

# From a customer request to a DDC v1.0



# The customer request

I need to show a long text HTML formatted in VA and list table does simply not cut it

```
<h1>Rutiner for personlig pleje </h1>

<p><strong>Pellentesque habitant morbi tristique</strong> senectus et netus et malesuada fames ac turpis egestas. Vestibulum to
Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, <code>commodo vitae</code>, orn
facilisis. Ut felis.</p>

<h2>Header Level 2</h2>

<ol>
  <li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
  <li>Aliquam tincidunt mauris eu risus.</li>
</ol>

<blockquote><p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus magna. Cras in mi at felis aliquet congue. Ut a e
est.</p></blockquote>

<h3>Header Level 3</h3>

<ul>
  <li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
  <li>Aliquam tincidunt mauris eu risus.</li>
</ul>

<h1>Mere eksempel tekst</h1>
<p>Eget velit aliquet sagittis id consectetur purus ut faucibus. Amet cursus sit amet dictum sit amet justo donec. Integer quis aucto
<p>Vulputate enim nulla aliquet porttitor lacus luctus. Lorem ipsum dolor sit amet consectetur adipiscing. Amet porttitor eget dolc
hendrerit. Interdum posuere lorem ipsum dolor sit amet.</p>
<p>Nulla facilisi cras fermentum odio eu feugiat pretium nibh. Vitae aliquet nec ullamcorper sit amet. Augue eget arcu dictum vari
```

# I found this on the internet:

[https://www.w3schools.com/howto/howto\\_js\\_tabs.asp](https://www.w3schools.com/howto/howto_js_tabs.asp)

The screenshot shows a web browser displaying the w3schools.com website. The URL in the address bar is `w3schools.com/howto/howto_js_tabs.asp`. The website has a dark navigation bar with various technology categories like HTML, CSS, JAVASCRIPT, SQL, PYTHON, JAVA, PHP, HOW TO (highlighted), W3.CSS, C, C++, C#, BOOTSTRAP, REACT, MYSQL, JQUERY, EXCEL, XML, DJANGO, NUMPY, PANDAS, and NODE.JS. A search bar is located in the top right. The main content area features a sidebar with a search bar and a list of 'HOW TO' topics, with 'Tabs' selected. The main content area has a title 'How TO - Tabs', a 'Previous' button, and a 'Next' button. Below the title, there is a sub-header 'Learn how to create tabs with CSS and JavaScript.' and a 'Workbook last saved: Just now' notification. The main content area contains a section titled 'Tabs' with a description: 'Tabs are perfect for single page web applications, or for web pages capable of displaying different subjects:'. Below this, there is a code example showing a tabbed interface with three tabs: 'London', 'Paris', and 'Tokyo'. The 'London' tab is active, displaying the text 'London is the capital city of England.'. A 'Try it Yourself »' button is located below the code example.

# The full code

[https://www.w3schools.com/howto/tryit.asp?filename=tryhow\\_js\\_tabs](https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_tabs)

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial;}

/* Style the tab */
.tab {
  overflow: hidden;
  border: 1px solid #ccc;
  background-color: #f1f1f1;
}

/* Style the buttons inside the tab */
.tab button {
  background-color: inherit;
  float: left;
  border: none;
  outline: none;
  cursor: pointer;
  padding: 14px 16px;
  transition: 0.3s;
  font-size: 17px;
}

/* Change background color of buttons on hover */
.tab button:hover {
  background-color: #ddd;
}

/* Create an active/current tablink class */
.tab button.active {
  background-color: #ccc;
}

/* Style the tab content */
.tabcontent {
  display: none;
  padding: 6px 12px;
  border: 1px solid #ccc;
  border-top: none;
}
</style>
</head>
```

Appearance



```
<body>
<h2>Tabs</h2>
<p>Click on the buttons inside the tabbed menu:</p>

<div class="tab">
  <button class="tablinks" onclick="openCity(event, 'London')">London</button>
  <button class="tablinks" onclick="openCity(event, 'Paris')">Paris</button>
  <button class="tablinks" onclick="openCity(event, 'Tokyo')">Tokyo</button>
</div>

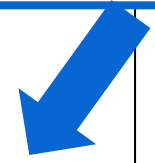
<div id="London" class="tabcontent">
  <h3>London</h3>
  <p>London is the capital city of the United Kingdom.</p>
</div>

<div id="Paris" class="tabcontent">
  <h3>Paris</h3>
  <p>Paris is the capital of France.</p>
</div>

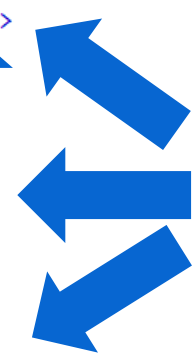
<div id="Tokyo" class="tabcontent">
  <h3>Tokyo</h3>
  <p>Tokyo is the capital of Japan.</p>
</div>

<script>
function openCity(evt, cityName)
{
  var i, tabcontent, tablinks;
  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
  }
  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace(" active", "");
  }
  document.getElementById(cityName).style.display = "block";
  evt.currentTarget.className += " active";
}
</script>
</body>
</html>
```

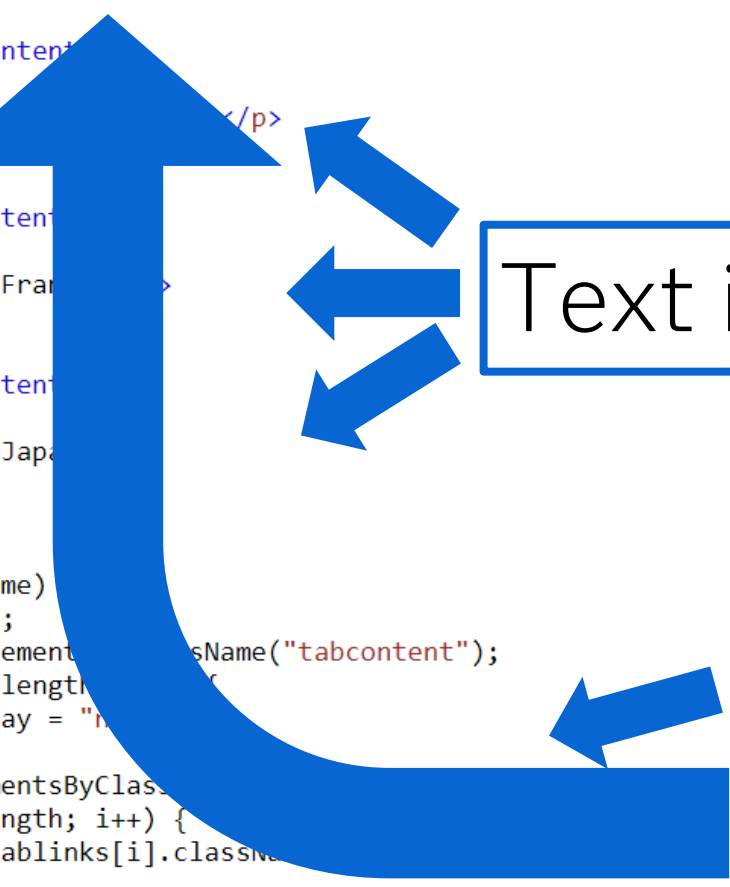
3 buttons defined



Text in tabs



Code that hides everything and afterwards show the text for the selected button



# How to build the html on-the-fly

<https://www.shecodes.io/athena/126003-how-to-use-innerhtml-in-javascript-to-update-or-append-html-content>

shecodes.io/athena/126003-how-to-use-innerhtml-in-javascript-to-update-or-append-html-content

Check out our NEW program **SheCodes Bootcamp** - and become a developer in 4 m

les

**SheCodes Athena says:** Answered in 4.79 seconds

The `innerHTML` property is used to access or modify the HTML content of an element. It is commonly used in JavaScript to update the content of an element dynamically.

You can use `innerHTML` in combination with a JavaScript variable or a string to set the HTML content of an element. Here's an example:

```
javascript Copy code
// HTML
<div id="myElement"></div>

// JavaScript
var myElement = document.getElementById("myElement");
myElement.innerHTML = "<p>This is the new content.</p>";
```

In this example, `getElementById` is used to select the element with the id "myElement". Then, the `innerHTML` property is used to set the HTML content of that element to the string "

This is the new content.

". As a result, the content of the div element will be replaced with the new HTML.

It's important to note that when using `innerHTML`, any existing content within the element will be overwritten. If you want to append or add new content instead, you can use the `+=` operator like this:

```
javascript Copy code
myElement.innerHTML += "<p>This is an additional paragraph.</p>";
```

Note use of += is not best practice (but it works)

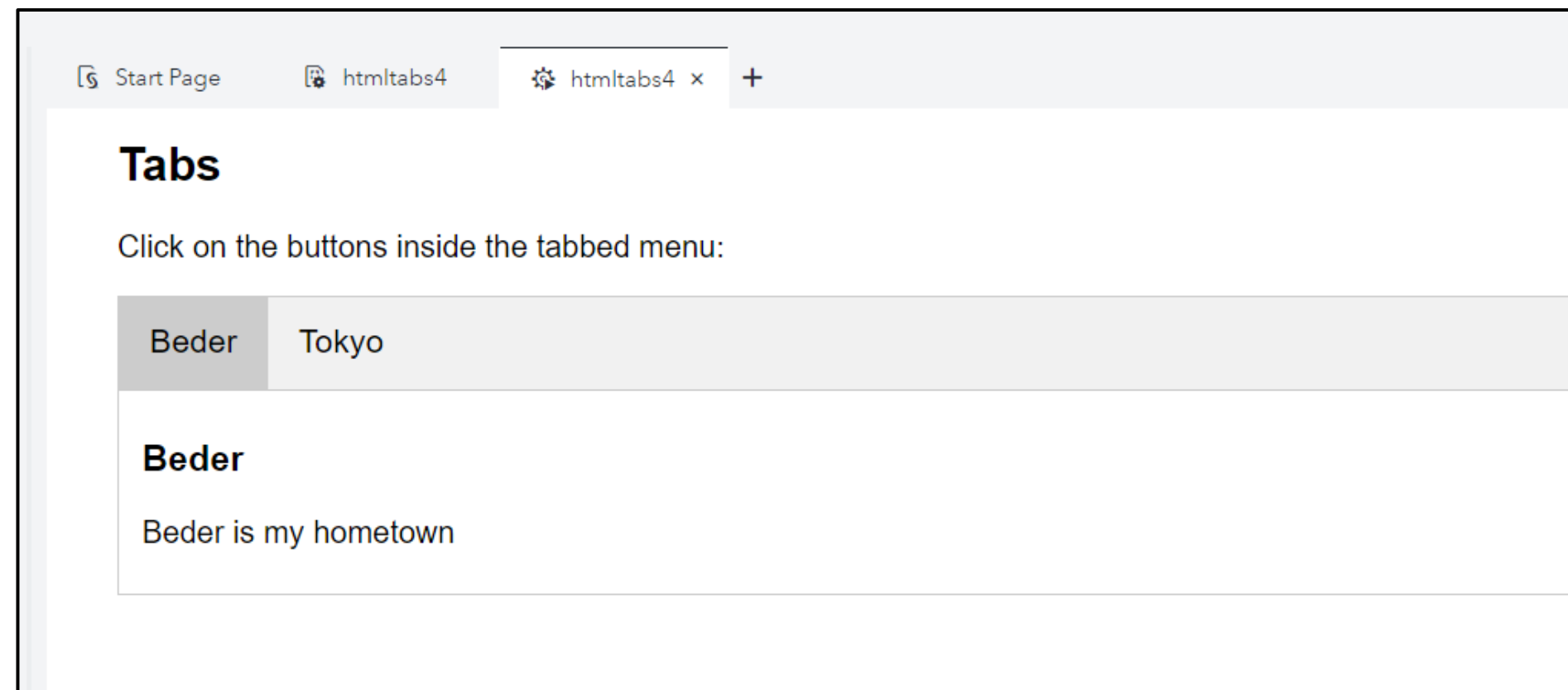
Refer to this instead

<https://stackoverflow.com/questions/11515383/why-is-element-innerhtml-bad-code>

# Applying Stack overflow to the html tab object

## Dynamically create the tabs

```
function addtabs2() {  
  console.log('adding tabs');  
  document.getElementById("tabbuttons").innerHTML += '<button class="tablinks" onclick="openCity(event, \'' + "Beder" + '\")">Beder</button>';  
  document.getElementById("tabbuttons").innerHTML += '<button class="tablinks" onclick="openCity(event, \'' + "Tokyo" + '\")">Tokyo</button>';  
  document.getElementById("cnt").innerHTML += '<div id="Beder" class="tabcontent"><h3>Beder</h3><p>Beder is my hometown</p></div>';  
  document.getElementById("cnt").innerHTML += '<div id="Tokyo" class="tabcontent"><h3>Tokeo</h3><p>Tokyo is the capital of Japan.</p></div>';  
  
  console.log('adding tabs...DONE');  
}  
  
</script>
```





# Data Driven Content, the basics

## Utility script and the onDataReceived function

```
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script type="text/javascript" src="/SASJobExecution/?_file=/Public/PoCs/UpdateObjPOC/js_utilities/messagingUtil.js"></script>
  <script type="text/javascript" src="/SASJobExecution/?_file=/Public/PoCs/UpdateObjPOC/js_utilities/contentUtil.js"></script>
</head>
</html>
```

Download from  
[SAS Visual Analytics third party visualizations](#)

Copy from one of  
the samples

```
async function onDataReceived(resultData)
{
  if (!resultData) return;

  _resultData = resultData;
  _resultName = resultData.resultName;

  let selections = va.contentUtil.initializeSelections(resultData); // good practice to remove eventual brush columns
  if (resultData.columns.length == 0) {
    // it needs at least one column and one row
    document.getElementById("JobResults").innerHTML = "";
    va.messagingUtil.postInstructionalMessage(
      _resultName,
      "Please, make sure at least one data item is assigned to this object's Role pane:\n"+
      "Parameter _job_wait_refresh is optional and sets the waiting time for refresh"
    );
    return;
  }
  resultData.data.forEach((element, index) => {

    // USE THE DATA PASSED TO DDC

  });
}
```

# Combine web page and the DDC skeleton

The JavaScript libraries from SAS

```
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script type="text/javascript" src="/SASJobExecution/?_file=/Public/js_utilities/contentUtil.js"></script>
  <script type="text/javascript" src="/SASJobExecution/?_file=/Public/js_utilities/messagingUtil.js"></script>
</head>
<body>
  <h2>Forklarende tekst for valgt enhed</h2>
  <p>Click on the buttons inside the tabbed menu:</p>
  <div id="tabbuttons" class="tab">
  </div>
  <div id="cnt" >
  </div>
  <script>
    "use strict";

    // Global variables
    let _resultData = null;
    let _resultName = null;
    let _optionalNode = null;

    async function onDataReceived(resultData)
    {
      var tablinks;
      if (!resultData) return;

      _resultData = resultData;
      _resultName = resultData.resultName;

      //var json = syntaxHighlight(resultData);

      let selections = va.contentUtil.initializeSelections(resultData); // good practice to remove eventual brush columns
      if (resultData.columns.length == 0) {
        // it needs at least one column and one row
        document.getElementById("JobResults").innerHTML = "";
        va.messagingUtil.postInstructionalMessage(
          _resultName,
          "Needs data, add tab title as first role and tabtext as second role\n"
        );
        return;
      }

      resultData.data.forEach((element, index) => {

        // BUILD WEB PAGE USING YOUR DATA, referencing the columns element[0], e

      });
    }

    function openCity(evt, cityName) {
      var i, tabcontent, tablinks;
    }
  </script>
</body>
</html>
```

Web page with placeholders for buttons and content, original hardcoded content is removed

The OnDataRecievedFunction

NEXT we are going to put the logic to build buttons and tabs here

The JavaScript to make a specific button visible, it does not care what the ID's of the tags are, so no changes needed here



# Adding the logic using the data passed to the DDC

```
async function onDataReceived(resultData)
{
  var tablinks;
  if (!resultData) return;

  _resultData = resultData;
  _resultName = resultData.resultName;

  //var json = syntaxHighlight(resultData);

  console.log(resultData);

  let selections = va.contentUtil.initializeSelections(resultData); // good practice to remove eventual brush columns
  if (resultData.columns.length == 0) {
    // it needs at least one column and one row
    document.getElementById("JobResults").innerHTML = "";
    va.messagingUtil.postInstructionalMessage(
      _resultName,
      "Needs data, add tab title as first role and tabtext as second role\n"
    );
    return;
  }

  idx=1;
  ididx="";
  document.getElementById("tabbuttons").innerHTML='';
  document.getElementById("cnt").innerHTML='';
  resultData.data.forEach((element, index) => {
    ididx="id"+idx;
    document.getElementById("tabbuttons").innerHTML += '<button class="tablinks" onclick="openCity(event, \''+ididx+'\'>'+element[0]+'</button>';
    document.getElementById("cnt").innerHTML += '<div id="'+ididx+'" class="tabcontent">'+element[1]+'</p></div>';
    idx=idx+1;
  });
  document.getElementById('id1').style.display = "block";
  tablinks = document.getElementsByClassName("tablinks");
  tablinks[0].className += " active";
}
```

Loop over data and insert buttons and tab content named id1, id2, id3...

Make the first one active and the content visible

# Move the code to Viya 3.5

/opt/sas/viya/home/var/www/html/htmlcommons/tabs

 contentUtil.js messagingUtil.js textintabs.html

References in html changed to

```
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script type="text/javascript" src="./messagingUtil.js"></script>
  <script type="text/javascript" src="./contentUtil.js"></script>
</head>
<body>
</body>
</html>
```

URL used in data driven object in the options menu

[http://\[Viya-server\]/htmlcommons/tabs/textintabs.html](http://[Viya-server]/htmlcommons/tabs/textintabs.html)



