```
// This is glue.js
// This filename is referred to in
D:\Users\privat\Documents\xampp_gf09_wiki\htdocs\wiki\extensions\FormulaApplet\extension.json
// and in D:\Users\privat\Laufwerk_E\gut\gf09\header.php

if (typeof gf09_path == 'undefined') {
  console.log('gf09_path undefined. This should not happen because it is defined in
  header.php or FormulaApplet.body.php');
  // var gf09_path = '/gf09/';
  // //    var server = document.location.hostname;
  // var href = document.location.href;
  // console.log(href);
  // // if (href.startsWith('http://localhost:8080')) {
  // //     gf09_path = '/gf09/';
  // // }
  // if (href.startsWith('https://test.grossmann.info')) {
  //     gf09_path = '/';
  // }
  // if (href.startsWith('http://localhost:8088')) {
  //     gf09_path = '/';
  // }
}
var jsPath = gf09_path + 'js/';
var libPath = jsPath + 'lib/';
var cssPath = gf09_path + 'css/';
console.log('jsPath=' + jsPath + ' libPath=' + libPath + '  cssPath=' + cssPath);

// var gluetest = 'Here is glue!';

if (typeof liblist === 'undefined') {
  // default for wiki
  var liblist = ['mathquill', 'prepare_page', 'tex_parser', 'decode', 'mathquillcss', 'gf09css',
  'vkbd', 'vkbdcss', 'hammer', 'translate'];
}

function task(source) {
  this.name = 'unknown';
  this.source = source;
  this.fallback = null;
  this.css = source.endsWith('.css');
  this.state = 'unused';
}

var jQuery_url = "https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js";
var jQuery_fallback = libPath + "jquery-3.4.1.min.js";
var tasks = {};
tasks['mathquillcss'] = new task(
'https://cdnjs.cloudflare.com/ajax/libs/mathquill/0.10.1/mathquill.css');
tasks['mathquillcss'].fallback = libPath + 'mathquill-0.10.1/mathquill.css';
tasks['mathquill'] = new task(
'https://cdnjs.cloudflare.com/ajax/libs/mathquill/0.10.1/mathquill.js');
tasks['mathquill'].fallback = libPath + 'mathquill-0.10.1/mathquill.js';
tasks['algebrite'] = new task('http://algebrite.org/dist/1.2.0/algebrite.bundle-for-browser.js'
);
tasks['algebrite'].fallback = libPath + 'Algebrite/dist/algebrite.bundle-for-browser.js';
tasks['kas'] = new task(libPath + 'KAS/KAS_loader.js');
tasks['hammer'] = new task(libPath + 'hammer.js');
tasks['hammer'].fallback = 'https://hammerjs.github.io/dist/hammer.js';
```

```
//
tasks['tex_parser'] = new task(jsPath + 'tex_parser.js');
tasks['vkbd'] = new task(jsPath + 'vkbd.js');
tasks['decode'] = new task(jsPath + 'decode.js');
tasks['translate'] = new task(jsPath + 'translate.js');
tasks['prepare_page'] = new task(jsPath + 'prepare_page.js');
tasks['gf09css'] = new task(cssPath + 'gf09.css');
tasks['vkbdcss'] = new task(cssPath + 'vkbd.css');
// tasks['tap4'] = new task(libPath + 'tap4.js');
// console.log(tasks);
// console.log(tasks);
// .forEach causes error 'foeEach is not a function' - maybe typescript error
// liblist.forEach(function (taskname) {
//     tasks[taskname].name = taskname;
// })
for (var i = 0; i < liblist.length; i++) {
  var taskname = liblist[i];
  // console.log(tasks[taskname]);
  tasks[taskname].name = taskname;
}

// Load jQuery (without using jQuery)

// helper function for loading, see:
//
https://stackoverflow.com/questions/950087/how-do-i-include-a-javascript-file-in-another-javas
cript-file
function loadScript(url, okFunction, errorFunction) {
  var script = document.createElement('script');
  script.type = 'text/javascript';
  script.src = url;
  // Then bind the event to the callback function.
  // There are several events for cross browser compatibility.
  script.onreadystatechange = okFunction;
  script.onload = okFunction;
  script.onerror = errorFunction;
  // Fire the loading
  document.head.appendChild(script);
}

//
https://stackoverflow.com/questions/7486309/how-to-make-script-execution-wait-until-jquery-is-
loaded
// defer -> waitfor_jquery    method -> cont
var try_counter = 0;
var try_counter_limit = 50;

function waitfor_jquery(cont) {
  //TODO replace by use of script.onerror
  // console.log( 'window.jQuery =' + window.jQuery);
  if (window.jQuery) {
    console.log('jQuery version = ' + $.fn.jquery);
    cont();
  } else {
    try_counter++;
    console.log('Waiting for jQuery... ' + try_counter);
    if (try_counter < try_counter_limit) {
      setTimeout(function () {
```

```
        waitfor_jquery(cont);
      }, 50);
    } else {
      second_try_for_jquery();
    }
  }
}

function second_try_for_jquery() {
  console.log('Try to load jQuery fallback');
  try_counter = 0;
  loadScript(jQuery_fallback, waitfor_jquery(load_libs));
}


// start loading of jQuery (if necessary)
if (window.jQuery) {
  // jQuery is already there.
  console.log('jQuery version (Wiki) = ' + $.fn.jquery);
  load_libs();
} else {
  // Start to load jQuery and wait until loaded
  loadScript(jQuery_url, waitfor_jquery(load_libs));
}
// Done with jQuery.


function errorFunc(task) {
  var fb = 'no fallback';
  if (task.fallback !== null) {
    fb = task.fallback;
  }
  console.log(task.name + ' ERROR ' + fb);
  task.state = 'error';
  // state();
  // console.log(task);
}

function OK_Func(task) {
  number_of_loaded_libs++;
  console.log(number_of_loaded_libs + ': ' + task.name);
  task.state = 'OK';
  // state();
  // console.log(task);
}

// ***************************** load CSS ***********************************
//
https://stackoverflow.com/questions/17666785/check-external-stylesheet-has-loaded-for-fallback
// https://www.phpied.com/when-is-a-stylesheet-really-loaded/
function appendStyleSheet(task, errorFunc, OK_Func, fallback) {
  var link = document.createElement("link");
  link.rel = "stylesheet";
  if (fallback) {
    link.href = task.fallback;
  } else {
    link.href = task.source;
  }
```

```
  // console.log('appendStyleSheet ' + link.href);
  link.onerror = errorFunc;
  // https://www.w3schools.com/tags/ev_onload.asp
  // link.onload = function () {
  //     console.log(link.href + ' successfully loaded.');
  //     OK_Func(task);
  // };
  link.onload = function () {
    OK_Func(task)
  };
  document.getElementsByTagName("head")[0].appendChild(link);
  console.log(link.href + ' appended to "head", but not yet loaded.');
}

function appendStyleSheetOrFallback(task, errorFunc, OK_Func) {
  // prepare for fallback
  var firstError = function () {
    if (task.fallback == null) {
      errorFunc(task);
    } else {
      // second try: fallback = true
      appendStyleSheet(task, errorFunc, OK_Func, true);
    }
  }
  // first try: fallback = false
  appendStyleSheet(task, firstError, OK_Func, false);
}

function appendScript(task, errorFunc, OK_Func, fallback) {
  if (fallback) {
    var url = task.fallback;
  } else {
    var url = task.source;
  }
  // console.log('appendScript ' + url);
  $.getScript(url)
    // .done(function (script, textStatus) {
    //     console.log(textStatus);
    // })
    // .fail(function (jqxhr, settings, exception) {
    //     console.log("Triggered ajaxError handler. " + exception);
    // });
    .done(function (script, textStatus) {
      OK_Func(task);
    })
    .fail(function (jqxhr, settings, exception) {
      errorFunc(task);
    });
}

function appendScriptOrFallback(task, errorFunc, OK_Func) {
  // prepare for fallback
  var firstError = function () {
    if (task.fallback == null) {
      errorFunc(task);
    } else {
      // second try: fallback = true
      appendScript(task, errorFunc, OK_Func, true);
```

```
    }
  }
  // first try: fallback = false
  appendScript(task, firstError, OK_Func, false);
}

function appendScriptOrStyleSheet(task, errorFunc, OK_Func) {
  if (task.css == true) {
    appendStyleSheetOrFallback(task, errorFunc, OK_Func);
  } else {
    appendScriptOrFallback(task, errorFunc, OK_Func);
  }
}

function state() {
  console.log('***');
  liblist.forEach(function (taskname) {
    var t = tasks[taskname];
    // console.log(taskname + ': ' + t.state+ ' ' + t.name);
    console.log(taskname + ': ' + t.state);
  });
  // console.log('Hammer=' + (typeof Hammer));
  // console.log('***');
}

function waitfor_num_of_libs_then_do(cont) {
  if (number_of_loaded_libs == liblist.length) {
    cont();
  } else {
    // console.log('Not enough libs: ' + number_of_loaded_libs);
    setTimeout(function () {
      waitfor_num_of_libs_then_do(cont)
    }, 50);
  }
}

var number_of_loaded_libs = 0;

function load_libs() {
  console.log(JSON.stringify(liblist));
  liblist.forEach(function (taskname) {
    tasks[taskname].state = 'wait for load';
  });
  // state();

  liblist.forEach(function (taskname) {
    tasks[taskname].state = 'pending';
    appendScriptOrStyleSheet(tasks[taskname], errorFunc, OK_Func);
  });
  waitfor_num_of_libs_then_do(prepare_pg);
}

function prepare_pg() {
  waitfor_mathquill_if_in_liblist_and_then_do(function () {
    console.log('MathQuill ready (2)');
    // if (typeof prepare_page_exists !== 'undefined') {
    if (typeof prepare_page !== 'undefined') {
      console.log('calling prepare_page...');
```

```
      prepare_page();
    } else {
      console.log('prepare_page undefined');
    }
    if (typeof init !== 'undefined') {
      init();
    } else {
      console.log('init undefined');
    }
  })
}

function waitfor_mathquill_if_in_liblist_and_then_do(mq_ready) {
  if (liblist.indexOf('mathquill') >= 0) {
    console.log('MathQuill in liblist. Waiting');
    waitfor_mathquill_and_if_ready_then_do(mq_ready);
  } else {
    mq_ready();
  }
}

// used by many *.php files
function waitfor_mathquill_and_if_ready_then_do(mq_ready2) {
  // console.log( typeof MathQuill );
  if ((typeof MathQuill) === "undefined") {
    console.log('waiting for MathQuill...');
    setTimeout(function () {
      waitfor_mathquill_and_if_ready_then_do(mq_ready2)
    }, 50);
  } else {
    console.log('MathQuill ready (1)');
    mq_ready2();
  }
}
```