

Toggle menu  
Blue Gold Program Wiki

## Navigation

- [Main page](#)
- [Recent changes](#)
- [Random page](#)
- [Help about MediaWiki](#)

## Tools

- [What links here](#)
- [Related changes](#)
- [Special pages](#)
- [Permanent link](#)
- [Page information](#)

## Personal tools

- [Log in](#)

## personal-extra

Toggle search

Search

Random page

## Views

- [View](#)
- [View source](#)
- [History](#)
- [PDF Export](#)

## Actions

# Module:Hatnote list

From Blue Gold Program Wiki

The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

*Documentation for this module may be created at [Module:Hatnote list/doc](#)*

```

-----
---
--
--                               Module:Hatnote list
--
--
--
-- This module produces and formats lists for use in hatnotes. In particular,
--
-- it implements the for-see list, i.e. lists of "For X, see Y" statements,
--
-- as used in {{about}}, {{redirect}}, and their variants. Also introduced
--
-- are andList & orList helpers for formatting lists with those conjunctions.
--
-----
---

local mArguments --initialize lazily
local mHatnote = require('Module:Hatnote')
local libraryUtil = require('libraryUtil')
local checkType = libraryUtil.checkType
local p = {}

-----
---
-- List stringification helper functions
--
-- These functions are used for stringifying lists, usually page lists inside
-- the "Y" portion of "For X, see Y" for-see items.
-----
---

--default options table used across the list stringification functions
local stringifyListDefaultOptions = {
    conjunction = "and",
    separator = ",",
    altSeparator = ";",
    space = " ",
    formatted = false
}

-- Stringifies a list generically; probably shouldn't be used directly
function stringifyList(list, options)
    -- Type-checks, defaults, and a shortcut
    checkType("stringifyList", 1, list, "table")
    if #list == 0 then return nil end
    checkType("stringifyList", 2, options, "table", true)
    options = options or {}
    for k, v in pairs(stringifyListDefaultOptions) do
        if options[k] == nil then options[k] = v end
    end
end

```

```

    local s = options.space
    -- Format the list if requested
    if options.formatted then list = mHatnote.formatPages(unpack(list))
end
    -- Set the separator; if any item contains it, use the alternate
separator
    local separator = options.separator
    --searches display text only
    local function searchDisp(t, f)
        return string.find(string.sub(t, (string.find(t, '|') or 0) +
1), f)
    end
    for k, v in pairs(list) do
        if searchDisp(v, separator) then
            separator = options.altSeparator
            break
        end
    end
    -- Set the conjunction, apply Oxford comma, and force a comma if #1
has "$"
    local conjunction = s .. options.conjunction .. s
    if #list == 2 and searchDisp(list[1], "$") or #list > 2 then
        conjunction = separator .. conjunction
    end
    -- Return the formatted string
    return mw.text.listToText(list, separator .. s, conjunction)
end

--DRY function
function conjList (conj, list, fmt)
    return stringifyList(list, {conjunction = conj, formatted = fmt})
end

-- Stringifies lists with "and" or "or"
function p.andList (...) return conjList("and", ...) end
function p.orList (...) return conjList("or", ...) end

-----
---
-- For see
--
-- Makes a "For X, see [[Y]]." list from raw parameters. Intended for the
-- {{about}} and {{redirect}} templates and their variants.
-----
---

--default options table used across the forSee family of functions
local forSeeDefaultOptions = {
    andKeyword = 'and',
    title = mw.title.getCurrentTitle().text,
    otherText = 'other uses',

```

```

        forSeeForm = 'For %s, see %s.',
    }

--Collapses duplicate punctuation
function punctuationCollapse (text)
    local replacements = {
        ["%.%.$"] = ".",
        ["%?%.$"] = "?",
        ["%!%.$"] = "!",
        ["%.%]%"%. "$"] = ".]"]",
        ["%?%]%"%. "$"] = "?]"]",
        ["%!%]%"%. "$"] = "!]"]"
    }
    for k, v in pairs(replacements) do text = string.gsub(text, k, v) end
    return text
end

-- Structures arguments into a table for stringification, & options
function p.forSeeArgsToTable (args, from, options)
    -- Type-checks and defaults
    checkType("forSeeArgsToTable", 1, args, 'table')
    checkType("forSeeArgsToTable", 2, from, 'number', true)
    from = from or 1
    checkType("forSeeArgsToTable", 3, options, 'table', true)
    options = options or {}
    for k, v in pairs(forSeeDefaultOptions) do
        if options[k] == nil then options[k] = v end
    end
    -- maxArg's gotten manually because getArgs() and table.maxn aren't
friends
    local maxArg = 0
    for k, v in pairs(args) do
        if type(k) == 'number' and k > maxArg then maxArg = k end
    end
    -- Structure the data out from the parameter list:
    -- * forTable is the wrapper table, with forRow rows
    -- * Rows are tables of a "use" string & a "pages" table of pagename
strings
    -- * Blanks are left empty for defaulting elsewhere, but can
terminate list
    local forTable = {}
    local i = from
    local terminated = false
    -- If there is extra text, and no arguments are given, give nil value
    -- to not produce default of "For other uses, see foo
(disambiguation)"
    if options.extratext and i > maxArg then return nil end
    -- Loop to generate rows
    repeat
        -- New empty row
        local forRow = {}

```

```

-- On blank use, assume list's ended & break at end of this
loop
  forRow.use = args[i]
  if not args[i] then terminated = true end
  -- New empty list of pages
  forRow.pages = {}
  -- Insert first pages item if present
  table.insert(forRow.pages, args[i + 1])
  -- If the param after next is "and", do inner loop to collect
params
  -- until the "and"'s stop. Blanks are ignored: "1|and||and|3"
  → {1, 3}
  while args[i + 2] == options.andKeyword do
    if args[i + 3] then
      table.insert(forRow.pages, args[i + 3])
    end
    -- Increment to next "and"
    i = i + 2
  end
  -- Increment to next use
  i = i + 2
  -- Append the row
  table.insert(forTable, forRow)
until terminated or i > maxArg
return forTable
end

-- Stringifies a table as formatted by forSeeArgsToTable
function p.forSeeTableToString (forSeeTable, options)
  -- Type-checks and defaults
  checkType("forSeeTableToString", 1, forSeeTable, "table", true)
  checkType("forSeeTableToString", 2, options, "table", true)
  options = options or {}
  for k, v in pairs(forSeeDefaultOptions) do
    if options[k] == nil then options[k] = v end
  end
  -- Stringify each for-see item into a list
  local strList = {}
  if forSeeTable then
    for k, v in pairs(forSeeTable) do
      local useStr = v.use or options.otherText
      local pagesStr = p.andList(v.pages, true) or
mHatnote._formatLink{link = mHatnote.disambiguate(options.title)}
      local forSeeStr = string.format(options.forSeeForm,
useStr, pagesStr)
      forSeeStr = punctuationCollapse(forSeeStr)
      table.insert(strList, forSeeStr)
    end
  end
  if options.extratext then table.insert(strList,
punctuationCollapse(options.extratext..'.')) end
end

```

```

-- Return the concatenated list
return table.concat(strList, ' ')
end

-- Produces a "For X, see [[Y]]" string from arguments. Expects index gaps
-- but not blank/whitespace values. Ignores named args and args < "from".
function p._forSee (args, from, options)
    local forSeeTable = p.forSeeArgsToTable(args, from, options)
    return p.forSeeTableToString(forSeeTable, options)
end

-- As _forSee, but uses the frame.
function p.forSee (frame, from, options)
    mArguments = require('Module:Arguments')
    return p._forSee(mArguments.getArgs(frame), from, options)
end

return p

```

Retrieved from "[https://www.bluegoldwiki.com/index.php?title=Module:Hatnote\\_list&oldid=3590](https://www.bluegoldwiki.com/index.php?title=Module:Hatnote_list&oldid=3590)"

## Namespaces

- [Module](#)
- [Discussion](#)

## Variants

This page was last edited on 15 November 2020, at 11:23.

## Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.

- [Privacy policy](#)
- [About Blue Gold Program Wiki](#)
- [Disclaimers](#)

Developed and maintained by Big Blue Communications for Blue Gold Program



[Blue Gold Program Wiki](#)