Article
José Pereira · Oct 23, 2021 4m read

Open Exchange

# A brief introduction on how to draw diagrams with mermaid library

Hello everyone!

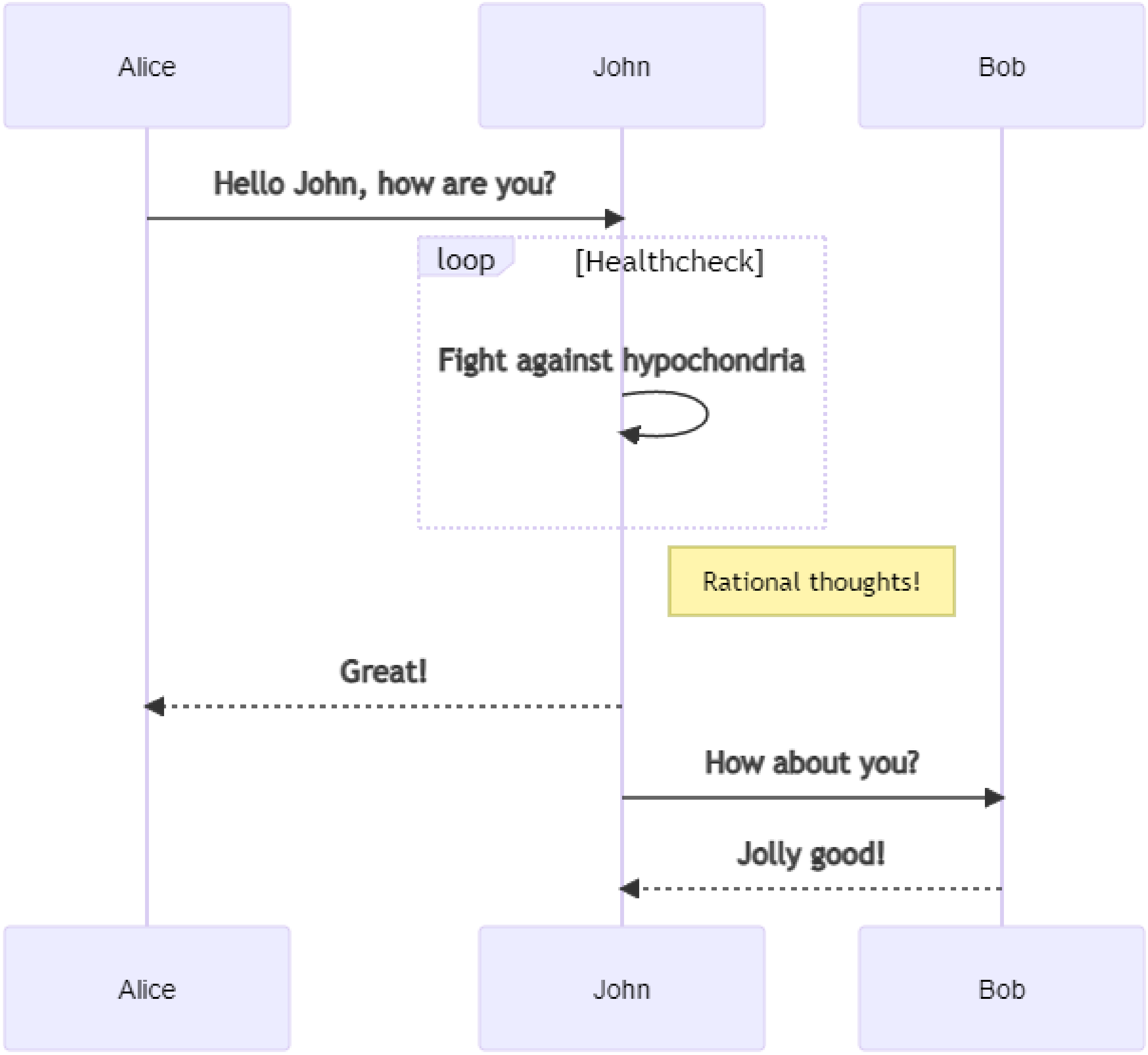
Me and @Henrique.GonçalvesDias proposed a new way to visualize messages in IRIS Interoperability in a recent update of MessageViewer. In such an update, we tried to give users a visualization based on a UML sequence diagram. You could get more information on the previous article.

In order to get all the hard geometry calculations needed to draw such a diagram done, we used the amazing mermaid JS open source library. And what I ’ d like to share with you in this article, is how to use this library. I ’ ll focus just on the sequence diagram, but be aware that such a library lets you do much more.

Mermaid uses a Markdown-inspired syntax to define diagrams. It ’ s super intuitive, so guess it ’ s better to show you an example instead of writing a lot of boring text:

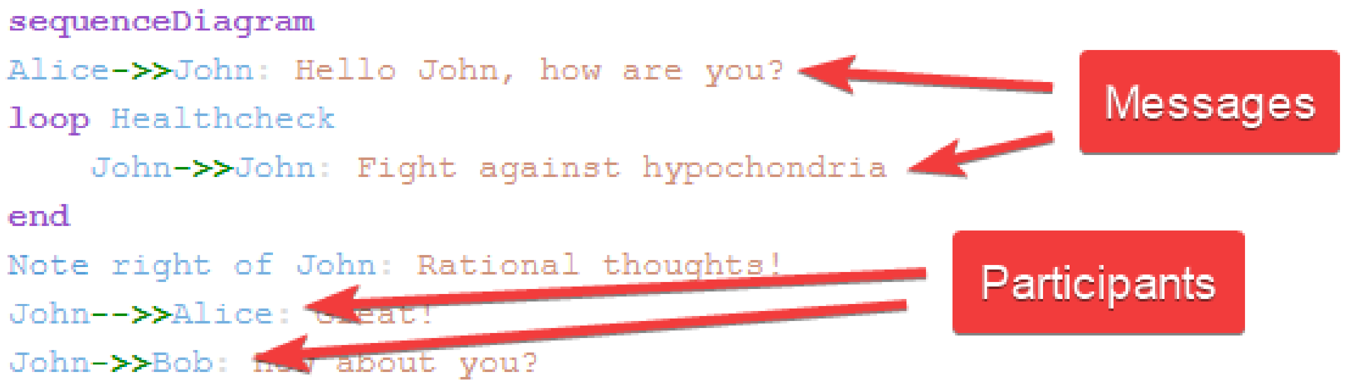
```
sequenceDiagram
Alice->>John: Hello John, how are you?
loop Healthcheck
    John->>John: Fight against hypochondria
end
Note right of John: Rational thoughts!
John-->>Alice: Great!
John->>Bob: How about you?
```

This definition lets mermaid engine to render the following diagram, directly in a web page using SVG:



Such a example was retrieved from mermaid documentation, and you can try it in this online editor. There are a lot of configurations that you can play on it.

As you can see, in the diagram definition you just need to specify the actors/participants and what events/messages they send each other.



```
sequenceDiagram
Alice->>John: Hello John, how are you?
loop Healthcheck
    John->>John: Fight against hypochondria
end
Note right of John: Rational thoughts!
John-->>Alice: Great!
John->>Bob: How about you?
```

And all that you need to have the diagram in your web page, are a div container with the diagram specification and, a JS code which initializes the mermaid engine and renders the diagram.

```
<div class="mermaid">
sequenceDiagram
Alice->>John: Hello John, how are you?
loop Healthcheck
    John->>John: Fight against hypochondria
end
Note right of John: Rational thoughts!
John-->>Alice: Great!
John->>Bob: How about you?
</div>
```

```
mermaid.initialize({
    startOnLoad: true,
    theme: 'forest'
});
```

You can find this example in this fiddle.

This is the frontend base of the proposed work. For the backend, all we have to do is setting up a REST endpoint which retrieves messages from an IRIS interoperability session, format it in a suitable JSON object and send it back to the frontend. As the focus of this article is the frontend code, I won ’ t pay attention on backend implementation, but you can check it out in dispatch and service classes.

The backend sends back a JSON like this:

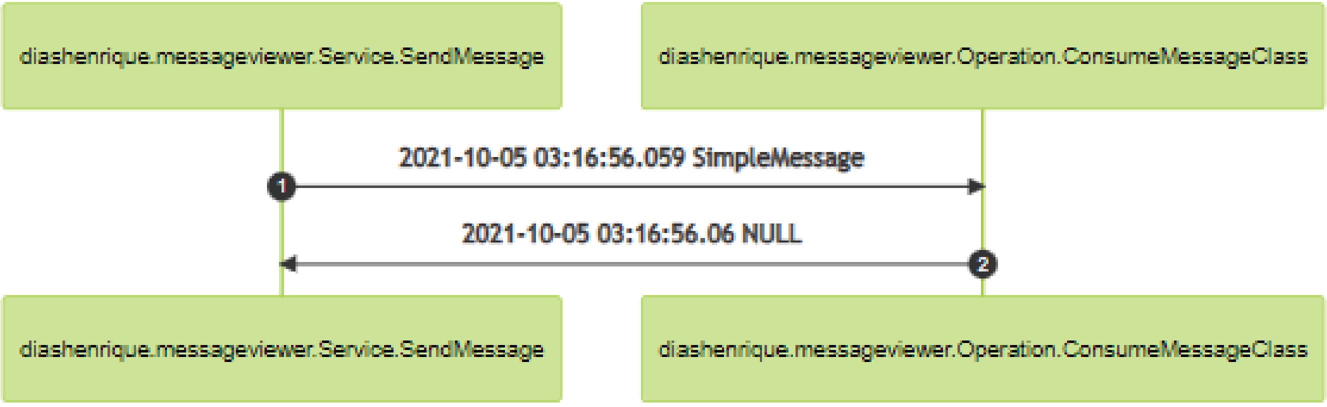
```
{
    "participants":[
        "diashenrique.messageviewer.Service.SendMessage",
        "diashenrique.messageviewer.Operation.ConsumeMessageClass"
    ],
    "messages":[
        {
            "id":"1182",
            "from":"diashenrique.messageviewer.Service.SendMessage",
            "to":"diashenrique.messageviewer.Operation.ConsumeMessageClass",
            "message":"2021-10-05 03:16:56.059 SimpleMessage"
        },
        {
            "id":"1183",
            "from":"diashenrique.messageviewer.Operation.ConsumeMessageClass",
            "to":"diashenrique.messageviewer.Service.SendMessage",
            "message":"2021-10-05 03:16:56.06 NULL"
        }
    ]
}
```

Finally, with simple JS functions you can transform this JSON in a mermaid sequence diagram specification, like this:

```
sequenceDiagram
autonumber
```

```
participant P0 as diashenrique.messageviewer.Service.SendMessage
participant P1 as diashenrique.messageviewer.Operation.ConsumeMessageClass
P0->>P1: 2021-10-05 03:16:56.059 SimpleMessage
P1->>P0: 2021-10-05 03:16:56.06 NULL
```

And this is the rendered sequence diagram:


diashenrique.messageviewer.Service.SendMessage
diashenrique.messageviewer.Operation.ConsumeMessageClass
2021-10-05 03:16:56.059 SimpleMessage
1
2021-10-05 03:16:56.06 NULL
2
diashenrique.messageviewer.Service.SendMessage
diashenrique.messageviewer.Operation.ConsumeMessageClass

You can check out the complete JS code here.

So, that is it. I hope this article could bring to you something useful that can help you in your amazing projects.

See you!

#Frontend #Interoperability #JavaScript #InterSystems IRIS
Check the related application on InterSystems Open Exchange

**Source URL:**https://community.intersystems.com/post/brief-introduction-how-draw-diagrams-mermaid-library