area, we don't use the native display handler by default. We still have a way to force this, but this is only for testing purposes. By default a formula is placed centered relative to the current text, including left and right margins.

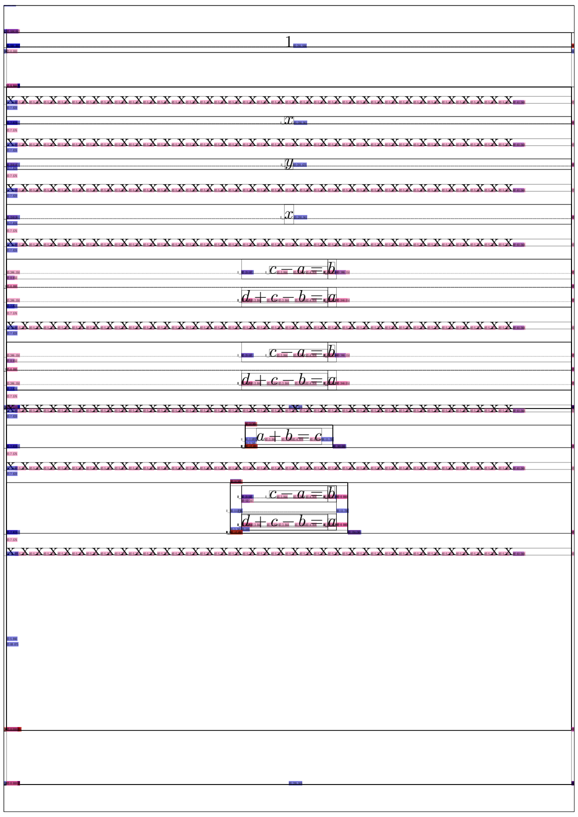
```
\fakewords{20}{40}

\startitemize
    \startitem
        \fakewords{20}{40}
        \placeformula
            \startformula
                \fakeformula
            \stopformula
    \stopitem
    \startitem
        \fakewords{20}{40}
    \stopitem
\stopitemize

\fakewords{20}{40}\epar
```



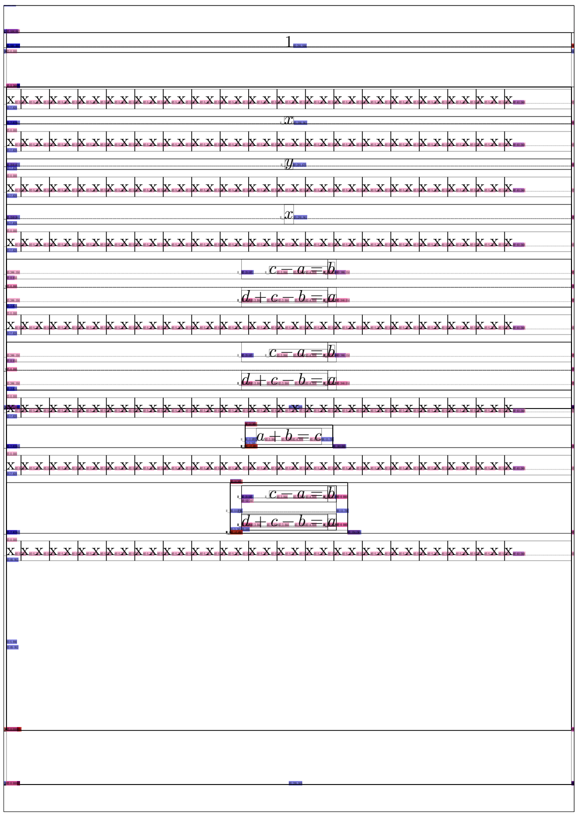
`natural + none + ws none`



`strut + none + ws none`

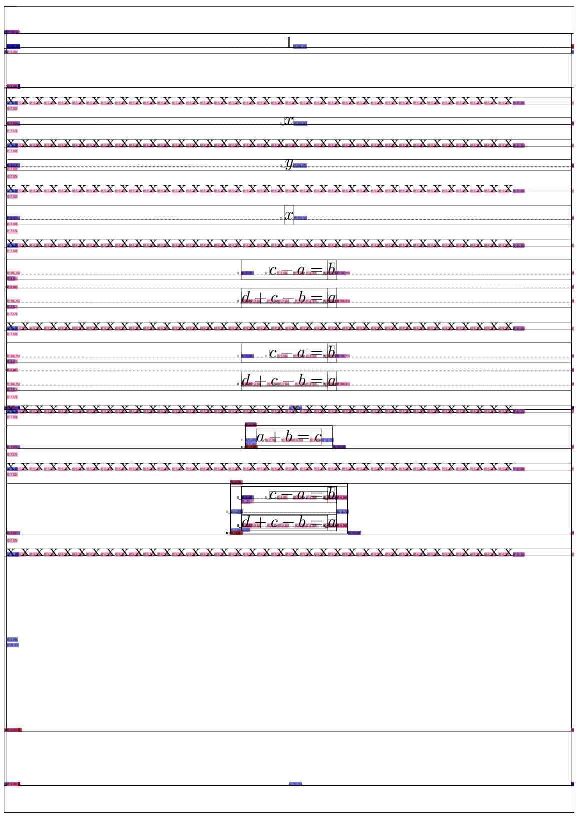


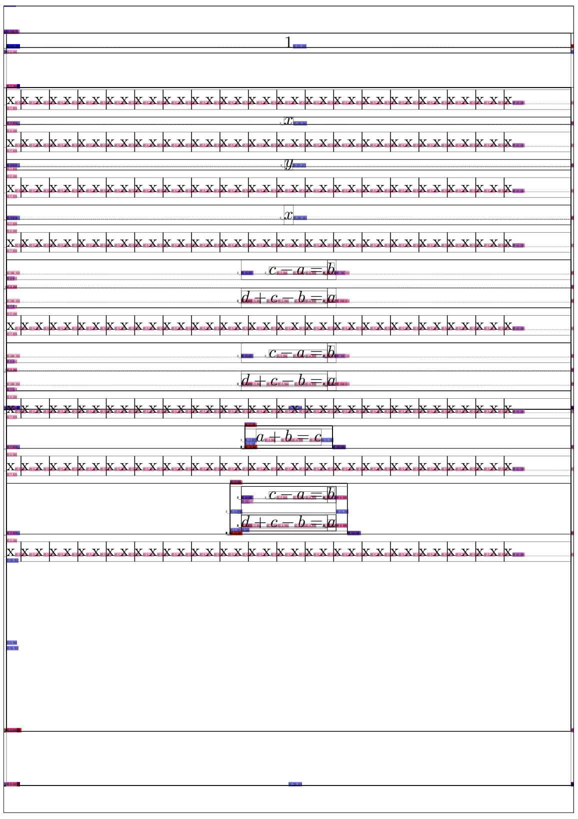
`natural + medium + ws none`

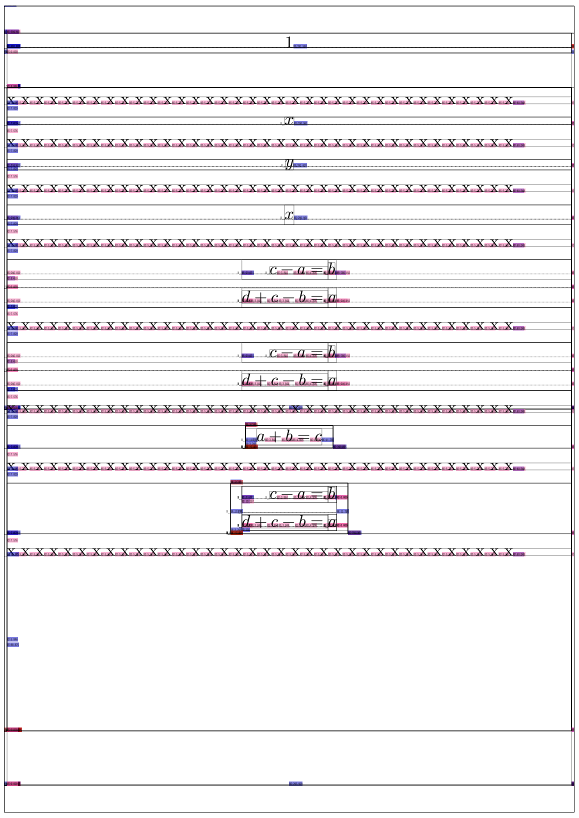


`strut + medium + ws none`

Figure 1.1 No whitespace.


natural + none + ws medium


strut + none + ws medium


natural + medium + ws medium

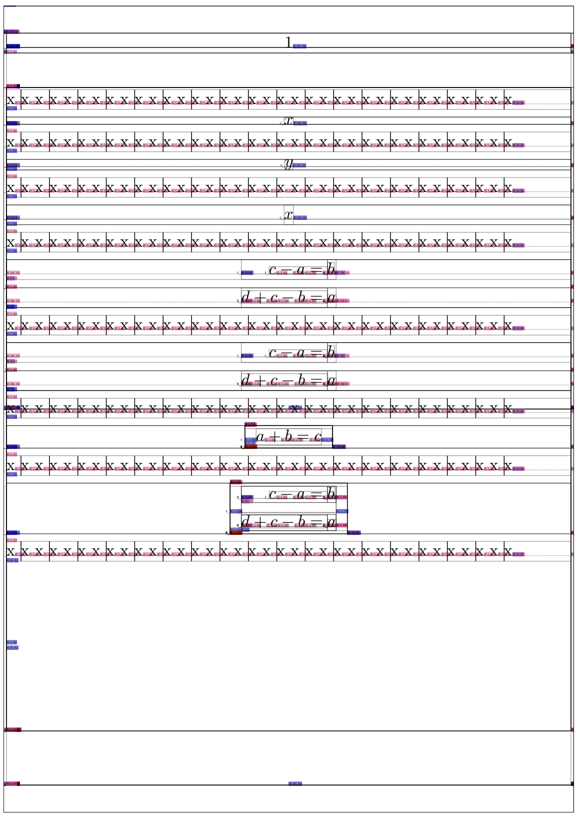
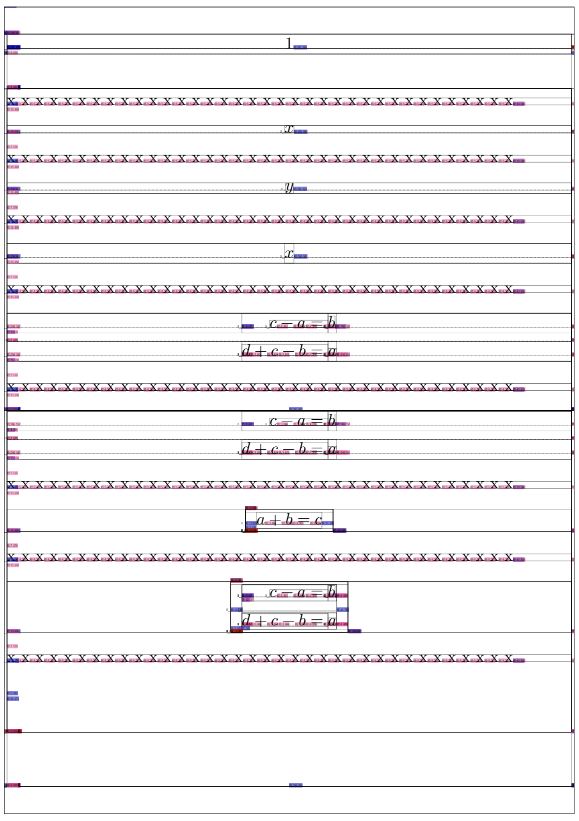
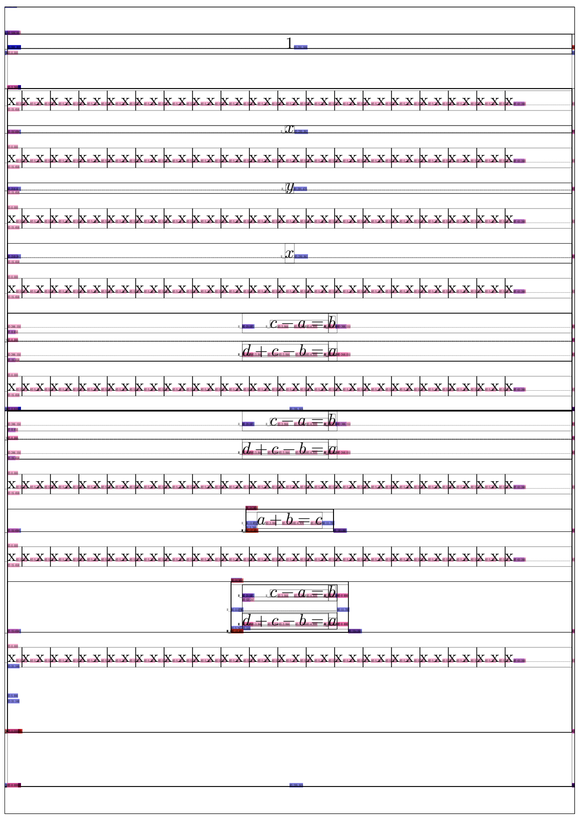
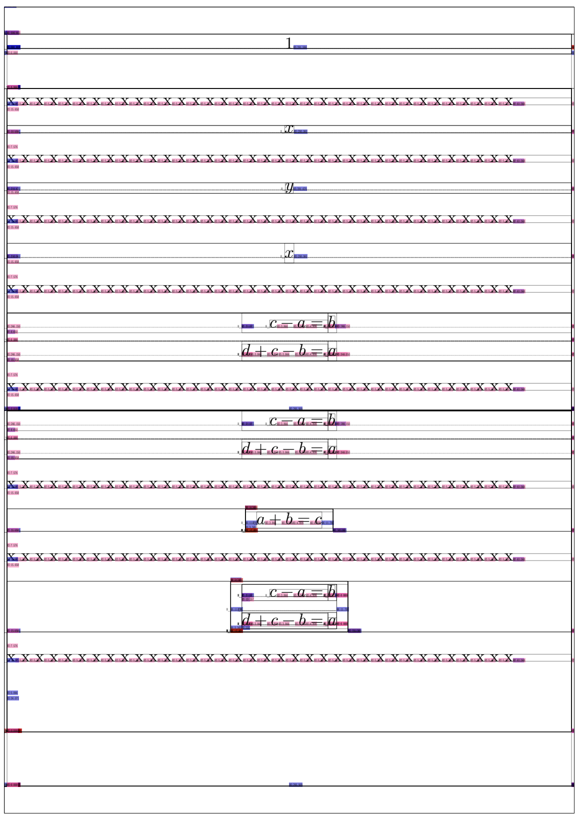

strut + medium + ws medium

Figure 1.2 Whitespace the same as display spacing.







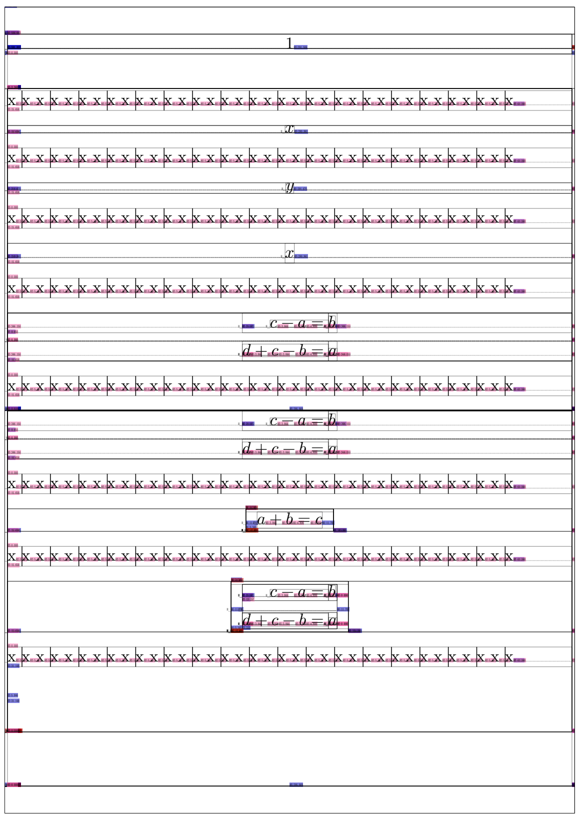


Figure 1.3 Whitespace larger than display spacing.

- ▬

  ■ + ■ + ■ + ■ = ■ (1.1)

- ▬

▬ ▬

In the next examples we explicitly align formulas to the left (`flushleft`), center (`middle`) and right (`flushright`):

```
\setupformulas[align=flushleft]
\startformula\fakeformula\stopformula
\setupformulas[align=middle]
\startformula\fakeformula\stopformula
\setupformulas[align=flushright]
\startformula\fakeformula\stopformula
```

The three cases show up as:

■ + ■ + ■ = ■

■ + ■ + ■ + ■ = ■

■ + ■ + ■ + ■ + ■ = ■

You can also set a left and/or right margin:

■ + ■ + ■ + ■ + ■ = ■

■ + ■ + ■ + ■ + ■ + ■ = ■

■ + ■ + ■ + ■ = ■

With formula numbers these formulas look as follows:

■ + ■ + ■ + ■ + ■ + ■ = ■ (1.2)

■ + ■ + ■ + ■ + ■ + ■ = ■ (1.3)

■ + ■ + ■ + ■ = ■ (1.4)

and the same with margins:

■ + ■ + ■ + ■ = ■ (1.5)

■ + ■ + ■ + ■ + ■ = ■ (1.6)

■ + ■ + ■ = ■ (1.7)

When the `margin` option is set to `standard` or `yes` the current indentation (when set) or left skip is added to the left side.

```
\setupformulas[align=flushleft]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{1.8}$$

```
\setupformulas[align=flushleft,margin=standard]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{1.9}$$

The distance between the formula and the number is only applied when the formula is left or right aligned.

```
\setupformulas[align=flushright,distance=0pt]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{1.10}$$

```
\setupformulas[align=flushright,distance=2em]
\startformula \fakeformula \stopformula
\placeformula \startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare \tag{1.11}$$

1.1 Scripts

Spacing is a trade off because there is no way to predict all usage. Of course a font can be very detailed in where italic correction is to be applied and how advanced stepwise kerns are used, but not many fonts have extensive information. Here are some differences in rendering. In OpenType the super- and subscript of an integral are moved right and left half of the italic correction.

$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$	$F_j = \int_a^b$
Latin Modern	Pagella	Dejavu	Cam- bria	Lucida OT	Xits

2 Framing

The `\framed` macro is one of the core constructors in ConTEXt and it's used all over the place. This macro is unlikely to change its behaviour and as it has evolved over years it comes with quite some options and some can interfere with the expectations one has. In general using this macro works out well but you need to keep an eye on using struts and alignment.

```
\framed{$e=mc^2$}
```

The outcome of this is:

$\boxed{e = mc^2}$

There is a bit of offset (that you can set) but also struts are added as can be seen when we visualize them:

$\boxed{e = mc^2}$

These struts can be disabled:

```
\framed[strut=no]{$e=mc^2$}
```

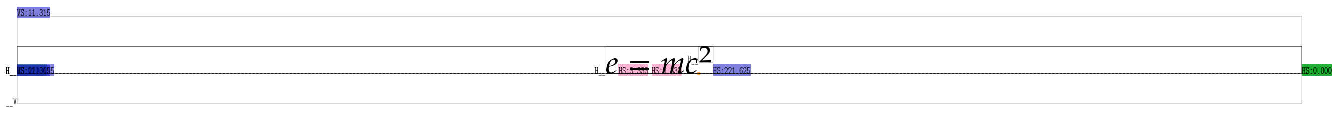
Now the result is more tight.

$\boxed{e = mc^2}$

These struts are the way to get a consistent look and feel and are used frequently in ConTEXt. We mention these struts because they get in the way when we frame a display formula. Let's first look at what happens when we just package a formula in a box:

```
\vbox\bgroup
    \startformula
        e = mc^2
    \stopformula
\egroup
```

We get:



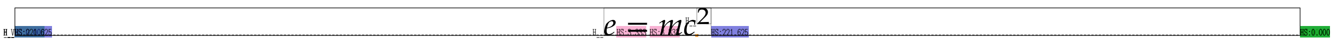
Now there are a few properties of displaymath that one needs to keep in mind when messing around with them this way. First of all display math is meant to be used as part of the page stream. This means that spacing above and below is adapted to what comes before and after. It also means that, because formulas can be numbered, we have some settings that relate to horizontal placement.

The default vertical spacing is easy to get rid of:

```
\vbox\bgroup
    \startformula[packed]
```

```
        e = mc^2
    \stopformula
\egroup
```

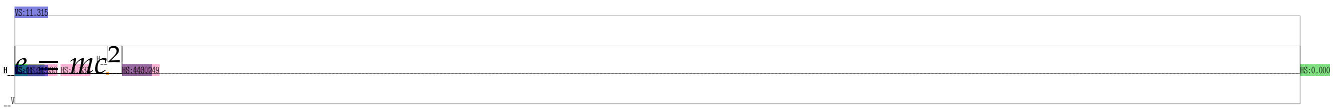
This gives:



Another handy keyword is `tight`:

```
\vbox\bgroup
    \startformula[tight]
        e = mc^2
    \stopformula
\egroup
```

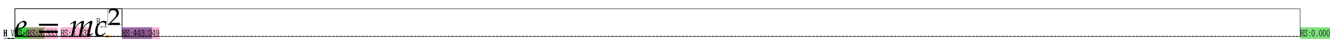
This gives:



We can combine these two:

```
\vbox\bgroup
    \startformula[packed,tight]
        e = mc^2
    \stopformula
\egroup
```

This gives:



Just in case you wonder why we need to go through these troubles: keep in mind that we are wrapping something (math) that normally goes in a vertical list with text above and below.

The `packed` and `tight` options can help when we want to wrap a formula in a frame:

```
\framed
    [strut=no]
    {
        \startformula[packed,tight]
            e = mc^2
        \stopformula
    }
```

which renders as:

$\boxed{e = mc^2}$

There is a dedicated math framed instance that is tuned to give better results and automatically switches to math mode:

```
\mframed {
    e = mc^2
}
```

becomes:

$\boxed{e = mc^2}$

Framing a formula is also supported as a option, where the full power of framed can be applied to the formula. We will illustrate this in detail on the next pages. For this we use the following sample:

```
\setuplayout[topspace=5mm,bottomspace=5mm,height=middle,header=1cm,footer=0cm]

\starttext

\startbuffer[sample]
    \enabletrackers[formulas.framed] \showboxes
    \startformula
        e = mc^2
    \stopformula
    \par
    \startformula
        e = mc^2
    \stopformula
    \startformula
        e = mc^2
    \stopformula
    \startformula
        e \dorecurse{12} { = mc^2 }
    \stopformula
    \startplaceformula
        \startformula
            e = mc^2
        \stopformula
    \stopplaceformula
    \startplaceformula
        \startformula
            e \dorecurse{12} { = mc^2 }
        \stopformula
    \stopplaceformula
\stopbuffer
```

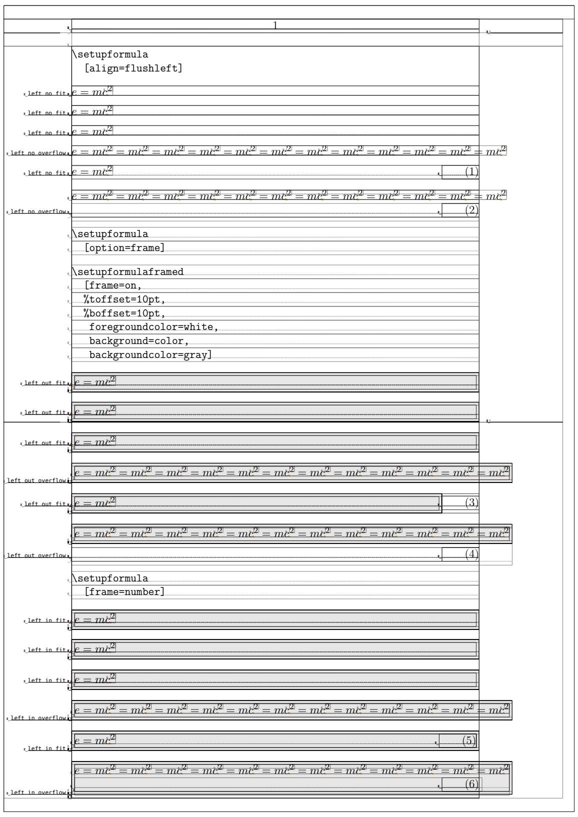
```
\startbuffer[setup-b]
\setupformula
  [option=frame]
\stopbuffer

\startbuffer[setup-d]
\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]
\stopbuffer

\startbuffer[setup-c]
\setupformula
  [frame=number]
\stopbuffer

\startbuffer[all]
\start
    \typebuffer[setup-a]
    \getbuffer[setup-a]
    \getbuffer[sample]
    \typebuffer[setup-b]
    \typebuffer[setup-d]
    \getbuffer[setup-b]
    \getbuffer[setup-d]
    \getbuffer[sample]
    \typebuffer[setup-c]
    \getbuffer[setup-c]
    \getbuffer[sample]
    \page
\stop
\stopbuffer

\startbuffer
    \startbuffer[setup-a]
    \setupformula
      [align=flushleft]
    \stopbuffer
    \getbuffer[all]
    \startbuffer[setup-a]
    \setupformula
```

```
      [align=flushleft,location=left]
    \stopbuffer
    \getbuffer[all]

    \startbuffer[setup-a]
    \setupformula
      [align=middle]
    \stopbuffer
    \getbuffer[all]
    \startbuffer[setup-a]
    \setupformula
      [align=middle,location=left]
    \stopbuffer
    \getbuffer[all]

    \startbuffer[setup-a]
    \setupformula
      [align=flushright]
    \stopbuffer
    \getbuffer[all]
    \startbuffer[setup-a]
    \setupformula
      [align=flushright,location=left]
    \stopbuffer
    \getbuffer[all]
\stopbuffer

\getbuffer

\startbuffer[setup-b]
\setupformula
  [option={tight,frame}]
\stopbuffer

\getbuffer

\stoptext
```

In figure 2.1, 2.2 and 2.3 you see some combinations. You can run this example on your machine and see the details.

With each formula class a framed variants is automatically created:

```
\defineformula
  [foo]
```



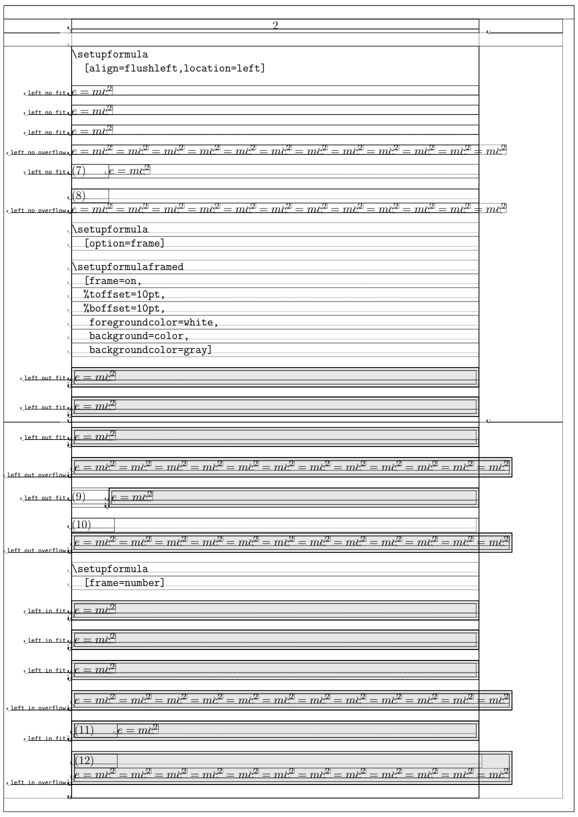
```
\setupformula
  [align=flushleft]
```

```
\setupformula
  [option=frame]
```

```
\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
    foregroundcolor=white,
    background=color,
    backgroundcolor=gray]
```

```
\setupformula
  [frame=number]
```

right + flushleft



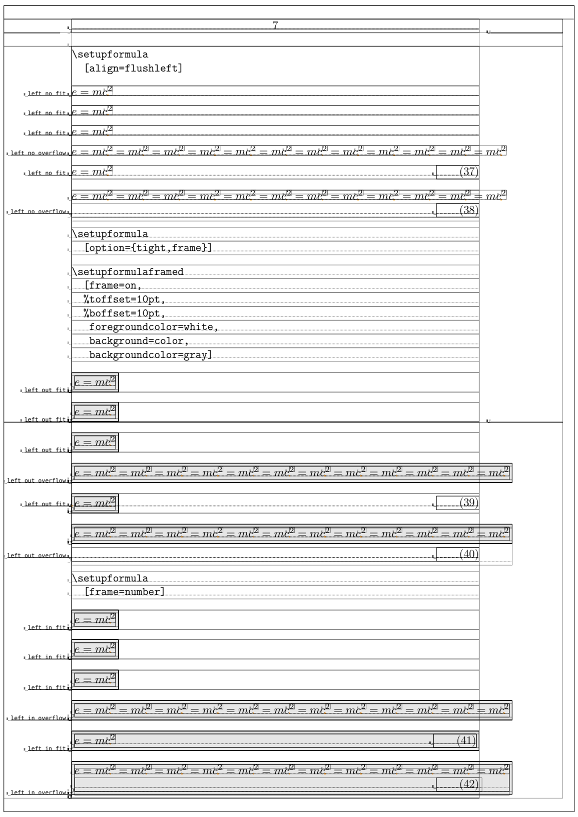
```
\setupformula
  [align=flushleft,location=left]
```

```
\setupformula
  [option=frame]
```

```
\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
    foregroundcolor=white,
    background=color,
    backgroundcolor=gray]
```

```
\setupformula
  [frame=number]
```

right + flushleft



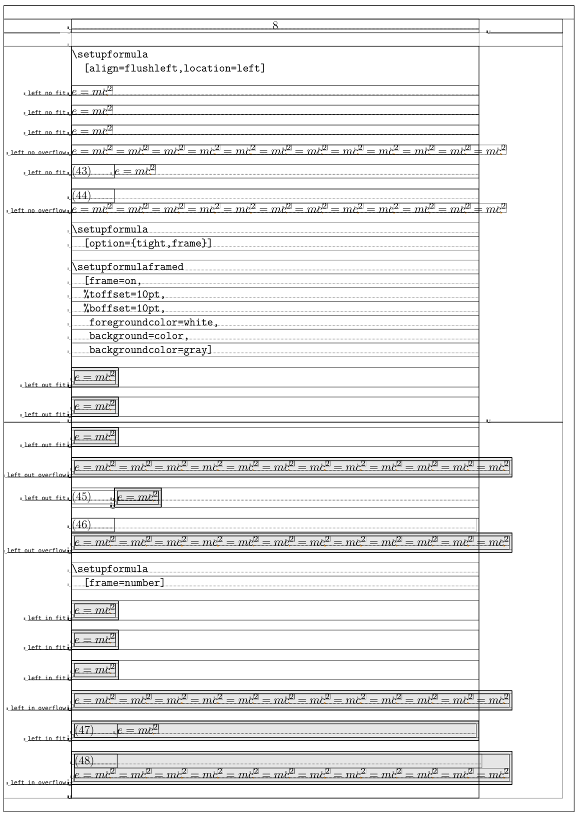
```
\setupformula
  [align=flushleft]
```

```
\setupformula
  [option={tight,frame}]
```

```
\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
    foregroundcolor=white,
    background=color,
    backgroundcolor=gray]
```

```
\setupformula
  [frame=number]
```

left + flushleft + tight



```
\setupformula
  [align=flushleft,location=left]
```

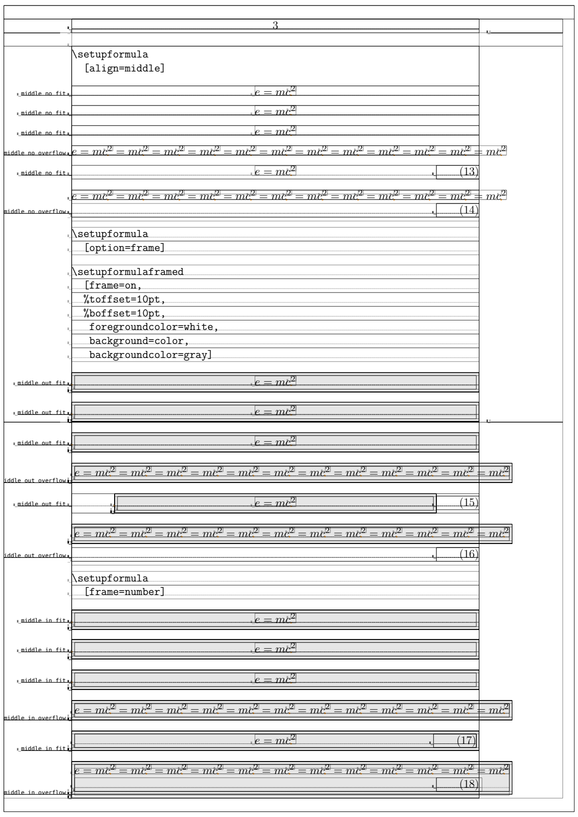
```
\setupformula
  [option={tight,frame}]
```

```
\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
    foregroundcolor=white,
    background=color,
    backgroundcolor=gray]
```

```
\setupformula
  [frame=number]
```

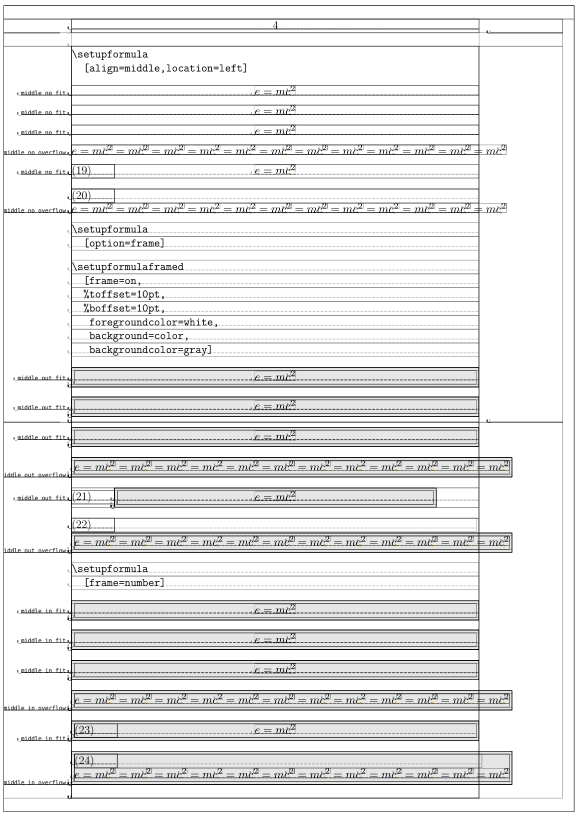
left + flushleft + tight

Figure 2.1 Framed formulas flushed left.



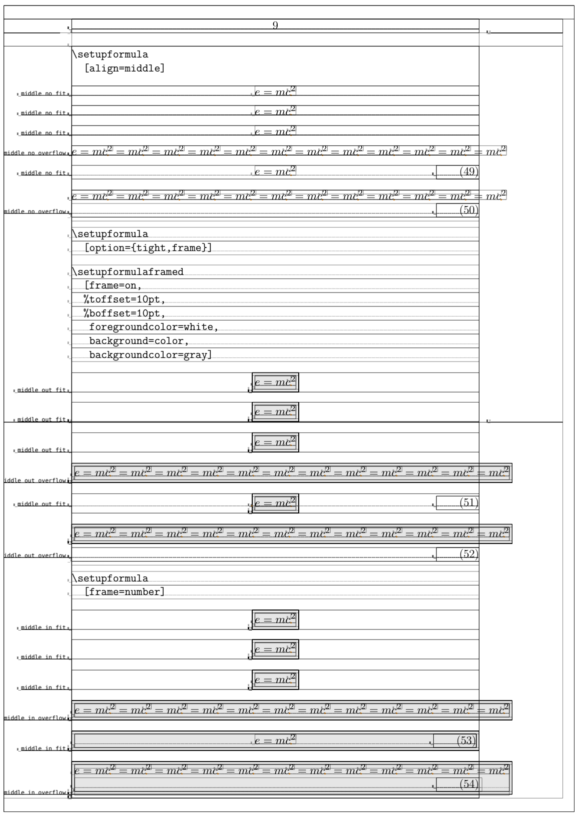
```
\setupformula
  [align=middle]

\setupformula
  [option=frame]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

right + middle



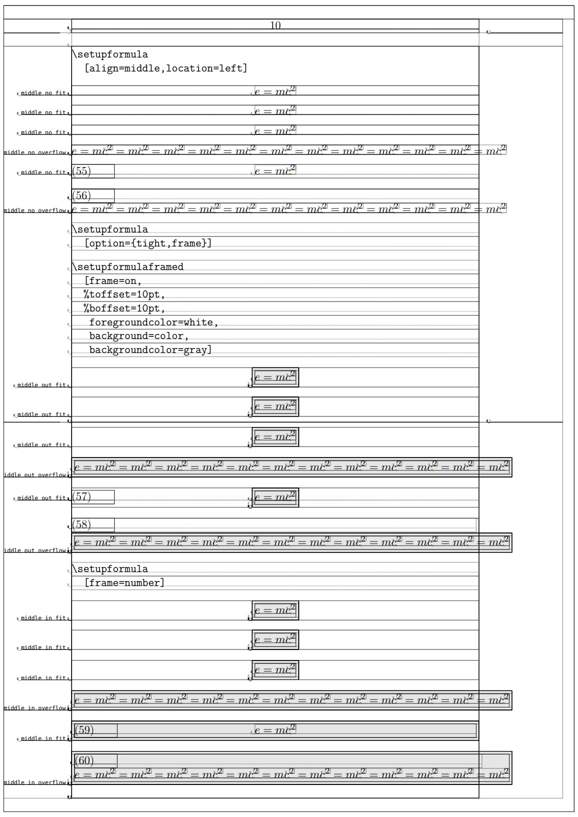
```
\setupformula
  [align=middle,location=left]

\setupformula
  [option=frame]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

right + middle



```
\setupformula
  [align=middle]

\setupformula
  [option={tight,frame}]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

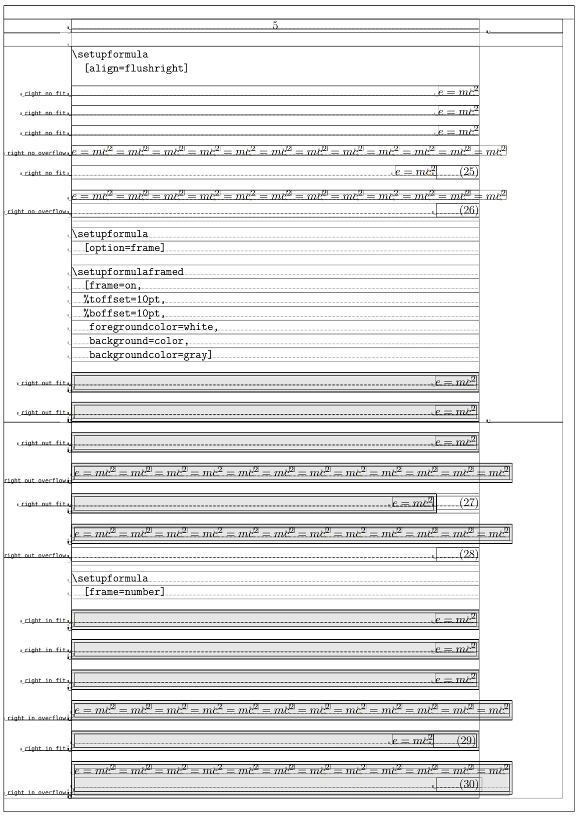
left + middle + tight



```
\setupformula
  [align=middle,location=left]

\setupformula
  [option={tight,frame}]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

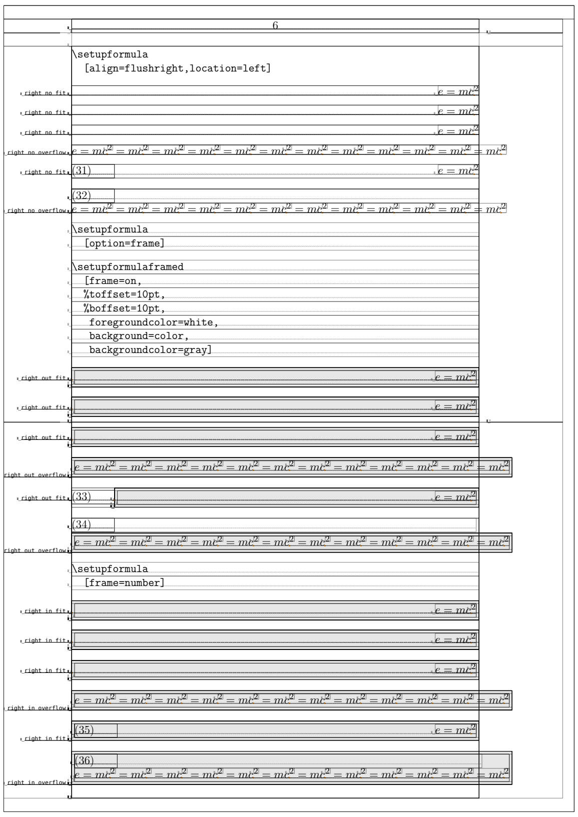
left + middle + tight

Figure 2.2 Framed formulas centered.



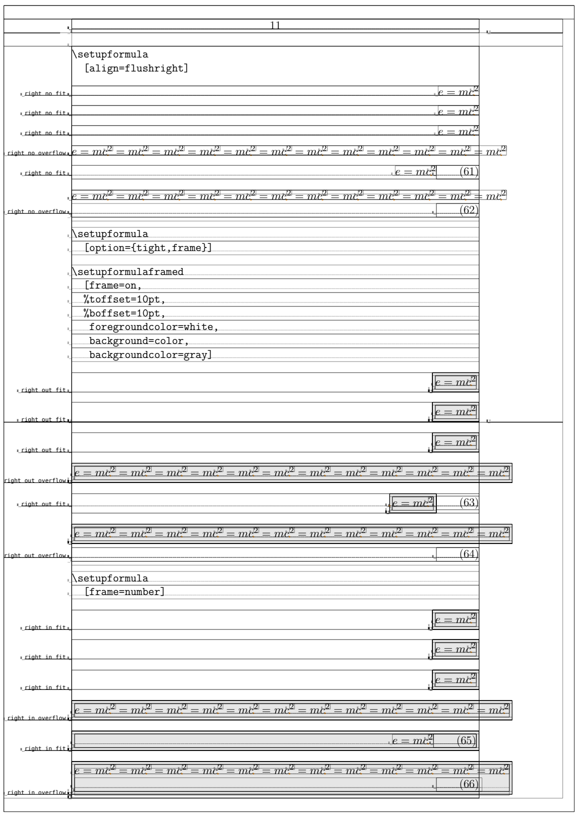
```
\setupformula
  [align=flushright]

\setupformula
  [option=frame]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

right + flushright



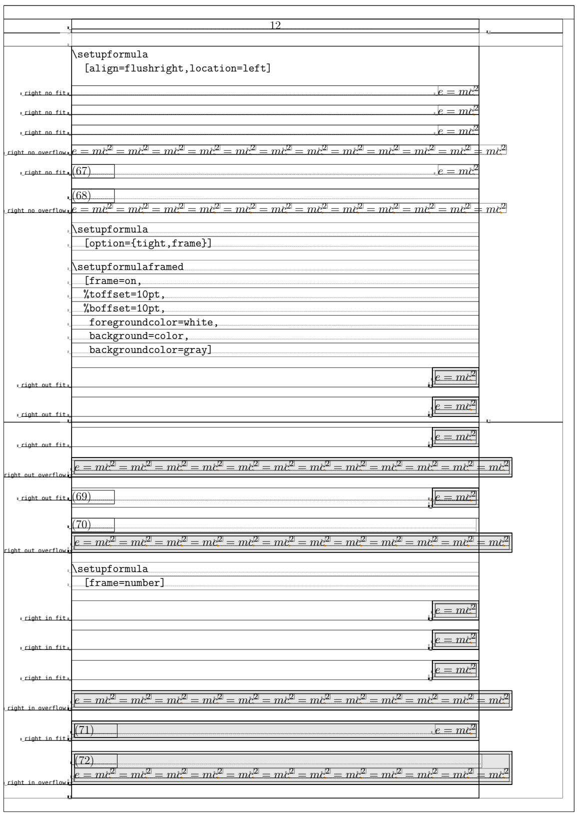
```
\setupformula
  [align=flushright,location=left]

\setupformula
  [option=frame]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

right + flushright



```
\setupformula
  [align=flushright]

\setupformula
  [option={tight,frame}]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

left + flushright + tight



```
\setupformula
  [align=flushright,location=left]

\setupformula
  [option={tight,frame}]

\setupformulaframed
  [frame=on,
  %toffset=10pt,
  %boffset=10pt,
   foregroundcolor=white,
   background=color,
   backgroundcolor=gray]

\setupformula
  [frame=number]
```

left + flushright + tight

Figure 2.3 Framed formulas flushed right.

```
\setupformulaframed
  [foo]
  [frame=on,
   framecolor=red]

\startfooformula[frame]
    e=mc^2
\stopfooformula
```

This time you get a red frame:

$$\boxed{e = mc^2}$$

You can also frame the number, as in:

```
\setupformulaframed[framecolor=red,frame=on,offset=1ex]
\setupformula[option=frame,color=blue]
\setupformula[numbercommand={\inframed[framecolor=green]}]

\startplaceformula
    \startformula
        2 + 2 = 2x
    \stopformula
\stopplaceformula
```

The boxes get properly aligned:

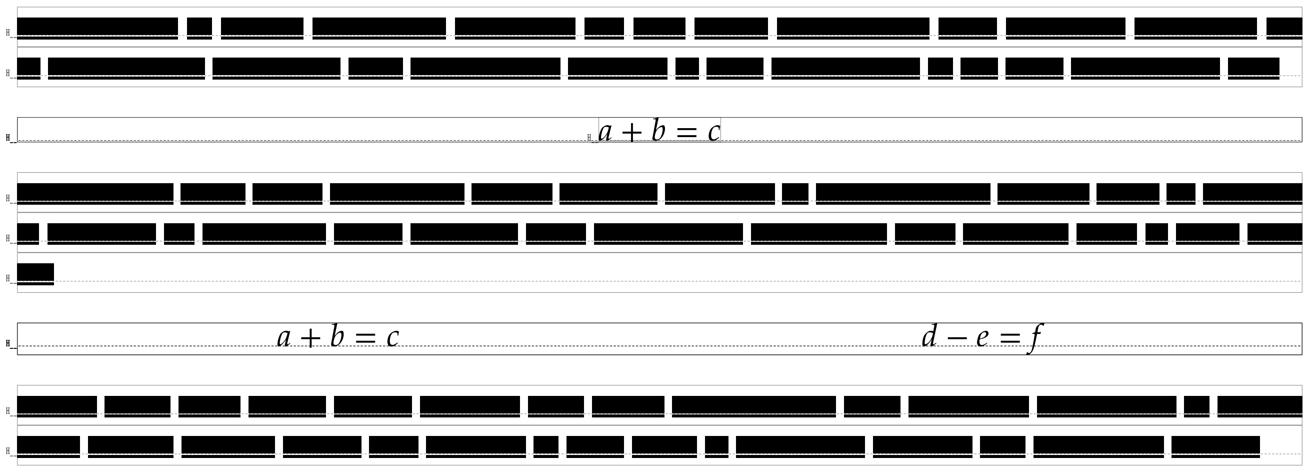
$$\boxed{2 + 2 = 2x} \qquad \boxed{(2.1)}$$

3 Combining formulas

Multiple formulas can be combined by wrapping them:

```
\fakewords{20}{30}

\startformula
    a + b = c
\stopformula

\fakewords{20}{30}

\startformulas
    \startformula
        a + b = c
    \stopformula
    \startformula
        d - e = f
    \stopformula
\stopformulas

\fakewords{20}{30}

\startformulas
    \startformula
        \frac{\frac{x}{y}}{b} = c
    \stopformula
    \startformula
        d - e = f
    \stopformula
\stopformulas

\fakewords{20}{30}
```

When we bump the space around formulas to big we get this:



$\frac{\frac{x}{y}}{b} = c$ $d - e = f$

[redacted]

The formulas get aligned on the baselline which in turn relates to the math axis of the formula.

4 Features

4.1 Default features

Math fonts are loaded in so called basemode, which gives them a traditional treatment in the engine. However, we do support features in basemode too, so setting them can influence what gets passed to TEX. Also, in math mode, some font features (like `dtls` and stylistic alternates) are applied dynamically.

The default `mathematics` feature set is as follows:

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mode	base
script	math

We don't discuss the exact meaning of these options here because normally you don't have to deal with them. If a math font demands something special, the place to deal with it is the related font goodie file.

This feature set is the parent of two other sets: `mathematics-l2r` and `mathematics-r2l`:

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mode	base
script	math

This one is the same as the parent but the right-to-left variant is different:

kern	yes
language	dflt
locl	yes
mathalternates	yes
mathitalics	yes
mode	base
rtlm	yes
script	math

Eventually we need size related feature sets and again we define a parent and direction specific ones: `math-text`, `math-script` and `math-scriptscript`.

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mode	base

script	math
ssty	no

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mathsize	yes
mode	base
script	math
ssty	1

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mathsize	yes
mode	base
script	math
ssty	2

The left-to-right sets `math-*-l2r` are:

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mode	base
script	math
ssty	no

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mathsize	yes
mode	base
script	math
ssty	1

kern	yes
language	dflt
mathalternates	yes
mathitalics	yes
mathsize	yes
mode	base

```
script         math
ssty           2
```

The right-to-left sets `math-*-r2l` are:

```
kern           yes
language       dflt
locl           yes
mathalternates yes
mathitalics    yes
mode           base
rtlm           yes
script         math
ssty           no
```

```
kern           yes
language       dflt
locl           yes
mathalternates yes
mathitalics    yes
mathsize       yes
mode           base
rtlm           yes
script         math
ssty           1
```

```
kern           yes
language       dflt
locl           yes
mathalternates yes
mathitalics    yes
mathsize       yes
mode           base
rtlm           yes
script         math
ssty           2
```

There are a few extra sets defined but these are meant for testing or virtual math fonts. The reason for showing these sets is to make clear that the number of features is minimal and that math is a real script indeed.

The `kern` features is questionable. In traditional TEX there are kerns indeed but in OpenType math kerns are not used that way because a more advanced kerning feature is present (and that one is currently always enabled). We used to set the following but these make no sense.

```
liga=yes, % (traditional) ligatures
```

```
tlig=yes, % tex ligatures, like -- and ---
trep=yes, % tex replacements, like the ' quote
```

Math fonts normally have no ligatures and supporting the TEX specific ones can actually be annoying. So, in todays ConTEXt these are no longer enabled. Just consider the following:

```
$- \kern0pt -   \kern 0pt \mathchar"2D$
$- \kern0pt --  \kern 0pt \mathchar"2D \mathchar"2D$
$- \kern0pt --- \kern 0pt \mathchar"2D \mathchar"2D \mathchar"2D$
```

The `-` is mapped onto a minus sign and therefore several in succession become multiple minus signs. The `\mathchar"2D` will remain the character with that slot in the font so there we will see a hyphen. If we would enable the `tlig` feature several such characters would be combined into an endash or emdash. So how do we get these than? Because getting a hyphen directly involves a command, the same is true for its longer relatives: `\endash` and `\emdash`.

− − -
− − −--
− − − − ---

As convenience we have defined a special `\mathhyphen` command. Watch the fact that a text hyphen in math mode is a minus in math! As comparison we also show the plus sign.

command	math	text
`\mathhyphen`	-	-
`\texthyphen`	−	-
`-`	−	-
`+`	+	+
`\endash`	–	–
`\emdash`	—	—

4.2 Stylistic alternates

todo

4.3 Dotless variants

todo