The template language

Templates are Handlebars. The markup is ordinary HTML, and anything in double braces is replaced by something from the report's data before the page is rendered. This sheet shows every part of it that a report actually needs, each one printed beside its own output.

1 The data

Every example below reads from this. It is the report's Sample data, which the designer opens from its header, and it is also what ships in the manifest so a consuming team knows the shape to send. A real render is given the same shape with real values in it.


```
{"report":{"title":"Invoice INS-2026-0418","issued":"2026-04-03T00:00:00"},"customer":{"name":"Layla Haddad","company":"Nomad \u0026 Co \u003CLtd\u003E","vip":true,"onHold":false},"notice":"\u003Cem\u003EPaid in full\u003C/em\u003E","settings":{"currency":"SAR","rate":0.15},"tags":{"region":"Riyadh","channel":"Direct"},"items":[{"code":"SRV-01","name":"Rack server","qty":2,"price":3120},{"code":"NET-14","name":"Switch, 24 port","qty":1,"price":1480.75},{"code":"INS-90","name":"Installation","qty":1,"price":700}],"backorders":[],"groups":[{"name":"Hardware","lines":["Rack server","Switch"]},{"name":"Services","lines":["Installation"]}],"totals":{"net":8420.75,"vat":1263.11,"gross":9683.86}}
```


That block is itself an example: `{{json}}` prints the whole context as JSON, which is the quickest way to see what a template can actually reach.

2 Printing a value

Two braces print a value and escape it, so text from the data can never become markup by accident. Three braces print it as it stands, which is what you want when the value *is* markup — and what you must not use for anything a stranger typed.

Two braces escape, three do not

TEMPLATE	OUTPUT
`{{customer.company}}`	`Nomad & Co <Ltd>`
`{{notice}}`	`<em>Paid in full</em>`
`{{{notice}}}`	*Paid in full*

A name that is not in the data prints nothing — it is not an error

TEMPLATE	OUTPUT
`[{{customer.name}}]`	`[Layla Haddad]`
`[{{customer.nickname}}]`	`[]`

Worth knowing. A misspelt name renders as nothing rather than stopping the report. That is deliberate — a report is built from someone else's data and must not fall over a field that did not arrive — but it does mean a typo shows up as a blank on the page rather than as a complaint. When something is missing from a render, check the spelling against the data above.

3 Paths

A dot walks into an object. A number walks into a list, counting from zero. Both can go as deep as the data does.

Into objects, and into lists by position

TEMPLATE

```
{{customer.name}}
{{totals.gross}}
{{items.0.name}}
{{items.[1].name}}
```

OUTPUT

```
Layla Haddad
9683.86
Rack server
Switch, 24 port
```

lookup, for when the field name is itself a value

TEMPLATE

```
{{lookup tags "region"}}
{{#each items}}{{lookup ../tags "channel"}} {{/each}}
```

OUTPUT

```
Riyadh
Direct Direct Direct
```

A dot cannot hold a variable. `{{tags.region}}` names the field in the template; `lookup` takes it as an argument instead, which is what you need when the field to read is decided by the data rather than by you.

4 Asking a question

`#if` keeps its block when a value is there and is not false, zero or an empty list. `#unless` is the same question asked the other way round. Both take an `else`.

#if with an else

TEMPLATE

```
{{#if customer.vip}}
  Priority handling
{{else}}
  Standard handling
{{/if}}
```

OUTPUT

```
  Priority handling
```

Where the blank lines come from. Look at the output above: a block tag written on a line of its own leaves that line's newline behind when the tag is removed. It is invisible in HTML, which collapses whitespace, and it matters in a `<pre>` like these examples or inside a `<style>`. Section 12 is how to take it out.

#unless, and an empty list counting as nothing

TEMPLATE

```
{{#unless customer.onHold}}
  Not on hold
{{/unless}}
{{#if backorders}}
  Some
{{else}}
  No backorders
{{/if}}
```

OUTPUT

```
  Not on hold
  No backorders
```

5 Repeating

`#each` runs its block once per item. Inside it, the item *is* the context — so its fields are named directly, and `this` means the item itself when it has no fields to name.

A table body, which is what most of this is for

TEMPLATE

```
{{#each items}}{{code}} — {{name}} × {{qty}}
{{/each}}
```

OUTPUT

```
SRV-01 — Rack server × 2
NET-14 — Switch, 24 port × 1
INS-90 — Installation × 1
```

Where you are in the loop

TEMPLATE

```
{{#each items}}
{{@index}} {{name}}{{#unless @last}},{{/unless}}
{{/each}}
```

OUTPUT

```
0 Rack server,

1 Switch, 24 port,

2 Installation
```

@first and @last, for rules that only apply at the ends

TEMPLATE

```
{{#each items}}
{{#if @first}}TOP: {{/if}}{{code}}
{{#if @last}} (last){{/if}}
{{/each}}
```

OUTPUT

```
TOP: SRV-01


NET-14


INS-90
 (last)
```

An each with nothing to do takes the else

TEMPLATE

```
{{#each backorders}}
  {{name}}
{{else}}
  Nothing outstanding
{{/each}}
```

OUTPUT

```
  Nothing outstanding
```

Over an object rather than a list — @key is the field name

TEMPLATE

```
{{#each tags}}
  {{@key}} = {{this}}
{{/each}}
```

OUTPUT

```
  region = Riyadh

  channel = Direct
```

6 Getting back out of a loop

Inside `#each` the names around you are the item's. `../` steps back one level for each time it is written, and `@root` jumps straight to the top whatever the depth.

../ steps out one level, and stacks when loops nest

TEMPLATE

```
{{#each groups}}{{name}}: {{#each lines}}{{this}} of {{../name}}; {{/each}}
{{/each}}
```

OUTPUT

```
Hardware: Rack server of Hardware; Switch of Hardware;
Services: Installation of Services;
```

@root reaches the whole payload from any depth

TEMPLATE

```
{{#each groups}}{{#each lines}}{{this}} — {{@root.customer.name}}
{{/each}}{{/each}}
```

OUTPUT

```
Rack server — Layla Haddad
Switch — Layla Haddad
Installation — Layla Haddad
```

> Prefer `@root` when you mean the top. A `../` counts levels, so wrapping the markup in one more loop later silently changes what it points at. `@root` keeps meaning the same thing.

7 Narrowing the scope

`#with` moves into an object for a block, so a run of fields from the same place can be written without repeating the path to it.

#with

TEMPLATE

```
{{#with totals}}
  Net {{net}}
  VAT {{vat}}
  Due {{gross}}
{{/with}}
```

OUTPUT

```
  Net 8420.75
  VAT 1263.11
  Due 9683.86
```

8 Comparing

Handlebars on its own cannot compare two things — `#if` only asks whether one value is there. These helpers are added by the engine, and they are written *inside* the `#if`, in brackets, which is what a subexpression is.

Helper	Asks
`eq / ne`	the same as / not the same as
`gt / gte`	greater than / greater than or equal to
`lt / lte`	less than / less than or equal to
`and / or / not`	combines the answers of the ones above

A comparison inside an #if

TEMPLATE

```
{{#if (gt totals.gross 5000)}}
  Approval needed
{{else}}
  Within limit
{{/if}}
```

OUTPUT

```
  Approval needed
```

Combining them, and comparing against a literal string

TEMPLATE

```
{{#if (and customer.vip (gte totals.gross 1000))}}VIP, over 1000{{/if}}
{{#if (or customer.onHold (not customer.vip))}}Flagged{{else}}Clear{{/if}}
{{#if (eq settings.currency "SAR")}}Amounts are in riyals{{/if}}
```

OUTPUT

```
VIP, over 1000
Clear
Amounts are in riyals
```

Comparisons work inside a loop, against the item

TEMPLATE

```
{{#each items}}
{{name}}: {{#if (gt qty 1)}}bulk{{else}}single{{/if}}
{{/each}}
```

OUTPUT

```
Rack server: bulk

Switch, 24 port: single

Installation: single
```

9 Formatting numbers and dates

A raw value prints the way the data holds it, which is rarely how it should be read. `number`, `money` and `date` take a .NET format string, and optionally a culture — anything with a hyphen in it is read as the culture.

number, with and without a format

TEMPLATE

```
{{totals.gross}} → {{number totals.gross "N2"}} → {{number totals.gross "N0"}}
{{number settings.rate "P1"}}
```

OUTPUT

```
9683.86 → 9,683.86 → 9,684
15.0 %
```

money, where the culture decides the symbol and where it sits

TEMPLATE

```
{{money totals.gross}} · {{money totals.gross "en-US"}} · {{money totals.gross "de-DE"}}
```

OUTPUT

```
¤9,683.86 · $9,683.86 · 9.683,86 €
```

date, which reads the string in the data and prints it as asked

TEMPLATE

```
{{report.issued}}
{{date report.issued}}
{{date report.issued "d MMMM yyyy"}}
{{date report.issued "dddd" "fr-FR"}}
```

OUTPUT

```
2026-04-03T00:00:00
04/03/2026
3 April 2026
vendredi
```

10 Two helpers for the document itself

json — the context, or any part of it, printed as JSON

TEMPLATE

```
{{json customer}}
```

OUTPUT

```
{"name":"Layla Haddad","company":"Nomad \u0026 Co \u003CLtd\u003E","vip":true,"onHold":false}
```

Why the angle brackets came out as `<`. The json helper escapes `<`, `>` and `&`. That is not cosmetic: this helper's real job is to hand data to chart code inside a `<script>` block, and a value containing `</script>` would otherwise close the tag and break out of it. JavaScript reads the escapes back as the characters they stand for, so nothing is lost.

pageBreak — starts a new sheet where it stands

TEMPLATE

```
{{pageBreak}}
```

OUTPUT

```
(writes a break; nothing to show)
```

A page is not a sheet. A report's pages are separate files, each with its own size and orientation. `pageBreak` works inside one of those, where content runs longer than a sheet and you want to choose where it divides — this reference is a single page that has run onto several sheets, and it could have been broken by hand this way.

11 Notes to yourself, and to nobody

A Handlebars comment is removed before the page is rendered, so it never reaches the document. An HTML comment does reach it — it is invisible on the page but present in the file, which for a report that gets sent to someone is a difference worth keeping in mind.

The two kinds of comment

TEMPLATE

```
A{{! never shipped }}B
C<!-- shipped, just unseen -->D
```

OUTPUT

```
AB
CD
```

12 Whitespace, and printing braces

A tilde inside the braces eats the whitespace next to the tag in the template. It trims what surrounds the tag, never the value itself. And a backslash in front of a tag stops it running at all — which is how every left-hand column on this sheet was written.

A tilde trims the template around the tag

TEMPLATE

```
[
  {{settings.currency}}
]
[
  {{~settings.currency~}}
]
```

OUTPUT

```
[
  SAR
]
[SAR]
```

A backslash prints the tag instead of running it

TEMPLATE

```
\{{customer.name}}
```

OUTPUT

```
{{customer.name}}
```

13 Things that surprise people

What you write	What happens
`{{customre.name}}`	Nothing prints, and the render succeeds. A blank on the page is the only symptom of a misspelt path.
`{{customer.vip}}`	Prints `True` — a boolean comes out as .NET writes it, capital letter and all. Use `#if` to turn it into words worth reading.
`{{notice}}`	Markup in the data is escaped and shows as text. Use three braces only when you know what the value is.
`{{#if items.length}}`	Does not work. A list is already false when it is empty, so `{{#if items}}` is the question you meant.
`{{money 1234.5}}`	Uses the invariant culture unless told otherwise, which is why the currency helper takes one.

> Everything on this sheet is live. The left column is this file with its braces escaped and the right column is this file running, so if an example prints something unexpected, that is genuinely what the engine does with it — not what someone wrote down once and left behind.