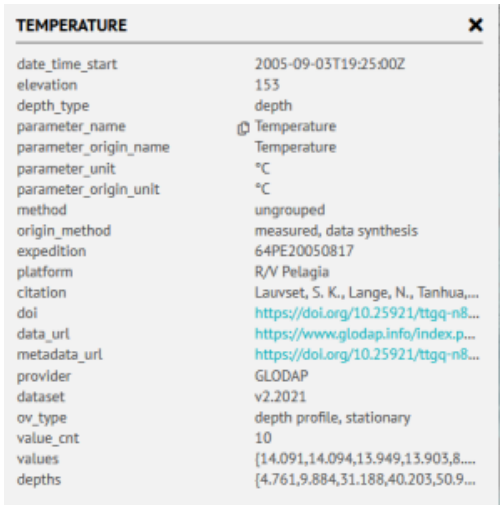
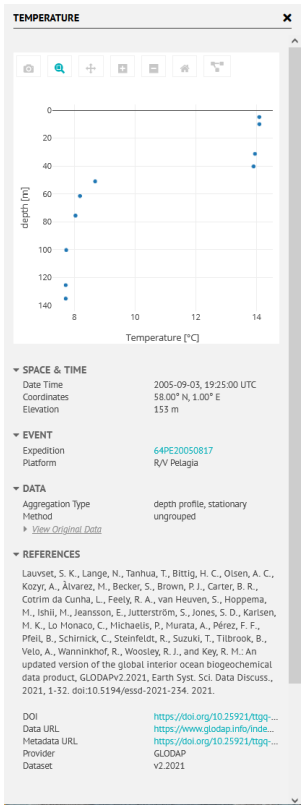
Metadata Templates

VEF-based viewers allow users to style the popup and sidebar metadata that appear when users click on features on the map, improving readability and visualization. Templates are based on the Markdown language and allow for flexible customization of metadata rendering. Predefined function calls allow the use of advanced visualizations, such as displaying charts or a video player. Without the use of templates, the original metadata is displayed in a simple table, using the technical names of each property.

Example of a Styled Sidebar

The following screenshots are taken from a VEF-based viewer showing the metadata of the same clicked geometry on the map.

Unstyled Metadata	Styled Metadata
	

Format Specifications

The metadata templates are based on the lightweight markup language Markdown. Markdown has a lot of flexibility for formatting text and is human-readable in its unparsed form.

The root for the displayed metadata is structured like a GeoJSON feature and contains a **geometry**, **properties**, and **layer** object. To display the metadata properties, they must be enclosed in curly braces.
The metadata is inside an object and can be accessed using dot notation or like dictionary entries.

A predefined list of function calls can be used to perform additional metadata transformations. Function calls must also be enclosed in curly brackets and use round brackets to enclose parameters.
Function parameters can be a string or a reference to a property from the properties, layer, or geometry object. When passing numbers, they must be passed as strings enclosed in quotes.

Notation	Explanation
`{properties.expediton}`	Access the value of the `expedition` property of the clicked feature.
`{properties["general.name"]}`	Access the value of the `event.name` property of the clicked feature. Dot notation cannot be used here because the property name itself contains a dot.
`{layer.name}`	Acces the name of the layer itself.

`{formatDate(properties.date_time_end)}`	Call a custom function to format a date string using the `date_time_end` property of the clicked feature as a parameter.
`{addFilterButton("device", properties.device)}`	Call a custom function with two parameters to display a filter button in the table column that filters by device.

Available Functions

Name	Description
`formatLatLng(latitude, longitude)`	Format lat/lng into a readable string with decimal degrees. `Example: {formatLatLng(properties.begin_latitude, properties.begin_longitude)}`
`formatGeometry(geometry)`	Format the GeoJSON geometry and display it as human-readable coordinates. Works only for POINT geometries. `Example: {formatGeometry(geometry)}`
`addFilterButton(name, value)`	Add a button to request filtering by an attribute. `Example: {addFilterButton("platform", properties.platform)}`
`formatDate(isoDateString)`	Improve ISO timestamp readability. `Example: {formatDate(properties.date_time_start)}`
`formatUnit(value, unit, targetUnit (optional), precision (optional))`	Improve the readability of numbers by simplifying their representation with a unit prefix. `Example: {formatUnit(properties["mediainfo.media.track.Video.BitRate"], "bit/s")}`
`isDefined(valueToCheck, then, else)`	Return the given string only if the first value is not null or undefined. Used as a way to implement conditions. Uses the optional `else` value if `valueToCheck` is null or undefined. `Example: {isDefined(properties.date_time_end, "Date Time Start", "Date Time")}`
`concatIfDefined(concat1, concat2, ...)`	Concatenate values if each value is defined (not null or undefined). `Example: {concatIfDefined(properties.elevation, " m")}`
`encodeURLComponent(urlComponent)`	Encode a string to URL-compatible characters. `Example: {encodeURLComponent("PS133/2_0_Underway-6")}`
`round(number, decimals)`	Round a number to the specified decimal places. `Example: {round("1.12345", "3")}`
`splitString(string, substring, index)`	Split a string at a substring and return the given index. `Example: {splitString("Hello World", " ", "1")}`
`substring(string, startIndex, endIndex)`	Split a string at an index and get the substring. `Example: {substring("Hello World", "0", "5")}`
`setTitle(str1, str2, ...)`	Define the title for the entire popup/sidebar content. This title will be displayed in the header of the popup/sidebar `Example: {setTitle("Event: ", properties.event)}`
`parseJSON(jsonstring, key1, key2, ...)`	Parse a JSON string and return the value of the given property identified by the list of keys. `Example: {parseJSON('{"outer": {"inner": "Hello World"}}', "outer", "inner")}`
`noCopy()`	Prevent the copy button from being displayed in a table row.
`hideOnSiteStart(site)` `hideOnSiteEnd(site)`	Prevent content from being rendered by inserting a comment when the page matches. These functions must be used together to mark an area to be removed. `Example: {hideOnSiteStart("gallery")} ..... {hideOnSiteEnd("gallery")}`

`parseMarkdown(text)`	Parse Markdown text to HTML. Useful when text from an attribute/property is formatted as markdown. `Example: {parseMarkdown(layer.abstract)}`
`joinArray(array, seperator)`	Join array values as a string. Default separator is a comma (`,`). `Example: {joinArray(array, " - ")}`
`displayLayerLegend()`	Display the active layer's legend graphic.
`displayImage(src, width (optional), height (optional), srcset (optional))`	Display an image. Clicking on an image opens it in a full-screen, zoomable lightbox. `Example: {displayImage(properties.picture, properties.width, properties.height, properties.srcset)}`
`displayVideo(media, width, height, thumbnail (optional), cover (optional), poster (optional))`	Display a video in a video player. The following arguments are supported: "media" (URL to media file), thumbnail (URL to webvtt file for seekbar thumbnails), cover (URL to video cover) and poster (URL to background poster on video player loading). They contain the respective URLs. `Example: {displayVideo(properties.video, properties.width, properties.height, properties.thumbnail, properties.cover)}`
`displayVideoPlayerKeys()`	Display a link to open the documentation of video player hotkeys.

HTML Post-Processing

The resulting HTML markup is modified by the VEF-based viewer after the templates have been evaluated for optimal visualization of the metadata.

- Headlines are grouped into collapsible sections. Using ### for headlines in recommended.
- Empty collapsible groups are not rendered
- Copy button automatically added to table rows to copy a property value
- Removes table rows with empty content in the second cell/column

Recommendations

- The content should be organized vertically as pop-ups and the sidebar have limited width.
- Ideally, metadata is structured in a two-column table, as in the following example

Example Template

Template for Expedition Tracklines

```
{isDefined(properties.attachment_url, "<h3>Cruise Overview</h3>")}
{displayImage(properties.attachment_url)}

### Space & Time
| | |
|--|--|
| {isDefined(properties.date_time_end, "Date Time Start", "Date Time")} | {formatDate(properties.
date_time_start)} |
| Date Time End | {formatDate(properties.date_time_end)} |
| Coordinates | {formatGeometry(geometry)} |
| Elevation | {concatIfDefined(properties.elevation, " m")} |

### Event
| | |
|--|--|
| Expedition | [{properties.expedition}](https://marine-data.de/?site=expedition&expedition={encodeURLComponent
(properties.expedition)}") {addFilterButton("expedition", properties.expedition)} |
| Expedition Alias | {properties.expedition_alias} |
| Event | [{properties.event}](https://marine-data.de/?site=data&qf=events.name/{encodeURLComponent(properties.
event)}") {addFilterButton("event", properties.event)} |
| Event Alias | {properties.event_alias} |
| Platform | {properties.platform} {addFilterButton("platform", properties.platform)} |
| Device | {properties.device} {addFilterButton("device", properties.device)} |
| Sensor URI | [{properties.sensor_uri}]({properties.sensor_uri}) |

### Data
| | |
|--|--|
| Aggregation Type | {properties.ov_type} {addFilterButton("ov_type", properties.ov_type)} |
| Current Velocity | {getMarehubOverviewValue("values_mag")} |
| Current Direction |  {getMarehubOverviewValue("values_dir", "°")} |
| {isDefined(properties.parameter_name, properties.parameter_name, "Value")} | {getMarehubOverviewValue()} |
| Method | {properties.method} {addFilterButton("method", properties.method)}|
| {isDefined(properties.depth_type, properties.depth_type, "Depth")} | {getMarehubOverviewValue("depths", "m")}
|

#### View Original Data
| | |
|--|--|
| Original Parameter Name | {properties.parameter_origin_name} |
| Original UC Name | {properties.uc_origin_name} |
| Original VC Name | {properties.vc_origin_name} |
| Original Unit | {properties.parameter_origin_unit} |
| Original Method | {properties.origin_method} |

### References
{properties.citation}
| | |
|--|--|
| License | {properties.license} |
| DOI | [{properties.doi}]({properties.doi}) |
| Data URL | [{properties.data_url}]({properties.data_url}) |
| Metadata URL |[{properties.metadata_url}]({properties.metadata_url}) |
| SOP |[{properties.sop_url}]({properties.sop_url}) |
| Provider | {properties.provider} {addFilterButton("provider", properties.provider)} |
| Dataset | {properties.dataset} |
| Curator | {properties.curator} |
```