AI Blur Extension

Volker Weber's konstante Versuche KI überall raus zu bekommen brachte mich auf die Idee eine Firefox Extension zu machen, die schlichtweg jeden Absatz der bestimmte Schlagwörter enthält ausblendet...

Auch die ist vollkommen Vibe-Coded mit KI - so als kleine Rebellion... Also auf eigene Gefahr. Veröffentlicht ist noch nix, Kinderkrankheiten gibts auch nocht... Der "Versperren" Button ist bei Google Suchtreffern zum Beispiel noch verkehrt herum und eigentlich mag ich Verschwommenheit nicht so richtig. Das müssen irgendwann klare Blockelemente sein die nach Paywall aussehen müssen...

> VIBE CODED - auf eure eigene Gefahr

Dateien in Ordner schreiben, Firefox about:debugging aufrufen und via "Dieser Firefox" eine temporäre Erweiterung laden.

So wie sie ist kommt die Erweiterung ohne Icons. Generiert euch ein paar mit irgendeiner, verkleinert die auf 96x96 und 48x48 und legt die als PNG in einen assets-Unterordner, wenn ihr denn ein Icon haben wollt...

content.js

```
// Standardwörter laden
const DEFAULT_KEYWORDS = ['KI', 'AI', 'künstliche Intelligenz', 'Artificial Intelligence'];

let customKeywords = [];
let blurredElements = new Set();

// Auf Speicher-Änderungen hören
browser.storage.onChanged.addListener((changes) => {
  if (changes.customKeywords) {
    customKeywords = changes.customKeywords.newValue || [];
    // Reset und neu verarbeiten
    blurredElements.clear();
    processPage();
  }
});
```

```
// Gespeicherte benutzerdefinierte Wörter laden
async function loadCustomKeywords() {
  try {
    const result = await browser.storage.local.get('customKeywords');
    customKeywords = result.customKeywords || [];
  } catch (error) {
    console.error('Fehler beim Laden der Keywords:', error);
  }
}

// Prüfe ob Text ein Keyword enthält (nur als ganzes Wort)
function containsKeyword(text, keywords) {
  for (let keyword of keywords) {
    // Erstelle Regex mit Word Boundaries
    // Escape special regex characters
    const escapedKeyword = keyword.replace(/[.*+?^${}()|[\]\\]/g, '\\$&');
    // \b = Word boundary (nur ganze Wörter)
    const regex = new RegExp(`\\b${escapedKeyword}\\b`, 'gi');
    if (regex.test(text)) {
      return true;
    }
  }
  return false;
}

// Seite verarbeiten
async function processPage() {
  await loadCustomKeywords();
  const allKeywords = [...DEFAULT_KEYWORDS, ...customKeywords];

  // Überschriften verarbeiten (zuerst!)
  ['h1', 'h2', 'h3', 'h4', 'h5', 'h6'].forEach(tag => {
    document.querySelectorAll(tag).forEach(heading => {
      if (blurredElements.has(heading)) return;

      const text = heading.textContent;
      if (containsKeyword(text, allKeywords)) {
        blurElement(heading, true);
        blurredElements.add(heading);
```

```
    }
  });
});

// Paragraphen und Text-Container verarbeiten
const walker = document.createTreeWalker(
  document.body,
  NodeFilter.SHOW_ELEMENT,
  {
    acceptNode: function(node) {
      // Ignoriere Script, Style, und bereits verarbeitete Elemente
      if (node.tagName.match(/^(SCRIPT|STYLE|NOSCRIPT)$/i)) {
        return NodeFilter.FILTER_REJECT;
      }
      if (blurredElements.has(node)) {
        return NodeFilter.FILTER_REJECT;
      }
      // Nur p, div, span, article, section verarbeiten (größere Container)
      if (node.tagName.match(/^(P|DIV|SPAN|ARTICLE|SECTION|BLOCKQUOTE)$/i)) {
        return NodeFilter.FILTER_ACCEPT;
      }
      return NodeFilter.FILTER_SKIP;
    }
  },
  false
);

let node;
while (node = walker.nextNode()) {
  // Prüfe nur direkten Text, nicht Text in Kind-Elementen
  let hasKeyword = false;
  for (let child of node.childNodes) {
    if (child.nodeType === Node.TEXT_NODE) {
      if (containsKeyword(child.textContent, allKeywords)) {
        hasKeyword = true;
        break;
      }
    }
  }
```

```
    if (hasKeyword) {
      blurElement(node, false);
      blurredElements.add(node);
    }
  }
}

// Element verschwommen darstellen
function blurElement(element, isHeading = false) {
  // Für Überschriften: stärkerer Blur
  const blurAmount = isHeading ? '12px' : '8px';

  // State für Toggle speichern
  let isBlurred = true;

  // Wrapper für den Unlock-Button erstellen
  let unlockButton = null;
  let buttonContainer = null;

  // Styling für das Element
  element.style.cursor = 'pointer';
  element.style.transition = 'filter 0.3s ease';

  // Initial verschwommen
  updateBlur(true);

  // Click-Handler für Entsperrung
  element.addEventListener('click', (e) => {
    // Ignoriere Klicks auf den Button selbst
    if (unlockButton && e.target === unlockButton) {
      return;
    }

    // Wenn versperrt: Entsperren und Klick blockieren
    if (isBlurred) {
      e.preventDefault();
      e.stopPropagation();
      isBlurred = false;
      updateBlur(false);
    }
```

```
  // Wenn entsperrt: Link-Klicks erlauben (kein preventDefault)
});

function updateBlur(blur) {
  if (blur) {
    element.style.filter = `blur(${blurAmount})`;
    element.style.opacity = '0.8';
    element.style.userSelect = 'none';
    element.style.pointerEvents = 'auto';
    element.title = 'Klick zum Lesen';

    // Button entfernen wenn vorhanden
    if (buttonContainer) {
      buttonContainer.remove();
      buttonContainer = null;
      unlockButton = null;
    }
  } else {
    element.style.filter = 'blur(0px)';
    element.style.opacity = '1';
    element.style.userSelect = 'auto';
    element.style.pointerEvents = 'auto';
    element.title = '';

    // Button-Container neben dem Element erstellen
    buttonContainer = document.createElement('span');
    buttonContainer.style.cssText = `
      display: inline-block;
      margin-left: 8px;
      white-space: nowrap;
    `;

    // Button hinzufügen zum Wieder-Versperren
    unlockButton = document.createElement('button');
    unlockButton.textContent = '□□Versperren';
    unlockButton.style.cssText = `
      padding: 6px 12px;
      background: #ff6b6b;
      color: white;
      border: none;
```

```
        border-radius: 4px;
        cursor: pointer;
        font-size: 12px;
        font-weight: bold;
        box-shadow: 0 2px 8px rgba(0, 0, 0, 0.2);
        transition: background 0.2s ease;
        vertical-align: middle;
      `;

      unlockButton.addEventListener('mouseover', () => {
        unlockButton.style.background = '#ff5252';
      });

      unlockButton.addEventListener('mouseout', () => {
        unlockButton.style.background = '#ff6b6b';
      });

      unlockButton.addEventListener('click', (e) => {
        e.preventDefault();
        e.stopPropagation();
        isBlurred = true;
        updateBlur(true);
      });

      buttonContainer.appendChild(unlockButton);
      element.parentNode.insertBefore(buttonContainer, element.nextSibling);
    }
  }
}

// Initial verarbeiten
processPage();

// Bei dynamischen Inhalten neu prüfen
const observer = new MutationObserver((mutations) => {
  // Nur wenn tatsächlich neue Nodes hinzugefügt werden
  let shouldProcess = false;

  for (let mutation of mutations) {
    if (mutation.type === 'childList' && mutation.addedNodes.length > 0) {
```

```
      shouldProcess = true;
      break;
    }
  }

  if (shouldProcess) {
    // Kleine Verzögerung um Flimmern zu vermeiden
    setTimeout(() => {
      processPage();
    }, 100);
  }
});

observer.observe(document.body, {
  childList: true,
  subtree: true
});
```

options.js

```
let customKeywords = [];

// Optionen beim Laden laden
async function loadOptions() {
  try {
    const result = await browser.storage.local.get('customKeywords');
    customKeywords = result.customKeywords || [];
    displayKeywords();
  } catch (error) {
    console.error('Fehler beim Laden der Optionen:', error);
  }
}

// Alle Stichwörter anzeigen
function displayKeywords() {
  const list = document.getElementById('keywordsList');
  list.innerHTML = '';

  if (customKeywords.length === 0) {
    list.innerHTML = '<p style="color: #999;">Noch keine benutzerdefinierten Wörter
```

```
hinzugefügt.</p>';
    return;
  }

  customKeywords.forEach((keyword, index) => {
    const div = document.createElement('div');
    div.className = 'keyword-item';
    div.innerHTML = `
      <span><strong>${escapeHtml(keyword)}</strong></span>
      <button class="remove-btn" data-index="${index}">Löschen</button>
    `;
    list.appendChild(div);
  });
}

// Wort hinzufügen
async function addKeyword() {
  const input = document.getElementById('newKeyword');
  const keyword = input.value.trim();

  if (!keyword) {
    showStatus('Bitte geben Sie ein Wort ein!', 'error');
    return;
  }

  if (customKeywords.includes(keyword)) {
    showStatus('Dieses Wort existiert bereits!', 'error');
    return;
  }

  if (keyword.length > 50) {
    showStatus('Das Wort ist zu lang (max. 50 Zeichen)!', 'error');
    return;
  }

  customKeywords.push(keyword);

  try {
    await browser.storage.local.set({ customKeywords });
    showStatus(`"${keyword}" hinzugefügt!`, 'success');
```

```
    input.value = '';
    displayKeywords();
  } catch (error) {
    showStatus('Fehler beim Speichern!', 'error');
    console.error(error);
  }
}

// Wort löschen (Event Delegation)
async function removeKeyword(index) {
  const keyword = customKeywords[index];
  customKeywords.splice(index, 1);

  try {
    await browser.storage.local.set({ customKeywords });
    showStatus(`"${keyword}" gelöscht!`, 'success');
    displayKeywords();
  } catch (error) {
    showStatus('Fehler beim Löschen!', 'error');
    console.error(error);
  }
}

// Status-Meldung anzeigen
function showStatus(message, type) {
  const status = document.getElementById('status');
  status.textContent = message;
  status.style.display = 'block';
  status.style.background = type === 'error' ? '#ffebee' : '#d4edda';
  status.style.color = type === 'error' ? '#c62828' : '#155724';
  status.style.borderColor = type === 'error' ? '#ef5350' : '#28a745';

  setTimeout(() => {
    status.style.display = 'none';
  }, 3000);
}

// XSS-Schutz
function escapeHtml(text) {
  const div = document.createElement('div');
```

```
  div.textContent = text;
  return div.innerHTML;
}

// Beim Seitenaufrufen laden und Event-Listener zuweisen
document.addEventListener('DOMContentLoaded', () => {
  loadOptions();

  // Button-Click Listener für Hinzufügen
  const addButton = document.querySelector('.add-form button');
  if (addButton) {
    addButton.addEventListener('click', addKeyword);
  }

  // Enter-Taste zum Hinzufügen
  const input = document.getElementById('newKeyword');
  if (input) {
    input.addEventListener('keypress', (e) => {
      if (e.key === 'Enter') {
        e.preventDefault();
        addKeyword();
      }
    });
  }

  // Event Delegation für Löschen-Buttons
  const keywordsList = document.getElementById('keywordsList');
  if (keywordsList) {
    keywordsList.addEventListener('click', (e) => {
      if (e.target && e.target.classList.contains('remove-btn')) {
        const index = parseInt(e.target.dataset.index, 10);
        removeKeyword(index);
      }
    });
  }
});
```

popup.js

```
function openOptions() {
  browser.runtime.openOptionsPage().catch(error => {
    console.error('Fehler beim Öffnen der Optionsseite:', error);
  });
}

// Alternativ: Button-Click direkt zuweisen
document.addEventListener('DOMContentLoaded', () => {
  const button = document.querySelector('button');
  if (button) {
    button.addEventListener('click', openOptions);
  }
});
```

options.html

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>AI Blur - Optionen</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: system-ui, -apple-system, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            padding: 30px 20px;
            min-height: 100vh;
        }

        .container {
            max-width: 600px;
            margin: 0 auto;
            background: white;
```

```
        padding: 30px;
        border-radius: 12px;
        box-shadow: 0 10px 40px rgba(0, 0, 0, 0.2);
    }

    h1 {
        color: #667eea;
        margin-bottom: 20px;
    }

    .section {
        margin-bottom: 30px;
    }

    .section h2 {
        color: #764ba2;
        font-size: 1.2em;
        margin-bottom: 15px;
        border-bottom: 2px solid #667eea;
        padding-bottom: 10px;
    }

    .section p {
        color: #666;
        margin-bottom: 15px;
    }

    .keyword-item {
        display: flex;
        gap: 10px;
        margin-bottom: 10px;
        padding: 10px;
        background: #f7f7f7;
        border-radius: 6px;
        align-items: center;
    }

    .keyword-item span {
        flex: 1;
    }
```

```
input[type="text"] {
    flex: 1;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 6px;
    font-size: 1em;
}

button {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 6px;
    cursor: pointer;
    font-weight: bold;
    transition: transform 0.2s;
}

button:hover {
    transform: translateY(-2px);
}

.remove-btn {
    background: #f5576c;
    padding: 8px 12px;
}

.remove-btn:hover {
    background: #d63447;
}

#keywordsList {
    max-height: 300px;
    overflow-y: auto;
    margin-bottom: 15px;
}

.status {
```

```
            padding: 10px;
            background: #d4edda;
            border: 1px solid #28a745;
            color: #155724;
            border-radius: 6px;
            margin-bottom: 15px;
            display: none;
        }

        .add-form {
            display: flex;
            gap: 10px;
            margin-bottom: 15px;
        }

        ul {
            margin-left: 20px;
            color: #666;
        }

        ul li {
            margin-bottom: 5px;
        }

        .section h3 {
            color: #667eea;
            margin-bottom: 10px;
            font-size: 1em;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>□□AI Blur Extension Einstellungen</h1>

        <div class="section">
            <h2>Benutzerdefinierte Wörter</h2>
            <p>Zusätzliche Wörter hinzufügen die automatisch verschwommt werden sollen:</p>

            <div class="add-form">
```

```
                <input type="text" id="newKeyword" placeholder="z.B. ChatGPT, Neural..."
maxlength="50">
                <button onclick="addKeyword()">Hinzufügen</button>
            </div>

            <div class="status" id="status"></div>

            <h3>Aktuell konfigurierte Wörter:</h3>
            <div id="keywordsList"></div>
        </div>

        <div class="section">
            <h2>Standard-Wörter</h2>
            <p>Diese Wörter werden immer automatisch erkannt:</p>
            <ul>
                <li>KI</li>
                <li>AI</li>
                <li>künstliche Intelligenz</li>
                <li>Artificial Intelligence</li>
            </ul>
        </div>
    </div>

    <script src="options.js"></script>
</body>
</html>
```

popup.html

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>AI Blur Popup</title>
    <style>
        body {
            width: 350px;
            padding: 15px;
            font-family: system-ui, -apple-system, sans-serif;
```

```
            background: #f5f5f5;
            margin: 0;
        }

        h2 {
            color: #667eea;
            margin: 0 0 15px 0;
            font-size: 1.1em;
        }

        .info {
            background: #e3f2fd;
            border-left: 4px solid #667eea;
            padding: 10px;
            margin-bottom: 15px;
            border-radius: 4px;
        }

        button {
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
            border: none;
            padding: 10px 15px;
            border-radius: 6px;
            cursor: pointer;
            font-weight: bold;
            width: 100%;
            transition: transform 0.2s;
        }

        button:hover {
            transform: translateY(-2px);
        }
    </style>
</head>
<body>
    <h2>□□AI Blur Extension</h2>
    <div class="info">
        <strong>Funktioniert!</strong><br>
        KI-bezogene Texte werden verschwommt.
```

```
    </div>
    <button onclick="openOptions()">⚙ Optionen öffnen</button>
    <script src="popup.js"></script>
</body>
</html>
```

manifest.json

```
{
  "manifest_version": 3,
  "name": "AI Blur Extension",
  "version": "1.0.0",
  "description": "Verschwommelt KI-Inhalte automatisch - Klick zum Entsperren",
  "author": "Dein Name",

  "permissions": [
    "storage"
  ],

  "host_permissions": [
    "https://*/*",
    "http://*/*",
    "file:///*"
  ],

  "icons": {
    "48": "assets/icon-48.png",
    "96": "assets/icon-96.png"
  },

  "action": {
    "default_title": "AI Blur Extension Optionen",
    "default_popup": "popup.html"
  },

  "options_page": "options.html",

  "content_scripts": [
    {
      "matches": [
```

```
        "https://*/*",
        "http://*/*",
        "file:///*"
      ],
      "js": ["content.js"],
      "run_at": "document_end"
    }
  ]
}
```

Version #1
Erstellt: 2025-12-15 21:59:17 UTC von Konstantin
Zuletzt aktualisiert: 2025-12-15 22:09:53 UTC von Konstantin