The physics markup language

pml

Version 1.08

Ian Lawrence Wednesday, 24 July 2019

What's new

VectorQuantity,FractionVectorABC,ProductVector-dotABC,ProductVectorcrossABC added to Adding technical text

Section 1: Introduction

The physics markup language (pml) is a language for authoring and editing educational physics text that will be rendered using HTML, so it is designed for websites that need to represent physics accurately, and where it's assumed that the ability to maintain the website is important. The language is rather simple to extend, and so can be adapted to local needs, for example extensions for supporting physics teaching coaches, or rendering multiple choice questions.

You can write in any text processor, and then a live server can render web pages as you type, using a javascript parser to render the page.

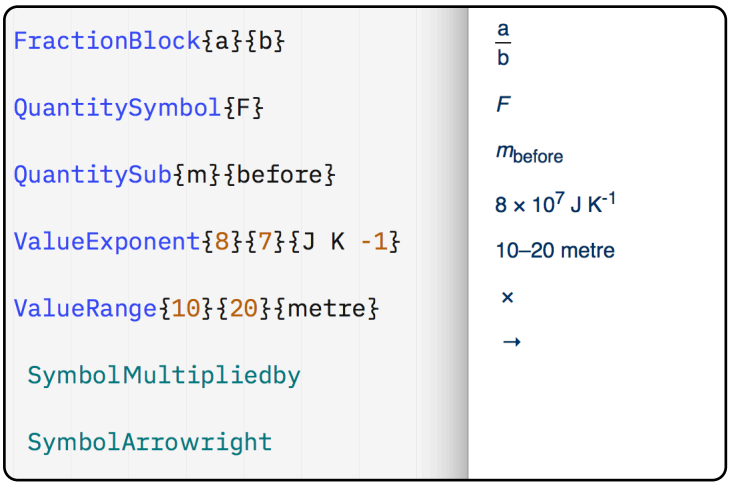
The authors or editors are assumed to be physics graduates or be working closely with such people, and a knowledge of their competences and styles of thinking frames the design. There is assumed to be at least some supervisory teaching intelligence, so whereas the design and implementation should minimise syntactical error, semantic errors necessarily remain the responsibility of the humans. However, because it is just plain text, and many tools exist to deal with multiple text files, it’s somewhat easier to reduce inconsistencies and to hunt down infelicities.

A further design constraint is that expressing physics requires precision beyond correctly punctuated English: there are technical conventions which have to appear inline as well as in blocks by themselves.

Some examples

```
FractionBlock{a}{b}
QuantitySymbol{F}
QuantitySub{m}{before}
ValueExponent{8}{7}{J K −1}
ValueRange{10}{20}{metre}
 SymbolMultipliedby
 SymbolArrowright
```

In these few examples you see two implementation principles: use of camelcase words, and parameters provided in curly braces. Thse are used throughout. pml provides a set of rules, instantiated as scripts, that transform combinations of these special elements embedded in other plain text to standards-compliant html, which can be displayed by any browser. Wrapping such output can then provide web pages. With appropriate css files, you can get output like this:



On the left is a text processor, in which the special camelcase words are automatically highlighted: on the right is a browser window in which these expressions have been rendered. The same arrangement of panes is used in all exemplifying screen shots in this document.

Entering and editing and storing pml

pml is stored in markdown files, with a file extension '.md'. You can write such files on any word or text processor, making a wider range of devices is therefore available. If writing a lot of pml, you might like to invest in configuring a proper text editor, so that you can see the special words and syntax highlighted, so reducing errors, and so that you can use auto-completion and a snippet system to ease the remembering and typing load. You can store such markdown files almost anywhere.

Nesting

More complex constructions can be achieved by nesting expressions (a keyword together with its parameters).

Start with a simple example.

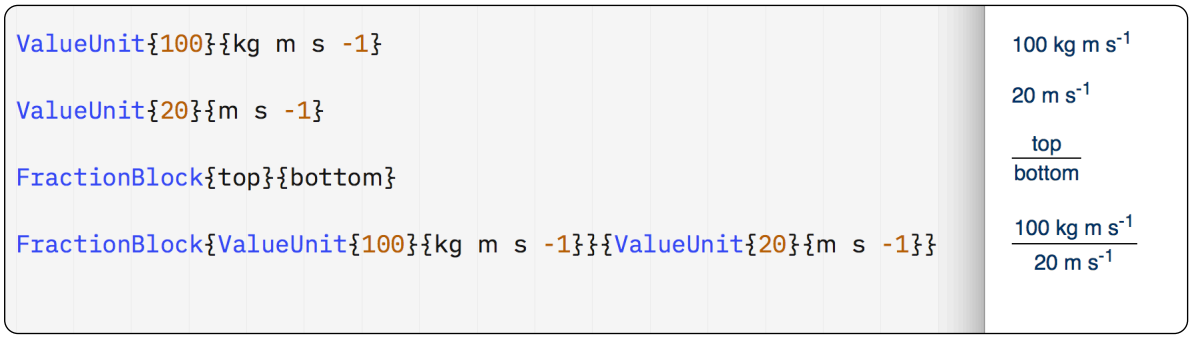
Here are three keywords with parameters:

```
ValueUnit{100}{kg m s -1}
ValueUnit{20}{m s -1}
FractionBlock{top}{bottom}
```

You can combine these by replacing the 'top' and 'bottom' words in the FractionBlock with the contents of the first two lines.

```
FractionBlock{ValueUnit{100}{kg m s -1}}{ValueUnit{20}{m s -1}}
```

Again, with the text processor on the left and a browser window on the right, you can see the following.



Rendering pml

pml files an be acted on by a script to generate html: this mapping is the essence of the design of the language. How that html appears is controlled by css: there is a minimal set of css on which pml depends to render the physics correctly, but beyond that you can write css to display the html classes embedded by the script as you wish.

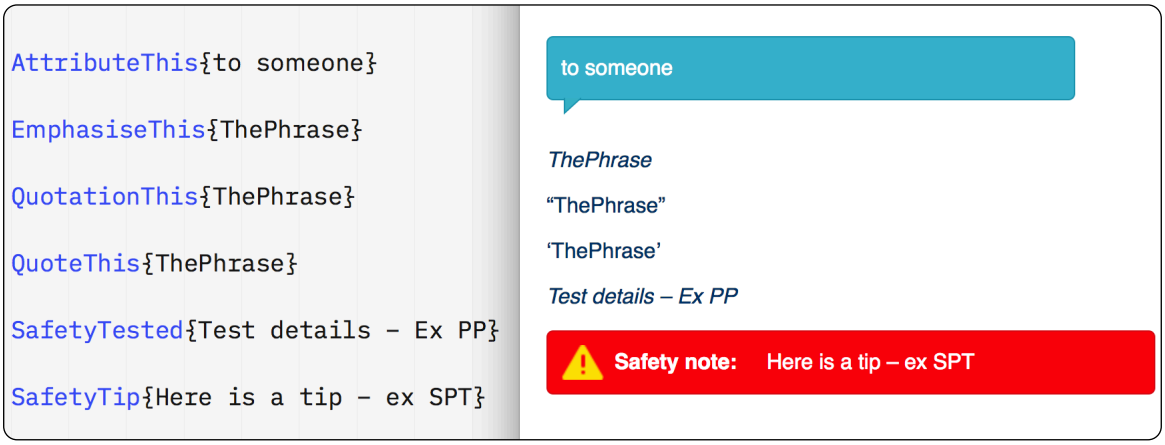
Section 2: keywords, grouped

Styling text

Use these to style the text, for emphasis, or attribution, or to add single or double quotation marks. How the tags are interpreted when rendered will depend on the css. The final two are supporting particular features present on IoP web pages.

```
AttributeThis{to someone}
EmphasiseThis{ThePhrase}
BoldThis{ThePhrase}
QuotationThis{ThePhrase}
QuoteThis{ThePhrase}
SafetyTested{Test details - Ex PP}
SafetyTip{Here is a tip - ex SPT}
```

Here is what you might see.



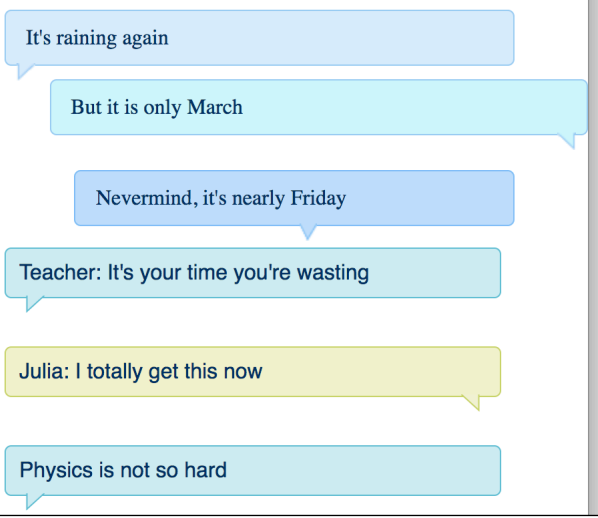
Adding speech bubbles

These are stylised speech bubbles, again dependent on the css to render.

```
TeacherOne{It's raining again}
TeacherTwo{But it is only March}
TeacherThree{Never mind, it's nearly Friday}
TeacherSpeak{It's your time you're wasting}
TeacherTip{Getting angry only wears you out}
PupilSpeak{Julia}{I totally get this now}
JustSpeak{Physics is not so hard}
WrongTrack{Mathematics is the language physics}
RightLines{Physics makes the world go round}
```

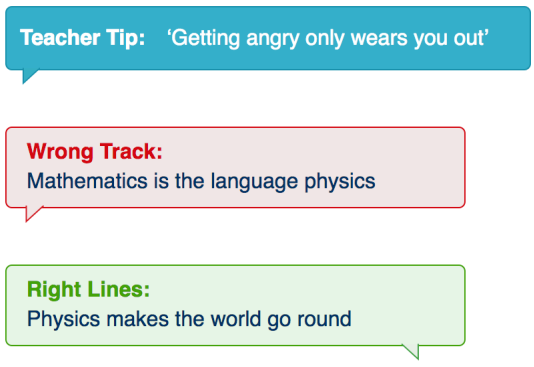
The speech might be rendered like this:

```
TeacherOne{It's raining again}

TeacherTwo{But it is only March}

TeacherThree{Nevermind, it's nearly Friday}

TeacherSpeak{It's your time you're wasting}

PupilSpeak{Julia}{I totally get this now}

JustSpeak{Physics is not so hard}
```



Or the same ideas co-opted for more formal elements in the text, like these:

```
TeacherTip{Getting angry only wears you out}

WrongTrack{Mathematics is the language physics}

RightLines{Physics makes the world go round}
```



Adding technical text

To display physical quantities:

```
QuantitySymbol{F}
QuantitySub{F}{gravity}
QuantitySuper{r}{2}
```

To go longhand:

```
WordSub{radius}{2}
WordSuper{radius}{2}
```

Or for standalone numbers:

```
NumberSuper{10}{-17}
```

```
QuantitySymbol{F}

QuantitySub{F}{gravity}

QuantitySuper{r}{2}

WordSub{radius}{2}

WordSuper{radius}{2}

NumberSuper{10}{-17}
```

F

$F_{gravity}$

r^2

$radius_2$

$radius^2$

10^{-17}

Vectors:

```
VectorOver{p}
VectorMagnitude{a}
VectorSub{v}{initial}
VectorMatrix{d}{2}{2}{2}
```

For nuclei:

```
NPNucleus{235}{92}{U}
```

```
VectorOver{p}

VectorMagnitude{a}

VectorSub{v}{initial}

VectorMatrix{d}{2}{2}{2}

NPNucleus{235}{92}{U}
```

$\vec{p}$

$\| \vec{a} \|$

$\vec{v}_{initial}$

$\vec{d} = \begin{matrix} 2 \\ 2 \\ 2 \end{matrix}$

$^{235}_{92}U$

More vectors

VectorQuantity{F}

FractionVectorABC{E}{F}{d}

ProductVectordotABC{E}{F}{d}

ProductVectorcrossABC{L}{F}{r}

```
QuantityVector{F}

FractionVectorABC{E}{F}{d}

ProductVectorDotABC{E}{F}{d}

ProductVectorCrossABC{L}{F}{r}
```

$$\boldsymbol{F}$$

$$\boldsymbol{E} = \frac{\boldsymbol{F}}{d}$$

$$E = \boldsymbol{F} \cdot \boldsymbol{d}$$

$$\boldsymbol{L} = \boldsymbol{F} \times \boldsymbol{r}$$

Units, and numbers;

JustUnit{m s -2}

ValueUnit{10}{N kg 1}

ValueExponent{8}{-9}{J s -4}

ValueRange{3}{5}{V}

ValueOrder{23}{kg}

```
JustUnit{N m 2 kg -2}

ValueUnit{10}{N kg 1}

ValueExponent{8}{-9}{J s -1}

ValueRange{3}{5}{V}

ValueOrder{23}{kg}
```

$\mathrm{N\,m^{2}\,kg^{-2}}$

$\mathrm{10\ N\,kg^{1}}$

$\mathrm{8 \times 10^{-9}\ J\,s^{-1}}$

3–5 V

$\sim 10^{23}\ \mathrm{kg}$

Quantities, units, numbers:

QuantityUnit{k}{6}{N m -1}

QuantityExponent{h}{6,6}{-34}{J s 1}

QuantityRange{v}{12}{34.2}{m s -1}

QuantityOrder{g}{-1}{N kg -1}

QuantityValue{F}{12}{N}

```
QuantityUnit{k}{6}{N m -1}

QuantityExponent{h}{6,6}{-34}{J s 1}

QuantityRange{v}{12}{34.2}{m s -1}

QuantityOrder{g}{-1}{N kg -1}

QuantityValue{F}{12}{N}
```

$k = 6\ \mathrm{N\,m^{-1}}$

$h = 6{,}6 \times 10^{-34}\ \mathrm{J\,s^{1}}$

$v = 12\text{–}34.2\ \mathrm{m\,s^{-1}}$

$g \sim 10^{-1}\ \mathrm{N\,kg^{-1}}$

$F = 12\ \mathrm{N}$

For Square roots

QuantityRoot{h}

NumberRoot{23}

```
FractionRoot{g}{l}
```

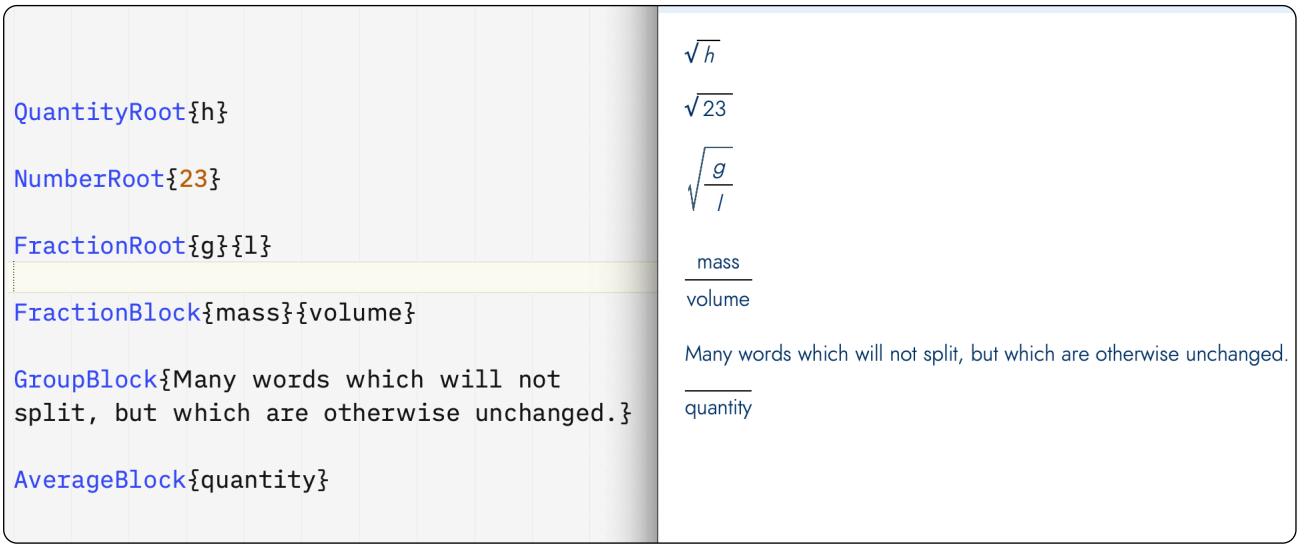
Three blocks, as components

```
FractionBlock{mass}{volume}
```

builds up fractions, as other technical primitives can be inserted in the braces, whereas grouping, to prevent splitting over lines, where you need to force this, is achieved by:

```
GroupBlock{Many words which will not split, but which are otherwise unchanged.}
```

```
AverageBlock{quantity}
```



Inserting Relationships

Many of these are 'convenience' primitives, since mostly you could build this up with assertions of equality or even from glyphs and grouping.

Some simple, three-term relationships to start off with:

```
SumABC{one}{two}{three}
```

```
DifferenceABC{one}{two}{three}
```

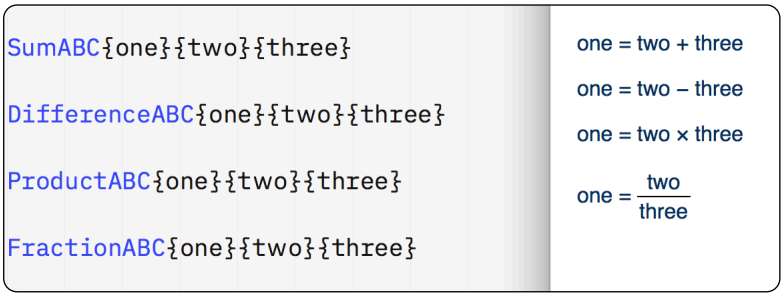
```
ProductABC{one}{two}{three}
```

```
FractionABC{one}{two}{three}
```

Sometimes you need a negative sign inserted, denoted by an 'n'.

```
ProductAnBC{one}{two}{three}
```

```
FractionAnBC{one}{two}{three}
```



And sometimes you know you're dealing with symbols for physical quantities, so insert a 'Quantity'.

ProductQuantityABC{A}{B}{C}

FractionQuantityABC{A}{B}{C}

You can do both quantity and negation:

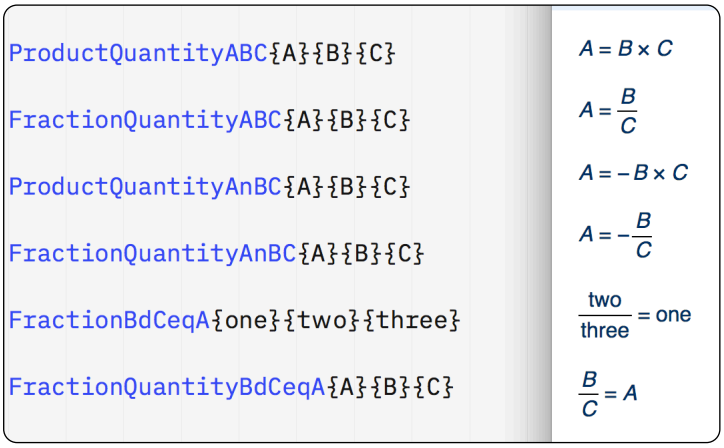
ProductQuantityAnBC{A}{B}{C}

FractionQuantityAnBC{A}{B}{C}

Occasionally you'll want to reverse the order, to change emphasis ('d' here is a reminder of what gets divided):

FractionBdCeqA{one}{two}{three}

FractionQuantityBdCeqA{A}{B}{C}



Common four term pattens are supported, where 'eq' denotes the location of the equality.

ProductABeqCD{one}{two}{three}{four}

FractionAdBeqCdD{one}{two}{three}{four}

And one six-term item:

```
FractionAdBeqCDdEF{one}{two}{three}{four}{five}{six}
```

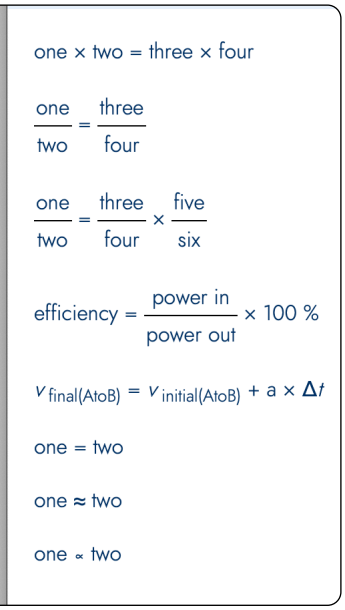
A couple of specials, for calculating efficiency and expressing accumulations:

```
EfficiencyCalc{power in}{power out}
AccumulateRelationship{v}{a}{AtoB}
```

And finally, you can just set things equal, approximately equal, or proportional (this, of course, looks after spacing, and prevents equations being split over lines, so its a fundamental building block):

```
EqualityAssertion{one}{two}
ApproxeqAssertion{one}{two}
ProportionAssertion{one}{two}
```

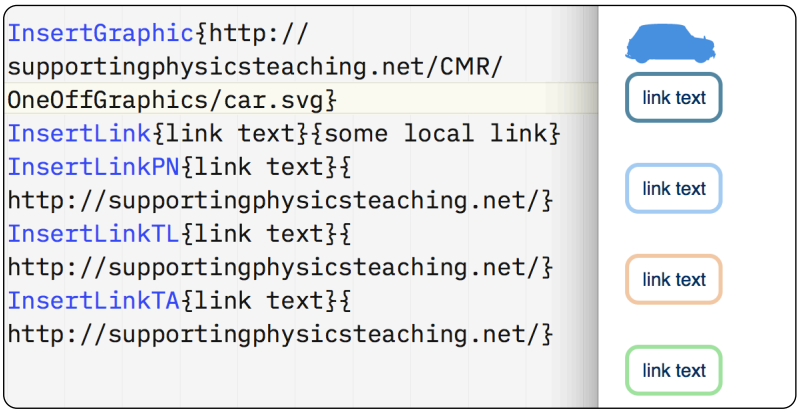
```
ProductABeqCD{one}{two}{three}{four}

FractionAdBeqCdD{one}{two}{three}{four}

FractionAdBeqCDdEF{one}{two}{three}{four}
{five}{six}

EfficiencyCalc{power in}{power out}

AccumulateRelationship{v}{a}{AtoB}

EqualityAssertion{one}{two}

ApproxeqAssertion{one}{two}

ProportionAssertion{one}{two}
```



The aim should be not to have any ad-hoc '=' or other symbols to 'short-cut' the layout, as these will not reliably render physics well when rendered on responsive displays.

Inserting links and graphics

```
InsertGraphic{http://supportingphysicsteaching.net/CMR/OneOffGraph-
ics/car.svg}
InsertLink{link text}{some local link in Drupal}
InsertLinkPN{link text}{ http://supportingphysicsteaching.net/}
InsertLinkTL{link text}{ http://supportingphysicsteaching.net/}
InsertLinkTA{link text}{ http://supportingphysicsteaching.net/}
```



Links are fairly straightforward, but there are many kinds of graphics. For both you need a knowledge of the intended location of the target graphic or linked item to provide the target information.

If you are inserting a specialised graphic, use a terminating suffix to trigger particular parsing actions (adapted from usage in SPT). The one shown above is the fallback. You can ignore the following notes if you are not inserting a specialised graphic.

Filenames terminating in IPSC or IPCC, are assumed to be paned explorable diagrams, will have a hyperlink inserted to an SVG of the same name terminating in EPSC or EPCC, which is explorable.

```
InsertGraphic{MyexplorablediagramIPSC}
```

Filenames terminating in PID are physics interactive diagrams, and need additional information, the width and height of the enclosing iframe, in pixels (these are set in the PID script).

```
InsertGraphic{MyPID}{500}{300}
```

Filenames terminating in DIP are discussion instances in physics.

```
InsertGraphic{MyPID}
```

There are others.

Inserting lists, tables and headings

List items are created as in markdown, and then wrappers style the list

```
- creates a list item
```

Equipment:

```
ListEquipment
- item one
- item two
ListEquipmentEnd
```

Sequence:

```
ListSequence
- item one
- item two
ListSequenceEnd
```

Information:

```
ListInformation
- item one
- item two
ListInformationEnd
```

Similarly, there are end markers for a table: each extra column is denoted by {something}. Tables are for presenting data, not for layout, etc.

```
StartTable
TableHeader{some}{small things}{are for data}
TableRow{first}{second}{1}
TableRow{another}{row}{5}
StopTable
```

```
ListEquipment
- item one
- item two
ListEquipmentEnd

ListSequence
- item one
- item two
ListSequenceEnd

ListInformation
- item one
- item two
ListInformationEnd

StartTable
TableHeader{some}{small things}{are for data}
TableRow{first}{second}{1}
TableRow{another}{row}{5}
StopTable
```

- item one
- item two

1. item one
2. item two

- item one
- item two

some	small things	are for data
first	second	1
another	row	5

Questions

A simple makup for questions with a stem and multiple possible reponses is provided. You can provide the answer and the code to check the answer is inserted in the html. There should be one stem and one answer, but can be any humber of items. Graphics can be inserted using the standard pml elements.

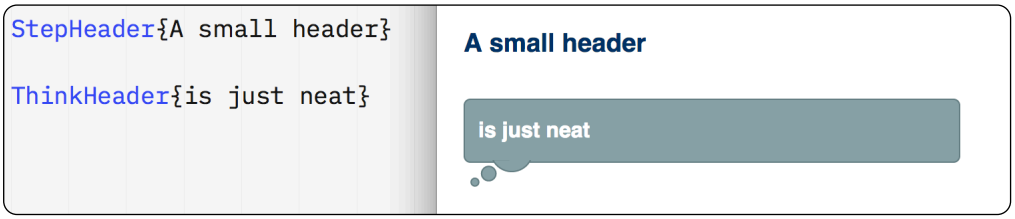
```
QuestionStem{free text, mixing pml elements with free text to ask the question}

QuestionItem{a single capital letter as a marker}{free text, mixing pml elements with free text to provide the possible reponse}

QuestionAnswer{a single capital letter coresponding to the correct answer}
```

In addition there is provision for metadata to be added, but not rendered on the page.

```
QuestionSource{free text}

QuestionID{free text}

QuestionDifficulty{free text}

QuestionSkill{free text}

QuestionTopic{free text}
```

There are two kinds of in-text headers only.

```
StepHeader{A small header}
ThinkHeader{is just neat}
```



Section 3: Forcing spacing

Note that a double line break is a paragraph break, which is standard MarkDown convention. Sometimes inserting a single linebreak before and after a complex expression makes the parsing/editing more reliable, without affecting the parsed output

spacing between lines

SpacingParabreak

SpacingLinebreak

spaces between characters

SpacingNonbreakspace

SpacingThinspace

Section 4: glyphs

These are predefined words, so you can think in physics, rather than in html-entity speak. There is some redundancy (many-to-one mappings), as the same greek letter denotes different physical elements in different contexts.

Technical glyphs

SymbolDelta
SymbolDifferential
SymbolPlusorminus
SymbolPositive
SymbolNegative
SymbolProportion
SymbolEquivalent
SymbolMultipliedby
SymbolMinus
SymbolPlus
SymbolArrowright
SymbolEqual
SymbolApproxequal
SymbolPi
SymbolCopyright
SymbolEndash
SymbolHalf
SymbolQuarter
SymbolAlpha
SymbolBeta
SymbolGamma
SymbolPercent
SymbolDegree
SymbolTemperaturecentigrade
SymbolOhm
SymbolAngle
SymbolAngularv
SymbolAngulara
SymbolWavelength
SymbolDensity
SymbolFlux
SymbolEpsilon
SymbolMu

SymbolSigma

SymbolStrain

SymbolStress

```
SymbolDelta   SymbolDifferential
SymbolPlusorminus   SymbolPositive
SymbolNegative   SymbolProportion
SymbolEquivalent   SymbolMultipliedby
SymbolMinus   SymbolPlus
SymbolArrowright   SymbolEqual
SymbolApproxequal   SymbolPi
SymbolCopyright   SymbolEndash
SymbolHalf   SymbolQuarter   SymbolAlpha
SymbolBeta   SymbolGamma   SymbolPercent
SymbolDegree   SymbolTemperaturecentigrade
SymbolOhm   SymbolAngle   SymbolAngularv
SymbolAngulara   SymbolWavelength
SymbolDensity   SymbolFlux   SymbolEpsilon
SymbolMu   SymbolSigma   SymbolStrain
SymbolStress
```

Δ d ± + − ∝ ≡ × − + → = ≈ π © –
½ ¼ α β γ % ° ℃ Ω θ ω α λ ρ Φ ε μ σ ε σ

Greek letters

Provided just in case, as a fall-back.

UcAlpha

UcBeta

UcGamma

UcDelta

UcEpsilon

UcZeta

UcEta

UcTheta

UcIota

UcKappa

UcLambda

UcMu

UcNu

UcXi

UcOmicron

UcPi

UcRho

UcSigma

UcTau

UcUpsilon

UcPhi

UcChi

UcPsi

UcOmega

```
LcAlpha
LcBeta
LcGamma
LcDelta
LcEpsilon
LcZeta
LcEta
LcTheta
LcIota
LcKappa
LcLambda
LcMu
LcNu
LcXi
LcOmicron
LcPi
LcRho
LcSigma
LcTau
LcUpsilon
LcPhi
LcChi
LcPsi
LcOmega
```

```
UcAlpha UcBeta UcGamma UcDelta
UcEpsilon UcZeta UcEta UcTheta UcIota
UcKappa UcLambda UcMu UcNu UcXi
UcOmicron UcPi UcRho UcSigma UcTau
UcUpsilon UcPhi UcChi UcPsi UcOmega
LcAlpha LcBeta LcGamma LcDelta
LcEpsilon LcZeta LcEta LcTheta LcIota
LcKappa LcLambda LcMu LcNu LcXi
LcOmicron LcPi LcRho LcSigma LcTau
LcUpsilon LcPhi LcChi LcPsi LcOmega
```

Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω

Section 5: Commenting

Comments are not rendered, that's the point. They are notes between authors, or notes to self, or editorial suggestions and commands. Nothing is rendered as html. These are simply an implementation of critic markup.

to suggestion making an addition

```
{++ additions ++}
```

to suggest making a subtraction

```
{-- delete this --}
```

just to pass a comment

```
{>> here 2018-01-11 at 10:28 <<}
```

to suggest a substitution

```
{~~ EeCircuitLoopIMCP ~> EeCircuitLoop ~~}
```

to highlight and then to comment on the highlight

```
{== highlight and  ==}{>> comment <<}
```

Section 6: A few examples

```
EqualityAssertion{QuantitySymbol{P} SymbolMultipliedby
QuantitySymbol{V}}{FractionBlock{1}{3} SymbolMulti-
pliedby N SymbolMultipliedby QuantitySymbol{m} Symbol-
Multipliedby AverageBlock{QuantitySuper{c}{2}}}
```

```
EqualityAssertion{QuantitySymbol{f}}{FractionBlock{1}{2
SymbolPi }FractionRoot{QuantitySymbol{k}}{QuantitySym-
bol{m}}}
```

```
FractionABC{ SymbolDensity }{mass}{volume}
```

```
FractionABC{pressure in newton / WordSuper{me-
tre}{2}}{force in newton}{area in WordSuper{metre}{2}}
```

```
FractionABC{acceleration}{ValueUnit{5}{newton}}{ValueU-
nit{0.5}{kilogram}}
```

```
EqualityAssertion{WordSub{force}{gravity}}{Fraction-
Block{QuantitySymbol{G} SymbolMultipliedby QuantitySym-
bol{M} SymbolMultipliedby QuantitySymbol{m}}{WordSu-
per{separation}{2}}}
```

```
EqualityAssertion{energy in the kinetic store}{Fraction-
Block{1}{2}QuantitySymbol{m}QuantitySuper{v}{2}}
```

```
ProductABC{VectorOver{p}}{QuantitySymbol{m}}{Vec-
torOver{v}}.
```

And how they turn out...

$$P \times V = \frac{1}{3} \times \mathrm{N} \times m \times \overline{c^2}$$

$$f = \frac{1}{2\pi}\sqrt{\frac{k}{m}}$$

$$\rho = \frac{\text{mass}}{\text{volume}}$$

$$\text{pressure in newton / metre}^2 = \frac{\text{force in newton}}{\text{area in metre}^2}$$

$$\text{acceleration} = \frac{5 \text{ newton}}{0.5 \text{ kilogram}}$$

$$\text{force}_{\text{gravity}} = \frac{G \times M \times m}{\text{separation}^2}$$

$$\text{energy in the kinetic store} = \frac{1}{2}mv^2$$

$$\vec{p} = m \times \vec{v}.$$