File AsyncAwaitPromise.docx Gunnar 2024-02-19

**Asynchronous function setTimeout**

A good example of asynchronous code is the case when the function setTimeout is used, i.e. setTimeout is an asynchronous function. Like starting another parallel process. After the set time it will call the defined function.

If there are code lines after the setTimeout statement they will be executed without waiting for the setTimeout function to finish.

In some of my applications there is some ugly used setTimeout statements. It stops the code for a while waiting for another asynchronous function to finish, like for instance loading a picture (element <image> src= …). This case can be solved with an event listener or like in the photo shows of the Homepage application. Before starting the photo show all images are loaded. TODO Verify in my code

The setTimeout statement is based on the callback principle. When an asynchronous process has finished a function (the callback function) will be called.

**Asynchronous functions object XMLHttpRequest**

In my web applications the class XMLHttpRequest is used load XML files. With a member functions the URL of the file is defined and there is also an event member function ***onreadystatechange*** that gets fired when the file has been loaded, i.e. an anonymous function will be called. This function is inside the function that instantiates XMLHttpRequest and member functions and attributes are available like for instance ***readyState*** and ***status***. For the jazz web applications almost only XML files are loaded and then an XML object is created with the XMLHttpRequest member function ***responseXML***.

From here comes the well-known error message (status=) 404 when the file can’t be loaded.

The anonymous function in the jazz applications aways calls a ***callback-function***. For the Homepage all season programs XML file are recursively called

**Asynchronous jQuery function $.post**

jQuery is a huge library of JavaScript functions. With the $ sign and a point is a function of the library called. The function ***post*** is one of them.

After the $ sign normally a selector is set to define the HTML elements that the function shall perform ‘action’ on. Such functions have not been used in the jazz applications. In other applications probably hide, fade, slide, ,, are used.

jQuery function post is an AJAX (Asynchronous JavaScript and XML) function that loads data in the background. AJAX functions are differently implemented in different browsers. Therefore the use of jQuery makes sense.

Another AJAX function ***load*** could have been used in the jazz applications to load an HTML element like for instance a <div> with the content of a text file.

The jQuery ***$.get*** is an alternative to the XMLHttpRequest function described above for requesting data from the server.

The jQuery function ***$.post*** submits data that shall be processed by PHP functions on the server. In the jazz applications this function is used to create (save XML) files where the submitted data is the content of the created file and the URL for the file. Also moving, deleting and copying files are made with the $.post function.

Syntax ***$.post(URL, data, callback);***

Parameter ***URL*** is the requested PHP file that processes the data

The optional parameter ***data*** are object properties written as name:value pairs separated by commas within curly braces {}. Example:

person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

The optional parameter ***callback*** is the name of the function that will be called when the data has been processed by the PHP function. The callback function shall be defined with two parameters ***(exec\_data, exec\_status)***. The ***exec\_data*** contains data (text) written with the PHP function ***echo***. The ***exec\_status*** is a string and in the jazz applications it is checked that it has the value ***success***. But please note that success means that the PHP function could be executed. It does not mean that for instance that a file could be created. In the jazz applications therefore the ***exec\_data*** is used to send own defined error codes. There must be some better way …. TODO

The syntax is simple: $.post(URL,data,callback); But the implementation looks very complex since the callback function is implemented as an anonymous function. There are so many parentheses, commas and curly braces…

Important to note is $.post that it is an asynchronous function that runs in the background. If there are other statements after the call of $.post these will be executed without waiting for $.post to finish.

In the jazz applications the $.post functions is not called directly. There is always another function calling it. Please refer to class UtilServer.

The function UtilServer.saveFile returns true if the file was successfully created. This return value is actually not meaningful and must be implemented with async/await, i.e. the anonymous function sets a parameter and this parameter is returned. This option can be used in some cases but probably seldom. The normal use is that there is a callback function.

https://www.w3schools.com/js/js\_async.asp

***References***

Promises

https://www.freecodecamp.org/news/synchronous-vs-asynchronous-in-javascript/

async await

https://www.google.com/search?q=js+await+function&sca\_esv=599429746&biw=2048&bih=983&tbm=vid&sxsrf=ACQVn0\_ssfXI7YzLTAxscTnI21XNXw8eHg%3A1705584597157&ei=1SepZa36CLjj7\_UP5vq80AM&oq=js+await&gs\_lp=Eg1nd3Mtd2l6LXZpZGVvIghqcyBhd2FpdCoCCAQyCBAAGIAEGMsBMggQABiABBjLATIIEAAYgAQYywEyCBAAGIAEGMsBMggQABiABBjLATIIEAAYgAQYywEyCBAAGIAEGMsBMggQABiABBjLATIIEAAYgAQYywEyCBAAGIAEGMsBSIxmULEjWJsucAB4AJABAJgBaKABoQSqAQM2LjG4AQHIAQD4AQHCAgQQIxgnwgIFEAAYgATCAgQQABgewgIGEAAYBRgewgIGEAAYFhgewgIKEAAYgAQYigUYQ8ICChAAGIAEGBQYhwKIBgE&sclient=gws-wiz-video#fpstate=ive&vld=cid:e7d37c39,vid:V\_Kr9OSfDeU,st:0

async await

https://www.google.com/search?q=js+await+function&sca\_esv=599429746&biw=2048&bih=983&tbm=vid&sxsrf=ACQVn0\_ssfXI7YzLTAxscTnI21XNXw8eHg%3A1705584597157&ei=1SepZa36CLjj7\_UP5vq80AM&oq=js+await&gs\_lp=Eg1nd3Mtd2l6LXZpZGVvIghqcyBhd2FpdCoCCAQyCBAAGIAEGMsBMggQABiABBjLATIIEAAYgAQYywEyCBAAGIAEGMsBMggQABiABBjLATIIEAAYgAQYywEyCBAAGIAEGMsBMggQABiABBjLATIIEAAYgAQYywEyCBAAGIAEGMsBSIxmULEjWJsucAB4AJABAJgBaKABoQSqAQM2LjG4AQHIAQD4AQHCAgQQIxgnwgIFEAAYgATCAgQQABgewgIGEAAYBRgewgIGEAAYFhgewgIKEAAYgAQYigUYQ8ICChAAGIAEGBQYhwKIBgE&sclient=gws-wiz-video#fpstate=ive&vld=cid:7abf7df8,vid:aP5WH5EdCoc,st:0

Callback, promise await and async

https://www.google.com/search?sca\_esv=599405545&sxsrf=ACQVn08GsfwGmJ0wbu3ghoocb7C9IhCwVA:1705566684534&q=js+async+await+tutorial&tbm=vid&source=lnms&sa=X&ved=2ahUKEwiz-vyCw-aDAxUh-QIHHalFDmIQ0pQJegQIDBAB&biw=2048&bih=983&dpr=1.25#fpstate=ive&vld=cid:c6f9d3bb,vid:ZYb\_ZU8LNxs,st:0