# Dokumentasjon NHMs samlingsportal

## Client side

|  |  |  |
| --- | --- | --- |
| hbs-files | js-files | |
| header.hbs   * appears in all pages * contains logo, links to all pages and language-selector * differs in styles between museums * changes when screen is narrowed | swapURL.js |  |
| footer.hbs   * contains in some instances information on when collection-data was last changed * differs in styles between museums |  |  |
| index.hbs   * main-page with search-field and search-results including map | search.js  advancedAccordion.js  paginateAndRender.js  map.js  resultElementsOnOff.js | textItems.js  renderLang.js  showCorrectMuseum.js |
| map.hbs   * accessed from search-result, shows large version of map | largeMap.js |
| object.hbs   * accessed from search result, shows data, photo and map for single objects | object.js  (map.js)  newObjectPage.js  renderLangObjPage.js |
| mapObject.hbs   * accessed from object.hbs, show large version of map for single object | largeMapObject.js |
| tools.hbs   * linked to from all pages * lists collection management tools | tools.js |
| showStat.hbs   * linked to from tools.hbs * shows statistics for collections and figures | showStat.js |
| journals.hbs   * linked to from all pages * lists scanned journals for NHM * exclusive for NHM | journaler.js |
| artsObs.hbs | getArtsObsData.js |
| checkCoord.hbs | checkCoordinates.js |
| dataErrors.hbs | dataObject.js  issuesObject.js  getMetricsFromGBiF.js |
| dataFlow.hbs | dataObject.js  checkDataFlow.js  artskartAPI.js  gbifAPI.js |
| DNAbarcodes.hbs | DNAbarcodes.js |
| getDOI.hbs | dataObject.js  getDOI.js |
| help.hbs   * explains how to search, and contact information |  |
| 404.hbs |  |
| about.hbs   * not in function, covered by help.hbs |  |

samlingsportal.nhm.uio.no/museum/nhm (eller um, tmu eller nbh):

* index.hbs
  + renderLang.js is run
    - renderText(language): fills elements with text
      * header
      * organismeknapper
  + showCorrectMuseum.js is run
    - redirectIfWrong()
    - getMuseumStyleSheet()
  + search.js is run
    - getFileList(): fetches fileList from backend
    - addOrgGroups() 🡪 ta fra fileList?
    - Window.onload()
      * emptySearch()
      * emptyCollection()
        + Eller hvis vi har organismegruppe
      * oldSearch()
  + advancedAccordion.js
    - renderAdvSearchFields(termNameArray): lister opp avansert-søk-felter
    - fillSearchFields() (in renderLang.js): fyller inn placeholders in adv-search-fields

Sessionstorage:

|  |  |  |
| --- | --- | --- |
| organismGroups | [botanikk, mykologi, osv] | Set in search.js-main prior to rendering org-buttons |
| organismGroup | f.eks. ‘botanikk’ | Set when org-button is clicked, in addCollectionsToSelect, in search.js |
| Options | [algae, vascular, moser, dna-vascular] f.eks. | Set when org-button is clicked, in addCollectionsToSelect, in search.js |
|  |  |  |
|  |  |  |

### swapURL.js

#### Description:

swaps URLs when museum is changed (Oslo, Bergen, Tromsø)

#### Functions:

swapURL(path, link)

* swaps url according to chosen museum for header-links (links in top field to various pages)
* parameter: path (string, path to page)
* parameter: link (string, id of header-link
* output: changed href-attribute to header-link…
* is called by getMuseumSpecificURL()…

getMuseumSpecificURL()

* calls swapURL-function for all header-links, with correct museum-path

### showCorrectMuseum.js

#### Description:

Switches style sheets to match museum

calls getMuseumStyleSheet()

#### Functions:

swapStyleSheet(sheet)

* sets attribute to css-file
* parameter: sheet (string, name and path to css-file)
* is called in getMuseumStyleSheet()

getMuseumStyleSheet()

* swaps style sheet when another museum is chosen in url
* is called in this file (showCorrectMuseum.js)

### search.js

#### Description:

Performs searches and add functionality to buttons and selects etc.

#### Functions:

emptySearch()

* empties session’s search-result (sessionStorage), removes elements from page, resets pagination variables
* calls
  + emptyTable() in resultElementsOnOff.js
  + emptyResultElements() in resultElementsOnOff.js
* is called by emptySearchButton.eventlistener, upateFooter() when error

getCurrentMuseum()

* figures out which museum we are in
* output: string, abbreviation for museum
* is called by doSearch() and updateFooter()

capitalizeFirstLetter(string)

* returns string with capitalized first letter
* parameter: string (string)
* not in use???

addTextInOrgButtons(a)

* returns text to put in a button (innerHTML)
* parameter: a (string, an organism group, one of five)
* output: string
* is called in makeButtons()

makeButtons()

* creates button for each organism-group and adds functionality
* output: buttons on web page
* is called by addOrgGroups()

addOrgGroups()

* fetches organism groups from file at backend
* calls makeButtons()
* output: buttons….
* Is called when page is rendered

addTextInCollSelect()

addCollectionsToSelect()

download(filename, text)

* downloads searchresult to file
* parameter: filename (string, name of outputfile)
* parameter: text (string, the text that goes into the file, that is, the search result)
* output: downloaded tab-separated txt-file

forceDownload(url, filename)

* downloads photo
* parameter: url (url to photo on server)
* parameter: filename (string, name to be given to downloaded photo)
* is called by downloadPhotoButton.eventlistener

doSearch(limit)

* deletes previous search results, resets value that says if search failed, resets Boolean sorting-values for result, hides buttons, performs search
* parameter: limit (number, line number of search result where search stops)
* calls
  + resetSortedBoolean in paginateAndRender.js
  + emptyTable() in resultElementsOnOff.js
  + emptyResultElements() in resultElementOnOff.js
* is called by searchForm.eventlistener

updateFooter()

* gets date of last change of the MUSIT-dump file
* output: string (date)
* is called in oldSearch() and collection-select-eventlistener

oldSearch()

* shows previous search-result when returning to main-page, if it exists in this session
* is called in search.js, that is, every time main page is rendered
* calls renderText(language), updateFooter(), load()

checkSeveralBoxes(subMusitData)

* checks or unchecks several check-boxes and updates searchresult in sessionStorage with this information
* parameter: data (JSON; part of search result that is rendered on page)
* is called by dropdown-menu with checkbox-options in searchresult-table

downloadbuttons-eventlisteners

searchForm-eventlistener

collection-select-eventlistener

emptySearchButton.eventlistener

* calls emtpySearch()

large-map-button.onclick

export-map-button.eventlistener

show-selected-records-in-map-button.eventlistener

* calls drawMap(array)

dropdown for checking or un-checkinig several checkboxes.onchange

* calls checkSeveralBoxes(JSONdata)

### advancedAccordion.js

#### Description:

Performs advanced search and spesicif-object-number-search, has functionality for accordions for these searches.

#### Functions:

doAdvancedSearch()

doObjListSearch()

### textItems.js

#### Description:

Contains a part of the generic urlPath that is common for all museums, and an array with strings to be rendered in html-elements, in Norwegian and English

Both elements are used in **pagingateAndRender.js**

### paginateAndRender.js

#### Description:

Renders result table and contains functionality related to this, e.g. sorting of table and splitting of result into pages.

#### Functions:

itemType(catalogNumber)

* returns itemType for a record, e.g. tissue, egg, sperm, skin etc.
* parameter: catalogNumber (string, catalogNumber)
* output: itemType (string, e.g. ‘tissue’, ‘egg’)
* is called in resultTable(..) (in the future, when stitching of files is done)

hide\_column(col\_no)

* hides column with genetic data in result-table for bergen and tromsø (museums without genetic database (corema))
* called by resultTable(…)

resultTable(subMusitData, musitData)

* renders result table
* parameter: subMusitData (JSON; part of search result that is rendered on page)
* parameter: musitData (JSON; searchResult, all of it)
* calls
  + fillResultHeaders(…) in resultElementOnOff.js
  + investigateChecked(i), to check if boxes should appear as checked
  + hide\_column(number) to hide corema-columns when necessary
  + showResultElement() in resultElementOnOff.js
  + drawMap(data) from map.js
  + checkSeveralBoxes(data) to add function to dropdown for checkboxes
* is called by
  + drawList()
  + addSortingText(…)

##### pagination-part

getNumberOfPages(numberPerPage)

* returns numberOfPages for rendering results
* output: numberOfPages (number, numberOfPages for rendering results)
* is called by
  + makeList()
  + hitsPerPage-select.eventlistener
  + resultTable(..)

nextPage()

* increases counter for currentPage for rendering results, if necessary puts text in html-element lastPageAlert
* calls load()
* is called in index.hbs when nextPage-button is created

previousPage()

* decreases counter for currentPage for for rendering results
* calls load()
* is called in index.hbs when previousPage-button is created

firstPage()

* sets currentPage for rendering results
* calls load()
* is called in index.hbs when firstPage-button is created

lastPage()

* sets currentPage for rendering results
* calls load()
* is called in index.hbs when lastPage-button is created

loadList()

* sets sessionStorage’s pageList (part of result that is to be rendered on page) and calls function(s) that render resultTable
* calls
  + check()
  + calls resultTable(pageList, list)
* is called by
  + hitsPerPage eventlistener
  + load()

check()

* disables page-buttons if necesary
* is called by loadList()

load()

* empties search-result, fetches search result from sessionStorage, sets numberOfPages for rendering of results, calls function that call resultTable and sets sessionStorage’s pagelist (what to rendered on page)
* calls
  + getNumberOfPages(..)
  + loadList()
* is called by nextPage(), previousPage(), firstPage(), lastPage()

hitsPerPage.eventlistener()

* calls getNumberOfPages
* calls loadList()

investigaetChecked(i)

* checks if checkbox is checked for each record when rendering results
* parameter: i (number; index for searchresult-array)
* output: boolean (true if box should be checked)
* is called by resultTable()

registerChecked(i)

* registers that a (single) checkbox is checked or unchecked
* parameter: i (number, index for searchresult-array)
* output: boolean (true if box should be checked)
* is called by resultTable() for each record

### renderLang.js

#### Description:

Renders text in relevant language (Norwegian or English) when page is rendered, and when language is changed

#### Functions:

italicSpeciesname(string)

* formats species name to be italic and potential author to be non-italic
* in: string, species name with (potential) author
* out: array with two strings; scientific name and author
* is called in makeTableHeader() and showData() in object.js

renderText(lang)

* renders text and images in html-elements
* parameter: lang (string, «Norwegian» or «English»)
* output: text or images in relevant HTML-elements
* is called in renderLang.js

language.eventlistener

* calls
  + fillResultHeaders(…) in resultElementsOnOff.js
  + checkSeveralBoxes()

### resultElementsOnOff.js

#### Description:

Toggles buttons and other elements related to rendering of search–result on and off (displays, or removes), creates an array propsSorted[] with Boolean values that keeps track of on which field the result table is sorted on.

#### Functions:

resetSortedBoolean()

* resets the Boolean sorting values for the resultTable
* is called by doSearch(limit) in search.js

emptyTable()

* empties table and result-related text-elements
* is called by
  + doSearch(..) in search.js
  + emptySearch(..) in search.js

emptyResultElements()

* hide buttons rendered with search result
* is called by
  + doSearch(..) in search.js
  + emptySearch() in search.js

showResultElements()

* display buttons rendered with search result
* is called by
  + resultTable() in paginateAndRender.js

sort\_by (prop, reverse, primer)

* aids in sorting search-result when header-button in tabel is clicked
* parameter: prop (number; to find the correct property to sort on, from the propsSorted-array)
* parameter: reverse (boolean, decides which way data are being sorted)
* parameter: primer (function; e.g. parseInt that will turn a propety into an integer?)
* output: sorting order of two elements (?)
* called by addSortingText(…)

addSortingText(id, prop, musitData)

* adds sorting-function to buttons that are table-headings in result-table; and contains the sorting-function; search-result is sorted and result re-rendered
* parameter: id (string; id of button)
* parameter: prop (number; to find the correct property to sort on, from the propsSorted-array)
* parameter: musitData (JSON; searchResult, all of it)
* calls
  + sort\_by(prop,reverse,primer)
  + resultTable(subMusitData, musitData)
* is called by fillResultHeaders(…)

getArrows(prop)

* renders image of arrow-up or arrow-down in table –header when result-table is being sorted
* parameter: prop (string; property of a record in the search result)
* output: image(s)
* is called by fillResultHeaders(…)

fillResultHeaders(cell1,cell2,cell3,cell4,cell5,cell6,cell7,cell8,cell9,cell10,cell11,musitData)

* puts content in headerbuttons in result-table
* calls
  + getArrows(..) for table-header-buttons
  + addSortingText(..) for tabel-header-buttons
* is called in language-button.addEventListener in renderLang.js

and in resultTable() in paginateAndRender.js

### map.js

#### Description:

Contains functions for drawing maps that are embedded in search-page or object-page

#### Functions:

drawMap(parsedData)

* draws map with clickable red markes for all records in search result with coordinates
* parameter: parsedData (JSON, search result)
* output: map with markers
* is called by resultTable(..) in paginateAndRender.js

drawMapObject(object)

* draws map with red marker for single record in object.hbs
* parameter: object (JSON object)
* output: map with marker
* is called by object.js

### largeMap.js

#### Description:

Calls function that draw map for all or chosen objects from search result.

#### Functions:

loadString()

* picks out checked records from search result to array
* is called in this file (largeMap.js)

### largeMapObject.js

#### Description:

Calls function that draw map for a single object.

#### Functions:

loadStringObjectMapPage()

* fetches JSON-version of search-result from session Storage
* is called in this file (largeMapObject.js)

### object.js

#### Description:

Renders text on object.hbs

#### Functions:

getCurrentMuseum()

* figures out which museum we are in
* output: string, abbreviation for museum
* is called by document.getElementById('large-map-object-button').onclick

getOrganismGroup()

* figures out which organism group we are in
* out: string
* is called below function to fill variable orgGroup

loadStringObject()

* reads last search-result from sessionStorage and puts it in an array
* output: an array with JSON-objects that represents museumobjects in the last search result. Empty array if there is no search result in sessionStorage
* is called in this file (object.js)

async setOrgGroup()

* fetches organism-group from backend and sets it in sessionStorage (botanikk, mykologi, zoologi, osv)
* is called in main()

async getspecimenData (allObject)

* gets relevant object from search result
* parameter: an array with JSON-objects that represents museumobjects in the last search result. Empty array if there is no search result in sessionStorage
* is called in main()

async whichFileAndDb (museum,collection)

* fetches type of file (stitched if collection is in both corema and musit, or only in corema, not-stitched if not) and db (musit or corema or access etc) from backend and put them in sessionStorage
* parameter: museum, string
* parameter: collection, string
* is called in main()
* is used a.o. when rendering data for items/preparations

makeNavButtons(allObject, specimenObject)

country(obj)

* facilitates correctly formatted locality information for a museum object
* parameter: obj (a JSON-object that rerpresens the museum object that is to be shown)
* output: string representing the collection-country followed by comma, or empty string

stateProvince(obj)

* facilitates correctly formatted locality information for a museum object
* parameter: obj (a JSON-object that represents the museum object that is to be shown)
* output: string representing the collection-county followed by comma, or empty string

county(obj)

* facilitates correctly formatted locality information for a museum object
* parameter: obj (a JSON-object that represents the museum object that is to be shown)
* output: string representing the collection-municipality followed by comma, or empty string

locality(obj)

* facilitates correctly formatted locality information for a museum object
* parameter: obj (a JSON-object that represents the museum object that is to be shown)
* output: string representing the collection-locality followed by comma, or empty string

coordinates(obj)

* formats coordinates to nice readable format
* parameter: obj (JSON-object that represents the museum object that is to be shown)
* output: string with the object’s coordinates

taxonomyString()

* fills variable taxonomy with content
* output: string with taxonomic hierarchy

makeTableHeader(table)

* makes first line in table for object-information, fills the cells (scientific name), and fills musit-regno-box
* parameter: table; html-element
* is called in makeUTADTable() and makePalTable()

makeBioTable()

* builds table for object-data for botany and zoology
* is called in showData()

makeUTADTable(object)

* builds table for object-data for UTAD's collections
* inserts row for each property the object has, and that we decided to show
* add cells, with id, class and style
* fills cells with data (not header-cells)
* in: specimenObject; collection-object
* calls makeTableHeader(table)
* is called in showData()

makePalTable(object)

* builds table for object-data for pal-collections
* inserts row for each property the object has, and that we decided to show
* add cells, with id, class and style
* fills cells with data (not header-cells)
* in: specimenObject; collection-object
* calls makeTableHeader(table)
* is called in showData()

makeGeoTable(object)

* builds table for object-data for geo collections
* inserts row for each property the object has, and that we decided to show
* add cells, with id, class and style
* fills cells with data (not header-cells)
* in: specimenObject; collection-object
* calls makeTableHeader(table)
* is called in showData()

showData()

* calls function that build table and fill headers, and fills data in table for zoology and botany
* calls makeUTADTable(obj) or makeGeoTable(obj) or makePalTable(obj) or makeBioTable(obj)
* is called below

addRow()

* adds row to table that lists object’s data on the fly – number of rows necessary varies between objects
* will be called in the future when stitched data is in place

changeImage(index, direction, smallImageList, imageList)

* shows next image when nextImage-button is clicked, if there are more images for object
* parameter: index (number, points to image)
* parameter: direction (string, ‘f’ or ‘r’ for ‘forward’ and ‘reverse’; dependent on which button is clicked; next or previous image)
* parameter: smallImageList (array with strings; names of small images)
* parameter: imageList (array with strings; names of large images)
* is called in next-photo-button.onclick and previous.photo.onclick

reducePhoto(photo)

* returns small version of photo
* parameter: photo (string, name of large photo)
* output: photo (string, name of small photo)
* is called by this file (object.js)

imageExists(image\_url)

* http-request to check if an image exists
* parameter: image\_url (string; name of photo)
* is called only by a console.log in this file

### tools.js

#### Description:

Renders content in tools.hbs

### showStat.js

#### Description:

Renders content in journaler.js

#### Functions:

Number.prototype.format(n,x,s,c)

* parameter: n (integer, length of decimal)
* parameter: x (integer, length of whole part)
* parameter: s (mixed, sections delimiter)
* parameter: c (mixed, decimal delimiter)
* is called in populateTable(..)

populateTable(data)

* puts data into the statistics-table with key value from each collection
* parameter: data (JSON object with data from the collections)

sortData(arrayData, by, ascOrDes)

sorts array with data

* parameter: arrayData (array [{label:value, data:value}, {label:value, data:value}]
* parameter: by (string, label or data (for what will be sorted by))
* parameter: ascOrDes (string, ‘asc’, for ascending sort, smalletst to largest) or ‘des’ for descending sort)
* output: sorted Array
* is called by
  + tilvekstData(..)
  + accumulativeCollectionsSize(…)
  + top20Land(..)

tilvekstData(data, currentCollection)

* returns labels and data for yearly growth to the different graphs
* parameter: data (JSON file with collection data)
* parameter: currentCollection (string, which collection to be shown (from dropdown-menu))
* output: an array with labels and data for yearly growth
* calls sortData(..)
* is called by
  + makeGraphs(..)
  + updateGraph()

accumulativeCollectionSize(data, currentCollection)

* returns label and data for accumulative numbers
* parameter: data (JSON file with collection data)
* parameter: currentCollection (string, which collection to be shown (from dropdown-menu))
* output: an array with labels and data for accumulative data
* calls sortData(..)
* is called by
  + makeGraphs(..)
  + updateGraph()

top20Land (data, currentCollection)

* returns label and data for top 20 countires except Norway
* parameter: data (JSON file with collection data)
* parameter: currentCollection (string, which collection to be shown (from dropdown-menu))
* output: an array with labels and data for top 20 countires except Norway
* calls sortData()
* is called by
  + makeGraphs(..)
  + updateGraph()

harKoordinater (data, currentCollection)

* returns label and data for coordinate pie?
* parameter: data (JSON file with collection data)
* parameter: currentCollection (string, which collection to be shown (from dropdown-menu))
* output: an array with labels and data for coordinate pie?
* is called by
  + makeGraphs(..)
  + updateGraph()

fraNorge(data, currentCollection)

* returns label and data for Norwegian data
* parameter: data (JSON file with collection data)
* parameter: currentCollection (string, which collection to be shown (from dropdown-menu))
* output: an array with labels and data for Norwegian data
* is called by
  + makeGraphs(..)
  + updateGraph()

getData()

* sends request to backend to fetch data
* is called by main()

makeGraphs(data)

* makes the graphs
* parameter: data (JSON file)
* calls
  + tilvekstData(..)
  + accumulativeCollectionSize(..)
  + harKoordinater(..)
  + fraNorge(..)
  + top20Land(..)
* is called by main()

updateGraph()

* make new graphs when new collection is chosen
* calls
  + tilvekstData(..)
  + accumulativeCollectionSize(..)
  + harKoordinater(..)
  + fraNorge(..)
  + top20Land(..)
* is called by collection-select.eventlistener

main()

* async function; renders the page the first time
* calls
  + makeGraphs(..)
  + populateTable(..)
* is called in this file (showStat.js)

### journaler.js

#### Description:

Renders content in journaler.hbs

#### Functions:

addHeaders(table, keys)

* creates the headers in the table
* parameter: table (html-table, to show on the page)
* parameter: keys (array?, source of header titles)
* is called in journalResultTable(..)

journalResultTable(children)

* creates table for the journals and fills it
* parameter: children (array, containing content to table, i.e. data on journals)
* calls addHeaders(..)
* is called in dorournalSearch(..)

doJournalSearch(limit)

* performs search and fetches data
* parameter: limit (integer; maximum number of records to display)
* calls journalResultTable(..)
* is called in journalSearchForm.eventlistener

updateFooter()

* sends request to server for date of last change of the journal-datafile
* is called in this file (journaler.js)

## Server side

Rammeverktøy er NodeJS og Express

<https://nodejs.org/en/>

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

**Express**

<https://www.npmjs.com/package/express>

Fast, unopinionated, minimalist web framework for node.

## Dependencies

**Hbs**

<https://www.npmjs.com/package/hbs>

Express.js view engine for handlebars.js

**Path**

<https://www.npmjs.com/package/path>

This is an exact copy of the NodeJS ’path’ module published to the NPM registry.

### DEV:

**nodemon**

<https://www.npmjs.com/package/nodemon>

nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.

## Web server

Express brukes som webserver for portalen. Dette er en liten og lett web server som passer utmerket til vårt bruk.

Web-serveren har følgende endepunkter:

|  |  |  |
| --- | --- | --- |
| Endepunkt | Returnerer | input |
| App.get('', (req, res) …) | Index.html | ingen |
| app.get('/search', (req, res) …) | **Unparsed.results**  Søkeresultater, som en tabseparert variable | http://localhost:3000/search/  ?search=' + searchTerm +'&samling=' + chosenCollection |
| app.get('/download', (req, res)…) | **Unparsed.results**  Søkeresultater, som en tabseparert variable | http://localhost:3000/search/  ?search=' + searchTerm +'&samling=' + chosenCollection |
| app.get('/footer-date', (req, res)...) | **date: date**  sist endret dato for datafila | http://localhost:3000/footer-date/?&samling=' + chosenCollection |
| app.get('/object', (req, res)…) | **myObject: req.query.id**  musit nummeret | http://localhost:3000/object/?id=${i} {data[i].catalogNumber} |
| app.get('/about', (req, res)…) | About.html |  |
| app.get('/help', (req, res)…) | Help.html |  |
| app.get('\*', (req, res)…) | 404.html  File not found | Plukker opp alle ugyldige nettadresser |

## Funksjoner

## Fil: footerDate.js

getFileUpdatedDate = (samling, callback) => {…}

**Input**:

text string fra nedvalgsliste, {karplanter; Sopp, Moser; Lav; Alger; Insekter}

**Retur**:

Leser filedatoen fra datafila som blir valgt og returnerer dato slik at den vises som last updated

## Fil: fileread.js

setCollection = (samling) => {…}

**Input**:

et valg fra droppdown feltet for samling

return musitFile

Oversetter samling til filnavn.

search = (samling, searchTerm, callback) => {… }

**input**:

samling (som oversettes til musti fil i funksjonen setCollection)

seachTrem =som er det brukeren søker på, kan være 2 ord skilt av space

**retur**:

results, en tabseparert variabel med linje for linje med resultatene