```
{{#Text}}<div class="prettify-flashcard">{{/Text}} <!this makes the box around
text-->

<br>
<script>
// ##############  HINT REVEAL SHORTCUTS  ##############
// All shortcuts will also open with "H" if using the Hint Hotkeys add-on
var ButtonShortcuts = {
    "Lecture Notes" : "Alt+1",
    "Missed Questions" : "Alt+2",
    "Pathoma" : "Alt+3",
    "Boards and Beyond" : "Alt+4",
    "First Aid" : "Alt+5",
    "Sketchy" : "Alt+6",
    "Sketchy 2": "",
    "Sketchy Extra": "",
    "Picmonic": "",
    "Pixorize" : "Alt+7",
    "Physeo" : "Alt+8",
    "Bootcamp" : "Alt+F2",
    "OME" : "Alt+F1",
    "Additional Resources" : "Alt+9",
}
var ToggleAllButtonsShortcut = "'"
var ToggleNextButtonShortcut = "H";
// ##############  SHOW HINTS AUTOMATICALLY  ##############
var ButtonAutoReveal = {
    "Lecture Notes" : false,
    "Missed Questions" : false,
    "Pathoma" : false,
    "Boards and Beyond" : false,
    "First Aid" : false,
    "Sketchy" : false,
    "Sketchy 2": false,
    "Sketchy Extra": false,
    "Picmonic": false,
    "Pixorize" : false,
    "Physeo" : false,
    "Bootcamp" : false,
    "OME" : false, // not currently a button
    "Additional Resources" : false,
}

var ScrollToButton = true;

// ##############  TAG SHORTCUT  ##############
var toggleTagsShortcut = "C";

// ENTER THE TAG TERM WHICH, WHEN PRESENT, WILL TRIGGER A RED BACKGROUND
var tagID = "XXXYYYZZZ"

// WHETHER THE WHOLE TAG OR ONLY THE LAST PART SHOULD BE SHOWN

var numTagLevelsToShow = 0;

// ##############  CLOZE ONE BY ONE  ##############
var revealNextShortcut = "N"
var revealNextWordShortcut = "Shift+N"
var toggleAllShortcut = ","

// Changes how "Reveal Next" and clicking behaves. Either "cloze" or "word".
// "word" reveals word by word.
var revealNextClozeMode = "cloze"
```

```
// What cloze is hidden with
var clozeHider = (elem) => "[...]"
/*
You can replace the above line with below examples. '▉' or '_' works well for
hiding clozes.

// Fixed length:
var clozeHider = (elem) => "▉▉▉"
// Replace each character with "▉":
var clozeHider = (elem) => "▉".repeat(elem.textContent.length)
// Show whitespaces:
var clozeHider = (elem) => "[" + elem.textContent.split(" ").map((t) =>
"▉".repeat(t.length)).join(" ") + "]"
// Color-filled box (doesn't hide images):
var clozeHider = (elem) => `<span style="background-color: red; color:
transparent;">${elem.innerHTML}</span>`
*/

</script>


<div class="clozefield" id="text">{{cloze:Text}}</div>

<!-- ###############  EDIT CLOZE DURING REVIEW  ###############
                  -change below (not above) to  "edit:cloze:Text" for editable
field,
                        but be sure to have the correct add-on installed-->

<div class="editcloze" id="text">{{edit:cloze:Text}}</div>




<!-- ###############  TEXT-TO-SPEECH ###############
replace the arrows/dashes from the statement below with double curly brackets-->

<!--tts en_US voices=Apple_Samantha,Microsoft_Zira speed=1.4:cloze-only:Text-->

<br>
<div class="prettify-divider prettify-divider--answer"></div>


<!-- BUTTON FIELDS -->
<!-- ClOZE ONE BY ONE BUTTONS -->

<div id="one-by-one" style="display: none;">{{One by one}}</div>

<div id="1by1-btns" style="display: none;">
  <button id="button-reveal-next" class="button-general"
onclick="revealNextCloze()">Reveal Next</button>
  <button id="button-toggle-all" class="button-general"
onclick="toggleAllCloze()">Toggle All</button>
</div>

<script>
  (() => {
    let clozeOneByOneEnabled = true;
      clozeOneByOneEnabled = document.getElementById("one-by-one").textContent !
== "";

    if (clozeOneByOneEnabled) {
      document.getElementById("1by1-btns").style.display = "block";
    }
    })()
```

```
  </script>

<!-- Remove 'style="display: none"' in the line saying '<span style="display:
none">' to show the OME field -->
{{#Lecture Notes}}
<span id = "hint-ln" class="hintBtn" data-name="Lecture Notes">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-ln')"></a>
  <button id="button-ln" class="button-general" onclick="toggleHintBtn('hint-
ln')">
    <img src="_Anking_v3.png"> Lecture Notes
  </button>
  <div dir="auto" id="notes" class="hints" style="display: none;">{{edit:Lecture
Notes}}</div>
</span>
{{/Lecture Notes}}

<!-- BUTTON FIELDS -->
{{#Missed Questions}}
<span id = "hint-mq" class="hintBtn" data-name="Missed Questions">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-mq')"></a>
  <button id="button-mq" class="button-general" onclick="toggleHintBtn('hint-
mq')">
    <img src="_Anking_v3.png"> Missed Questions
  </button>
  <div dir="auto" id="missed" class="hints" style="display: none;">{{edit:Missed
Questions}}</div>
</span>
{{/Missed Questions}}

{{#Pathoma}}
<span id = "hint-pat" class="hintBtn" data-name="Pathoma">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-pat')"></a>
  <button id="button-pat" class="button-general" onclick="toggleHintBtn('hint-
pat')">
    <img src="_pathoma.icon.png">
  </button>
  <div dir="auto" id="pathoma" class="hints" style="display:
none;">{{edit:Pathoma}}</div>
</span>
{{/Pathoma}}

{{#Boards and Beyond}}
<span id = "hint-bb" class="hintBtn" data-name="Boards and Beyond">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-bb')"></a>
  <button id="button-bb" class="button-general" onclick="toggleHintBtn('hint-
bb')">
    <img src="_b&b.icon.png"> Boards and Beyond
  </button>
  <div dir="auto" id="bnb" class="hints" style="display: none;">{{edit:Boards
and Beyond}}</div>
</span>
{{/Boards and Beyond}}

{{#Extra}}<p></p>
<div id="extra">{{edit:Extra}}</div>
<br>{{/Extra}}


<!-- Extra field -->
{{#First Aid}}
<span id = "hint-fa" class="hintBtn" data-name="First Aid">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-fa')"></a>
  <button id="button-fa" class="button-general" onclick="toggleHintBtn('hint-
fa')">
```

```
    <img src="_first-aid.icon_v2.1.png"> First Aid
  </button>
  <div dir="auto" id="firstaid" class="hints" style="display:
none;">{{edit:First Aid}}</div>
</span>
{{/First Aid}}

{{#Sketchy}}
<span id = "hint-sketchy" class="hintBtn" data-name="Sketchy">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-sketchy')"></a>
  <button id="button-sketchy" class="button-general"
onclick="toggleHintBtn('hint-sketchy')">
    <img src="_sketchy.icon.png"> Sketchy
  </button>
  <div dir="auto" id="sketchy" class="hints" style="display:
none;">{{edit:Sketchy}}</div>
</span>
{{/Sketchy}}

{{#Sketchy 2}}
<span id = "hint-sketchy2" class="hintBtn" data-name="Sketchy 2">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-sketchy2')"></a>
  <button id="button-sketchy2" class="button-general"
onclick="toggleHintBtn('hint-sketchy2')">
    <img src="_sketchy.icon.png"> Sketchy 2
  </button>
  <div dir="auto" id="sketchy2" class="hints" style="display:
none;">{{edit:Sketchy 2}}</div>
</span>
{{/Sketchy 2}}

{{#Sketchy Extra}}
<span id = "hint-sketchyextra" class="hintBtn" data-name="Sketchy Extra">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-sketchyextra')"></a>
  <button id="button-sketchyextra" class="button-general"
onclick="toggleHintBtn('hint-sketchyextra')">
    <img src="_sketchy.icon.png"> Sketchy Extra
  </button>
  <div dir="auto" id="sketchyextra" class="hints" style="display:
none;">{{edit:Sketchy Extra}}</div>
</span>
{{/Sketchy Extra}}

{{#Picmonic}}
<span id = "hint-picmonic" class="hintBtn" data-name="Picmonic">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-picmonic')"></a>
  <button id="button-picmonic" class="button-general"
onclick="toggleHintBtn('hint-picmonic')">
    <img src="_picmonic.icon.png"> Picmonic
  </button>
  <div dir="auto" id="picmonic" class="hints" style="display:
none;">{{edit:Picmonic}}</div>
</span>
{{/Picmonic}}

{{#Pixorize}}
<span id = "hint-pixorize" class="hintBtn" data-name="Pixorize">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-pixorize')"></a>
  <button id="button-pixorize" class="button-general"
onclick="toggleHintBtn('hint-pixorize')">
    <img src="_pixorize.png"> Pixorize
  </button>
  <div dir="auto" id="pixorize" class="hints" style="display:
none;">{{edit:Pixorize}}</div>
```

```
</span>
{{/Pixorize}}

{{#Physeo}}
<span id = "hint-physeo" class="hintBtn" data-name="Physeo">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-physeo')"></a>
  <button id="button-physeo" class="button-general"
onclick="toggleHintBtn('hint-physeo')">
    <img src="_physeologo.icon.png">
  </button>
  <div dir="auto" id="physeo" class="hints" style="display:
none;">{{edit:Physeo}}</div>
</span>
{{/Physeo}}

{{#Bootcamp}}
<span id = "hint-bootcamp" class="hintBtn" data-name="Bootcamp">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-bootcamp')"></a>
  <button id="button-bootcamp" class="button-general"
onclick="toggleHintBtn('hint-bootcamp')">
    <img src="_bootcamp.png"> Bootcamp
  </button>
  <div dir="auto" id="bootcamp" class="hints" style="display:
none;">{{edit:Bootcamp}}</div>
</span>
{{/Bootcamp}}

{{#OME}}
<div class="banner-ome"><a href="https://hubs.ly/Q01w9Y7d0">
  <img src="_OME_banner_memorize_understand.png"></a>
</div>

<span style="display: none">
  <span id="hint-ome" class="hintBtn" data-name="OME">
    <a onclick="toggleHintBtn('hint-ome')"></a>
    <button id="button-ome" class="button-general" onclick="toggleHintBtn('hint-
ome')">
      <img src="_omelogo.icon.png"> OnlineMedEd
    </button>
    <div id="ome" class="hints" style="display: none;">{{edit:OME}}</div>
  </span>
</span>
{{/OME}}

{{#Additional Resources}}
<span id = "hint-ar" class="hintBtn" data-name="Additional Resources">
  <a href="#" class="hint" onclick="toggleHintBtn('hint-ar')"></a>
  <button id="button-ar" class="button-general" onclick="toggleHintBtn('hint-
ar')">
    <img src="_Anking_v3.png"> Additional Resources
  </button>
  <div dir="auto" id="additional" class="hints" style="display:
none;">{{edit:Additional Resources}}</div>
</span>
{{/Additional Resources}}


<!-- ANKING HYPERLINK IMAGE -->
<a href="https://www.ankingmed.com"><img src="_AnKingRound.png" alt="The AnKing"
id="pic"></a>

<!-- NOT-PERSISTING EVENT LISTENER -->
<script>
  if (window.ankingEventListeners) {
```

```
    for (const listener of ankingEventListeners) {
      const type = listener[0]
      const handler = listener[1]
      document.removeEventListener(type, handler)
    }
  }
  window.ankingEventListeners = []

  window.ankingAddEventListener = function(type, handler) {
    document.addEventListener(type, handler)
    window.ankingEventListeners.push([type, handler])
  }
</script>

<!-- Shortcut Matcher Function -->
<script>
  var specialCharCodes = {
    "-": "minus",
    "=": "equal",
    "[": "bracketleft",
    "]": "bracketright",
    ";": "semicolon",
    "'": "quote",
    "`": "backquote",
    "\\": "backslash",
    ",": "comma",
    ".": "period",
    "/": "slash",
  };
  // Returns function that match keyboard event to see if it matches given
shortcut.
  function shortcutMatcher(shortcut) {
    let shortcutKeys = shortcut.toLowerCase().split(/[+]/).map(key =>
key.trim())
    let mainKey = shortcutKeys[shortcutKeys.length - 1]
    if (mainKey.length === 1) {
      if (/\d/.test(mainKey)) {
        mainKey = "digit" + mainKey
      } else if (/[a-zA-Z]/.test(mainKey)) {
        mainKey = "key" + mainKey
      } else {
        let code = specialCharCodes[mainKey];
        if (code) {
          mainKey = code
        }
      }
    }
    let ctrl = shortcutKeys.includes("ctrl")
    let shift = shortcutKeys.includes("shift")
    let alt = shortcutKeys.includes("alt")

    let matchShortcut = function (ctrl, shift, alt, mainKey, event) {
      if (mainKey !== event.code.toLowerCase()) return false
      if (ctrl !== (event.ctrlKey || event.metaKey)) return false
      if (shift !== event.shiftKey) return false
      if (alt !== event.altKey) return false
      return true
    }.bind(window, ctrl, shift, alt, mainKey)

    return matchShortcut
  }
</script>

<!-- HINT BUTTONS SETUP -->
```

```
<script>
    (function() {
      window.toggleHintBtn = function(containerId, noScrolling=false) {
        const container = document.getElementById(containerId)
        const link = container.getElementsByTagName("a")[0]
        const button = container.getElementsByTagName("button")[0]
        const hint = container.getElementsByTagName("div")[0]

        if (hint.style.display == "none") {
          button.classList.add("expanded-button")
          hint.style.display = "block"
          link.style.display = "none"
          if (ScrollToButton && !noScrolling) {
            hint.scrollIntoView({
              behavior: "smooth", // "auto" for instant scrolling
              block: "start",
              inline: "nearest"
            });
          }
        } else {
          button.classList.remove("expanded-button")
          hint.style.display = "none"
          link.style.display = ""
        }
      }

      window.toggleNextButton = function(){
        // adapted from Hint Hotkey add-on
        var customEvent = document.createEvent('MouseEvents');
        customEvent.initEvent('click', false, true);
        var arr = document.getElementsByTagName('a');
        for (var i=0; i<arr.length; i++) {
          var el = arr[i];
          if (el.style.display === 'none') {
            continue;
          }
          if (el.classList.contains("hint")) {
            el.dispatchEvent(customEvent);
            break
          }
        }
      }

      const isToggleNextShortcut = shortcutMatcher(ToggleNextButtonShortcut)
      ankingAddEventListener("keydown", (evt) => {
        if (evt.repeat) return
        if (isToggleNextShortcut(evt)) {
          toggleNextButton()
        }
      })

      const setupHintBtn = function(elem) {
        const containerId = elem.id
        const fieldName = elem.dataset.name
        const button = elem.getElementsByClassName("button")[0]

        if (ButtonAutoReveal[fieldName]) {
          toggleHintBtn(containerId, noScrolling=true)
        }

        const isShortcut = shortcutMatcher(ButtonShortcuts[fieldName])
        const isToggleAllShortcut = shortcutMatcher(ToggleAllButtonsShortcut)
        ankingAddEventListener("keydown", (evt) => {
          if (evt.repeat) return
```

```
          if (isShortcut(evt) || isToggleAllShortcut(evt)) {
            toggleHintBtn(containerId)
          }
        })
      }

      const hints = document.getElementsByClassName("hintBtn")
      for (let i = 0; i < hints.length; i++) {
        setupHintBtn(hints[i])
      }
    })()
    </script>

<!-- AUTOFLIP BACK SCRIPT -->
<script>
    // autoflip hides card in front template
    document.getElementById("qa").style.removeProperty("display")
</script>

<!-- CLOZE ONE BY ONE SCRIPT -->
<style>
  .cloze-replacer:hover {
    cursor: pointer;
  }
  .cloze-hidden {
    display: none;
  }
  .cloze-replacer .hidden {
    display: none;
  }
  .cloze-hint {
    color: #009400 !important;
  }
  #extra.hidden {
    display: none;
  }
</style>

<script>
  (function() {
    var clozeOneByOneEnabled = true;
      clozeOneByOneEnabled = document.getElementById("one-by-one").textContent !
== "";

    if (!clozeOneByOneEnabled) {
      return
    }

    // Needed for amboss to recognize first word in .cloze-hidden
    const CLOZE_REPLACER_SEP = "<span class='hidden'> </span>"

    const hideAllCloze = function(initial) {
      let clozes = document.getElementsByClassName("cloze")
      let count = 0 // hidden cloze count
      for (const cloze of clozes) {
        const existingHidden = cloze.getElementsByClassName("cloze-hidden")[0]
        if (existingHidden) {
          revealCloze(cloze);
        }
        if (cloze.offsetWidth === 0) {
          continue
        }
        const clozeReplacer = document.createElement("span")
        const clozeHidden = document.createElement("span")
```

```
        clozeReplacer.classList.add("cloze-replacer")
        clozeHidden.classList.add("cloze-hidden")
        while (cloze.childNodes.length > 0) {
          clozeHidden.appendChild(cloze.childNodes[0])
        }
        cloze.appendChild(clozeReplacer)
        cloze.appendChild(clozeHidden)

        if (window.clozeHints && window.clozeHints[count]) {
          clozeReplacer.classList.add("cloze-hint")
          clozeReplacer.innerHTML = window.clozeHints[count] +
CLOZE_REPLACER_SEP
        } else {
          clozeReplacer.innerHTML = clozeHider(cloze) + CLOZE_REPLACER_SEP
        }
        count += 1
        if (initial) {
          cloze.addEventListener("touchend", revealClozeClicked)
          cloze.addEventListener("click", revealClozeClicked)
        }
      }
      const extra = document.getElementById("extra");
      if (extra) {
        extra.classList.add("hidden");
      }
    }

    const revealCloze = function(cloze) {
      const clozeReplacer = cloze.getElementsByClassName("cloze-replacer")[0]
      const clozeHidden = cloze.getElementsByClassName("cloze-hidden")[0]
      if (!clozeReplacer || !clozeHidden) return;

      cloze.removeChild(clozeReplacer)
      cloze.removeChild(clozeHidden)
      while (clozeHidden.childNodes.length > 0) {
        cloze.appendChild(clozeHidden.childNodes[0])
      }
      maybeRevealExtraField()
    }

    const revealClozeWord = function(cloze) {
      const clozeReplacer = cloze.getElementsByClassName("cloze-replacer")[0]
      const clozeHidden = cloze.getElementsByClassName("cloze-hidden")[0]
      if (!clozeReplacer || !clozeHidden) return;

      let range = new Range()
      range.setStart(clozeHidden, 0)
      const foundSpace = setRangeEnd(range, clozeHidden, "beforeFirstSpace")
      if (!foundSpace) {
        range.setEnd(clozeHidden, clozeHidden.childNodes.length)
      }
      let fragment = range.extractContents()
      cloze.insertBefore(fragment, clozeReplacer)
      // Extract whitespaces after word
      range = new Range()
      range.setStart(clozeHidden, 0)
      const foundWord = setRangeEnd(range, clozeHidden, "beforeFirstChar")
      if (!foundWord) {
        range.setEnd(clozeHidden, clozeHidden.childNodes.length)
      }
      fragment = range.extractContents();
      cloze.insertBefore(fragment, clozeReplacer)
      if (!foundWord) {
        cloze.removeChild(clozeHidden)
```

```
    cloze.removeChild(clozeReplacer)
    maybeRevealExtraField()
    return
  }
  clozeReplacer.innerHTML = clozeHider(clozeHidden) + CLOZE_REPLACER_SEP
  if (clozeReplacer.classList.contains("cloze-hint")) [
    clozeReplacer.classList.remove("cloze-hint")
  ]
  maybeRevealExtraField()
}

const revealNextClozeOf = (type) => {
  const nextHidden = document.querySelector(".cloze-hidden")
  if(!nextHidden) {
      return
  }
  const cloze = clozeElOfClozeHidden(nextHidden);
  if (type === "word") {
      revealClozeWord(cloze)
  } else if (type === "cloze") {
      revealCloze(cloze)
  } else {
    console.error("Invalid type: " + type)
  }
}

const revealClozeClicked = function(ev) {
  let elem = ev.currentTarget
  if (!ev.altKey && (revealNextClozeMode !== "word")) {
    revealCloze(elem)
  } else {
    revealClozeWord(elem)
  }
  ev.stopPropagation()
  ev.preventDefault()
}

window.revealNextCloze = function() {
  revealNextClozeOf(revealNextClozeMode)
}

window.toggleAllCloze = function() {
  let elems = document.querySelectorAll(".cloze-hidden")
  if(elems.length > 0) {
    for (const elem of elems) {
      const cloze = clozeElOfClozeHidden(elem)
      revealCloze(cloze)
    }
  } else {
    hideAllCloze(initial=false)
  }
}

const clozeElOfClozeHidden = (cloze) => {
  while (!cloze.classList.contains("cloze")) {
    cloze = cloze.parentElement;
  }
  return cloze;
}

const maybeRevealExtraField = () => {
  let elems = document.querySelectorAll(".cloze-hidden")
  if (elems.length == 0) {
    const extra = document.getElementById("extra")
```

```
    if (extra) {
      extra.classList.remove("hidden")
    }
  }
}

/**
 * mode: 'beforeFirstSpace' or 'beforeFirstChar'
 * Return `true` if it exists and setEnd() was called, otherwise `false`
 */
const setRangeEnd = function(range, node, mode) {
  if (node.nodeType === Node.TEXT_NODE) {
    const regex = mode === 'beforeFirstSpace' ? /\s/ : /\S/
    const match = node.textContent.match(regex)
    if (match) {
      if (match.index === 0) {
        while (node.previousSibling === null) {
          node = node.parentElement
        }
        range.setEndBefore(node)
      } else {
        range.setEnd(node, match.index);
      }
      return true;
    } else {
      return false;
    }
  } else if (mode === 'beforeFirstChar' && isCharNode(node)) {
    range.setEndBefore(node)
    return true
  } else if (!ignoreSpaceInNode(node)) {
    for (const child of node.childNodes) {
      if (setRangeEnd(range, child, mode)) {
        return true;
      }
    }
    return false;
  }
}

const ignoreSpaceInNode = function (node) {
  return node.tagName === "MJX-ASSISTIVE-MML"
}

const isCharNode = function(node) {
  return ["IMG", "MJX-CONTAINER"].includes(node.tagName)
}

hideAllCloze(initial=true)

let isShowNextShortcut = shortcutMatcher(window.revealNextShortcut)
let isShowWordShortcut = shortcutMatcher(window.revealNextWordShortcut)
let isToggleAllShortcut = shortcutMatcher(window.toggleAllShortcut)
ankingAddEventListener("keydown", (ev) => {
  let next = isShowNextShortcut(ev)
  let word = isShowWordShortcut(ev)
  let all = isToggleAllShortcut(ev)
  if (next) {
    revealNextClozeOf("cloze")
  } else if (word) {
    revealNextClozeOf("word")
  } else if (all) {
    toggleAllCloze()
  } else {
```

```
        return;
      }
      ev.stopPropagation()
      ev.preventDefault()
    });
  })()
</script>

<!-- CLICKABLE COLORFUL TAGS -->
{{#Tags}}
<div id="tags-container">{{clickable::Tags}}</div>
<script>
  var tagContainer = document.getElementById("tags-container")
  if (tagContainer.childElementCount == 0) {
    var tagList = tagContainer.innerHTML.split(" ");
    var kbdList = [];
    var newTagContent = document.createElement("div");

    for (var i = 0; i < tagList.length; i++) {
      var newTag = document.createElement("kbd");
      var tag = tagList[i];
      // numTagLevelsToShow == 0 means the whole tag should be shown
      if(numTagLevelsToShow != 0){
        tag = tag.split('::').slice(-numTagLevelsToShow).join("::");
      }
      newTag.innerHTML = tag;
      newTagContent.append(newTag)
    }
    tagContainer.innerHTML = newTagContent.innerHTML;
    tagContainer.style.cursor = "default";
  }
  if (tagContainer.innerHTML.indexOf(tagID) != -1) {
    tagContainer.style.backgroundColor = "rgba(251,11,11,.15)";
  }

  function showtags() {
    var tagContainerShortcut = document.getElementById("tags-container");

    if (tagContainerShortcut.style.display
      === "none") {
      tagContainerShortcut.style.display = "block";
    } else {
      tagContainerShortcut.style.display =
        "none";
    }
  }

  var isShortcut = shortcutMatcher(toggleTagsShortcut)
  ankingAddEventListener('keyup', function (e) {
      if (isShortcut(e)) {
          showtags();
      }
  });

</script>
{{/Tags}}

<!-- WIKIPEDIA SEARCHES -->
<div id="popup-container">
  <button id="close-popup-btn" onclick="closePopup(true)">&times;</button>
  <a id="open-wiki-btn" href="">&#8618;</a>
  <div id="tc"></div>
  <div id="fadebottom_v"></div>
  <div id="ic"><img id="popup-image"></div>
```

```
</div>
<style>
  #tc {
    color: #222222;
    position: absolute;
    top: 16px;
    margin: 0px;
    left: 15px;
    text-decoration: none;
    height: 320px;
    overflow: hidden;
    overflow-y: scroll;
    white-space: pre-wrap;
    width: 300px;
  }

  #tc p {
    margin: 0px;
  }

  #tc::-webkit-scrollbar {
    display: none;
  }

  #fadebottom_v {
    height: 30px;
    width: 300px;
    background: -webkit-linear-gradient(270deg, rgba(255, 255, 255, 0.1),
rgba(255, 255, 255, 1));
    z-index: 111;
    position: absolute;
    bottom: 0px;
    left: 15px;
  }

  #hc {
    color: #666;
    font-weight: bold;
  }

  #ic {
    right: 0px;
    top: 30px;
    position: absolute;
  }

  #ic img {
    width: 160px;
    height: auto;
    object-fit: cover;
    overflow: hidden;
  }

  #popup-image {
    width: 140px;
    height: auto;
  }

  #popup-container {
    background: #fff;
    position: absolute;
    bottom: 30px;
    right: 10px;
    z-index: 110;
```

```
    -webkit-box-shadow: 0 30px 90px -20px rgba(0, 0, 0, 0.3), 0 0 1px 1px
rgba(0, 0, 0, 0.05);
    box-shadow: 0 30px 90px -20px rgba(0, 0, 0, 0.3), 0 0 1px 1px rgba(0, 0, 0,
0.05);
    padding: 0;
    display: none;
    font-size: 17px;
    line-height: 20px;
    border-radius: 2px;
    width: 480px;
    height: 340px;
    overflow: hidden;
    font-family: Arial;
    text-align: left;
    border: 1px solid #d0d0d0;
    border-radius: 5px;
  }

  #close-popup-btn {
    position: absolute;
    top: 1px;
    right: 5px;
    width: 32px;
    height: 32px;
    background: none;
    border: 0;
    cursor: pointer;
    font-family: 'Josefin Sans', sans-serif;
    font-size: 20px;
    outline: none;
    text-align: right;
    z-index: 112;
  }

  #open-wiki-btn {
    position: absolute;
    top: 10px;
    right: 30px;
    width: 15px;
    height: 32px;
    background: none;
    border: 0;
    cursor: pointer;
    text-decoration: none;
    color: #222222;
    font-family: 'Josefin Sans', sans-serif;
    font-size: 17px;
    outline: none;
    text-align: left;
    z-index: 112;
  }
</style>

<script>
  function getSummaryFor(word) {
    word = word.replace(/^[\.,\/#\!$%\^&\*;:{}=\-_`~() \'\s]+|[\.,\/#\!$%\^&\*;:
{}=\-_`~()\'\s]+$/g, "");
    var pc = document.getElementById("popup-container");
    var hc = document.getElementById("hc");
    var tc = document.getElementById("tc");
    var ic = document.getElementById("ic");
    var imgelem = document.getElementById("popup-image");
    imgelem.src = "";
    var shortsum = "";
```

```
    fetch("https://en.wikipedia.org/api/rest_v1/page/summary/" + word)
    .then(function (response) { return response.json(); })
    .then(function (response) {
      shortsum = response.description;
      shortsum = shortsum.replace(/(Disambiguation.*)/g, "Disambiguation");
      tc.innerHTML = "<span id='hc'>" + capfl(shortsum) + "</span>" + "\n" +
response.extract_html + "\n";
      tc.style.width = "420px";
      if (response.extract_html && !response.extract.endsWith("to:")) {
        pc.style.display = "block";
        document.getElementById("open-wiki-btn").href =
response.content_urls.desktop.page;
      } else {
        pc.style.display = "none";
      }
      if (!response.thumbnail.source || response.type === "disambiguation") {
        tc.style.width = "420px";
      } else {
        tc.style.width = "300px"; imgelem.src = response.thumbnail.source;
      }
    })
    .catch(function (error) {
      console.log(error);
    });
  }

  function closePopup(deselectAlso = false) {
    pcc.style.display = 'none';
    if (deselectAlso) { clearSelection(); }
  }

  var pcc = document.getElementById("popup-container");
  var prevSel = "";
  ankingAddEventListener('click', function () {
    var currentSelection = getSelectionText();
    if (currentSelection !== "") { prevSel = currentSelection; }
    if (currentSelection && !mustClickW) {
      getSummaryFor(currentSelection);
    } else { closePopup(); }
  });

  ankingAddEventListener('keyup', function (e) {
    if (e.key == "w") {
      if (pcc.style.display === "block") { closePopup(); } else
{ getSummaryFor(prevSel); }
    }
  });

  function getSelectionText() {
    var text = "";
    if (window.getSelection) {
      text = window.getSelection().toString();
    } else if (document.selection && document.selection.type != "Control")
{ text = document.selection.createRange().text; }
    return text;
  }

  function capfl(s) {
    return s.charAt(0).toUpperCase() + s.slice(1);
  }

  function clearSelection() {
    if (window.getSelection) { window.getSelection().removeAllRanges(); }
```

```
    else if (document.selection) { document.selection.empty(); }
  }

  //CUSTOMIZATION
  //this is a variable controlling whether user must click the "w" key to open
the popup.
  //if set to true: user must select text, then click the "w" key to open
wikipedia popup. Clicking "w" key again will close the popup.
  //if set to false: user only needs to select text. popup will open
automatically. Clicking "w" is an alternative but not obligatory way of
opening/closing the popup in this mode.
  //BELOW SET to true or to false.
  var mustClickW = true;
  //END CUSTOMIZATION
</script>
```