BLOCKBASIERTES HTML-TOOL MIT TUTORKOMPONENTE

07.07.2023 Felix Goergen, Lena Stelzer

Gliederung

1. Aufbau des Tools
    - *1.1 Die technische Seite des Designs*
    - *1.2 Blockly-Workspace (blocklyDiv)*
    - *1.3 Wie die Blöcke zu Code werden*
    - *1.4 Tutorkomponente (DidaktikDiv)*
    - *1.5 Code und Vorschau (CodePreviewDiv)*
    - *1.6 Mainpage*
    - *1.7 Restliche Buttons*
2. Ausblick und Verbesserungsmöglichkeiten

1. Aufbau des Tools

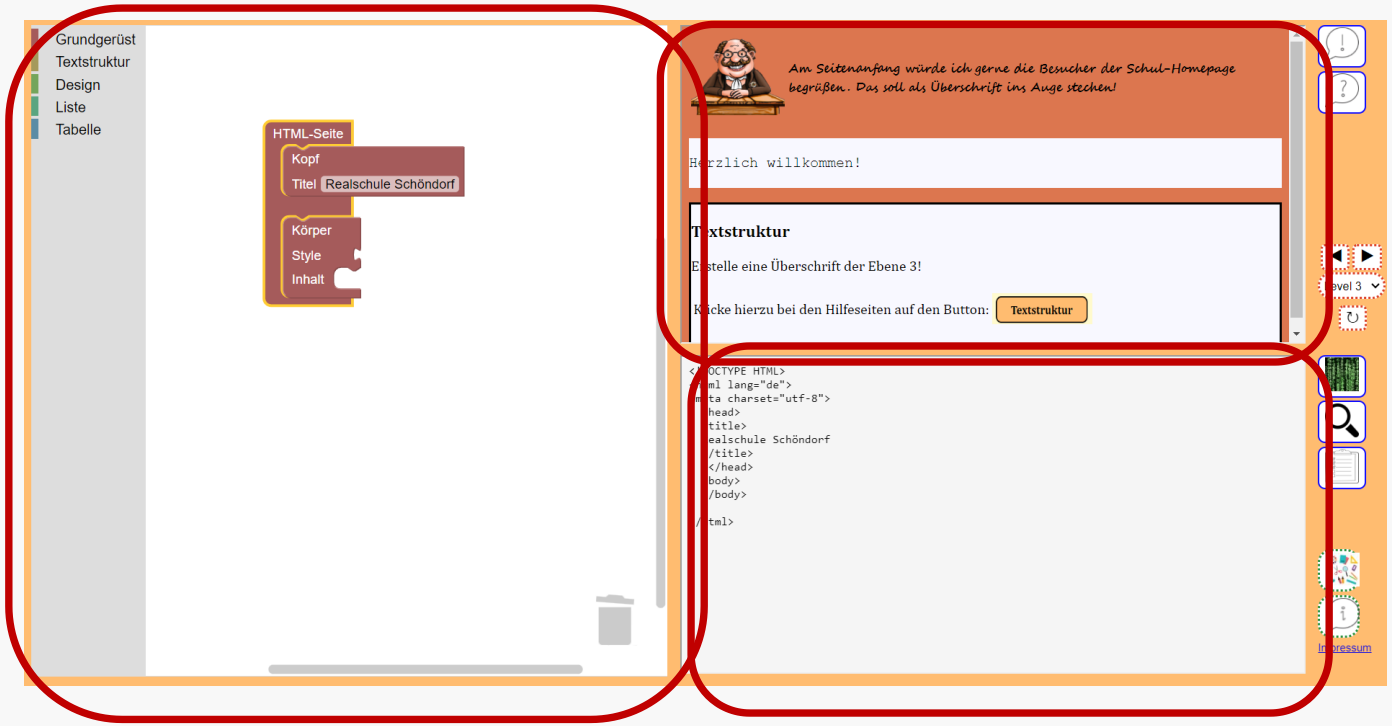
- Grundgerüst: HTML
- Seitendesign: CSS
- Funktionen: JavaScript (Google Blockly-Library)

- Zwischenergebnisse: XML

1.1 Die technische Seite des Designs
1.1.1 Seitenaufbau

- Grundstruktur der Hauptseite als grid-Raster

```
.grid-container{
  display: grid;
  grid-area: auto;
  background-color: #ffbc70;
  grid-template-areas:
    'blocklyDiv didaktikDiv codeButton'
    'blocklyDiv didaktikDiv nextLevelButton'
    'blocklyDiv didaktikDiv restartButton'
    'blocklyDiv codePreview switch'
    'blocklyDiv codePreview infoImpressum'
}
```

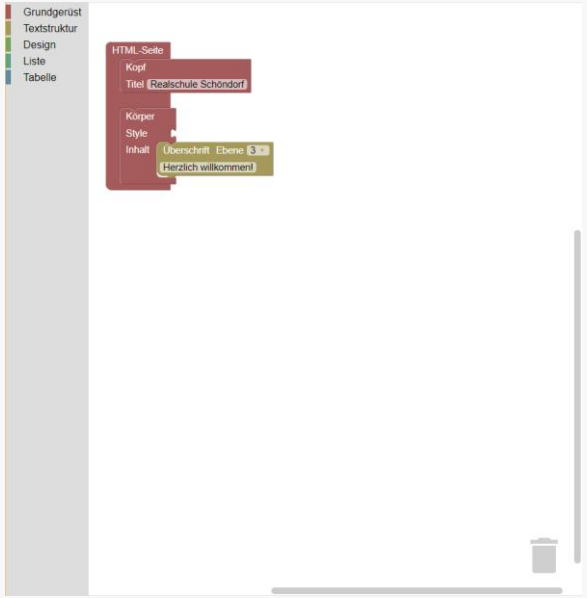


1.1 Die technische Seite des Designs
1.1.3 Bilder

- Bilder mittels Künstlicher Intelligenz DALL·E 2 erstellt *(https://openai.com/dall-e-2/)*

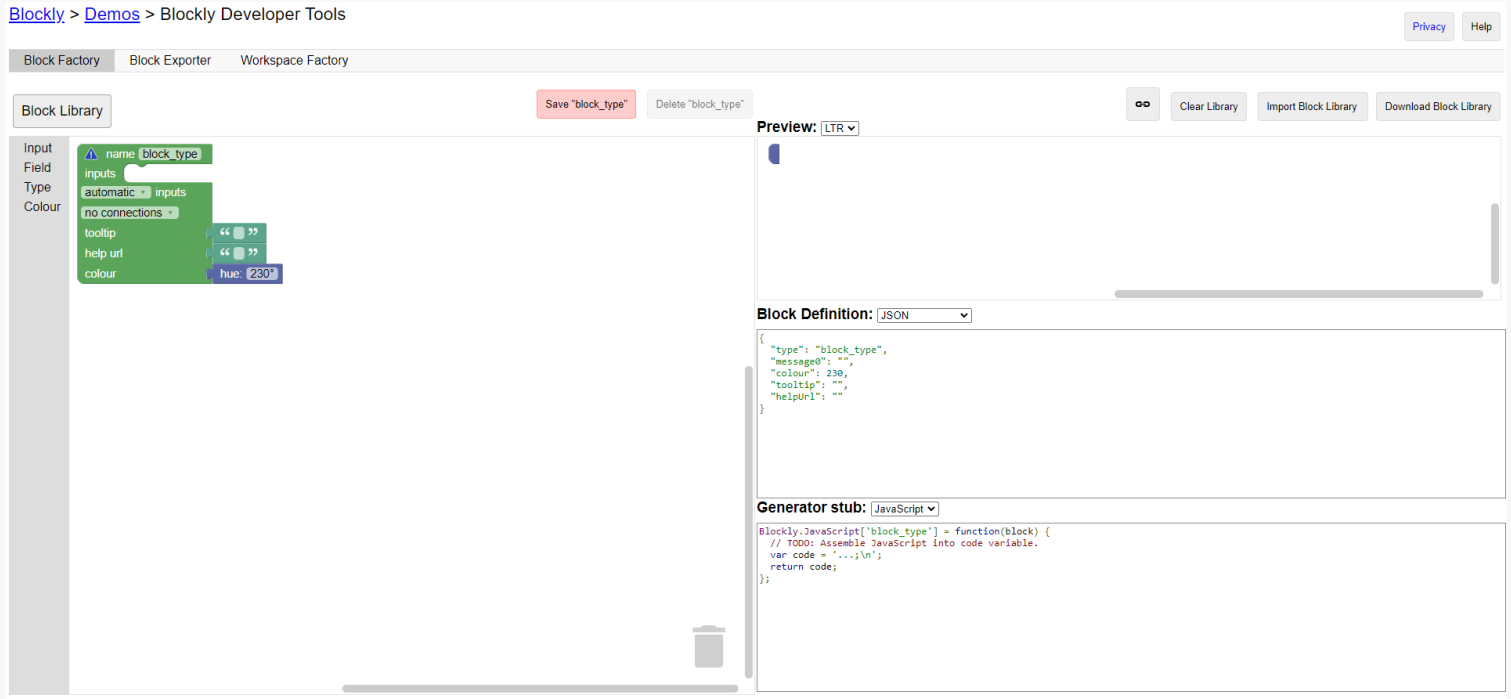
1.2 Blockly-Workspace (blocklyDiv)

- Blockly-Library in das Projekt einbinden
- Blöcke definieren
- Blöcke initialisieren (Blockly-Funktion)
- Toolbox definieren
- Toolbox in workspace laden und diesen in Division einfügen



1.2.1 Aufbau und Definition von Blöcken

- Toolbox mit verwendbaren Blöcken befüllen
- Block Factory: *https://blockly-demo.appspot.com/static/demos/blockfactory/index.html*



1.1.2 Aufbau und Definition von Blöcken – Beispiel „image“-Block

```
{
  "type": "image",
  "message0": "Bild %1 Breite %2 Höhe %3",
  "args0": [
    {
      "type": "input_statement",
      "name": "CONTENT",
      "check": "imageblock"
    },
    {
      "type": "field_number",
      "name": "WIDTH",
      "value": 240,
      "min": 1,
      "max": 2400
    },
    {
      "type": "field_number",
      "name": "HEIGHT",
      "value": 150,
      "min": 1,
      "max": 2400
    },
  ],
  "previousStatement": "image",
  "nextStatement": [...],
  "colour": 300
},
```

- type: referenzierbar in anderen Blockly-Funktionen
- message0: Reihenfolge der Beschriftung des Blocks
- args0: Array – was Block verarbeiten soll

input_statement	Möglichkeit einen beliebigen Block einzufügen
field_number	Textfeld für Zahlen

- previousStatement: welche checks, der Block erfüllen muss
- nextStatement: welche Blöcke angefügt werden dürfen
- colour: Farbe des Blocks

1.1.2 Aufbau und Definition von Blöcken – Beispiel „image“-Block

- Besondere Eigenschaften für Typechecking (Regeln, nach denen Blöcke zusammengefügt werden können) und Auswertung im Generator

```
{
  "type": "input_statement",
  "name": "CONTENT",
  "check": "imageblock"
},
```

- type: Aufbau des Blocks
- name: referenzierbar in anderen Blockly-Funktionen
- check: welche Blöcke eingesetzt werden dürfen
  →Blöcke mit „previousStatement“: „imageblock“

1.3 Wie die Blöcke zu Code werden

- Funktionen, die den einzelnen Blöcken Code zuweisen
- Generator erzeugen und diesen HTML zuweisen

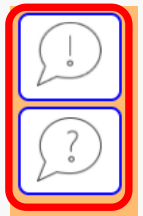
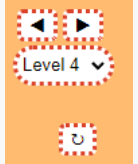

```
//Bild-Block
HtmlGenerator["image"] = function (block) {
  let statements_content = HtmlGenerator.statementToCode(block, name: "CONTENT");
  let statements_width = block.getFieldValue( name: "WIDTH");
  let statements_height = block.getFieldValue( name: "HEIGHT");

  return code = '&lt;img src="'  + statements_content + '" width="'+ statements_width
    + '" height="'+ statements_height + '"' + ' alt="default"/>\n';
};
```

1.4 Tutorkomponente (DidaktikDiv)

- IFrame: auf HTML-Seite weitere HTML-Seite integrieren
- Mittels Buttons Wechsel zwischen Aufgabenstellung und Hilfeseiten (Quelle des IFrames wird geändert)

```
//wird aufgerufen, wenn auf den Hilfe-Knopf gedrückt wird. Ändert die Source im IFrame
function showHelp(){
  document.getElementById("DidaktikDiv").src="hilfestellung.html";
}
```


Impressum

1.4 Tutorkomponente (DidaktikDiv)

Hilfeseite:

1.5 Code und Vorschau (CodePreviewDiv)

- auch ein Iframe
- sobald Workspace sich ändert → Code angezeigt
- Browserspeicher (localStorage)

```
// Update-Funktion, die aufgerufen wird, wenn sich im Workspace etwas ändert
function update(){
  // Umwandlung des workspace zu HTML-Code via HTMLGenerator
  code = HtmlGenerator.workspaceToCode(workspace);
  showCode();

  // Speichern des Zustandes des workspace in der localStorage des Browsers
  let xml = Blockly["Xml"].workspaceToDom(workspace);
  xml_text = Blockly["Xml"].domToText(xml);
  localStorage.setItem("blockly-html-code-learning", xml_text);
}
```

1.5 Code und Vorschau (CodePreviewDiv)

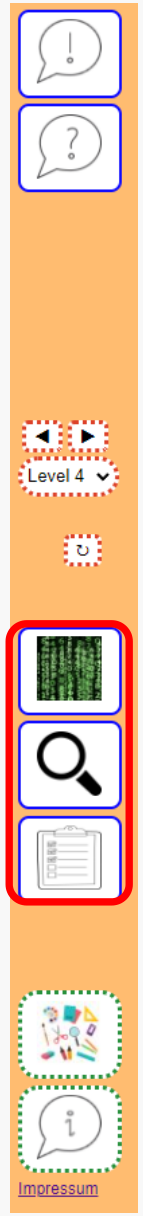
Code wird angezeigt

Vorschau des Codes

vorgefertigte Vorschau

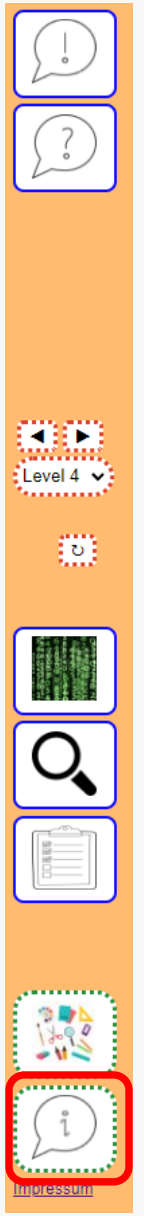
- auch ein Iframe
- Vorschau der aktuell im Workspace erzeugten HTML-Seite:

```
//function die den aktuellen Zustand der HTML-Seite anzeigt
function showCurrentState(){
  const ifr = document.preview;
  const searchRegExp = "&lt;";
  const replaceWith = "<";
  let previewCode = code.replaceAll(searchRegExp, replaceWith);
  ifr.document.write(previewCode);
  ifr.document.close();
}
```



1.6 Mainpage

- Aufruf der Hauptseite
  - *Überprüfung des Browserspeichers*
    →gespeicherte Aufgabenstellung und Code werden angezeigt
  - *PopUp und Button-Bilder an Fenstergröße angepasst*
  - *PopUp erscheint (eigene Division)*
  - *auch durch Klicken des Info-Buttons aufrufbar*



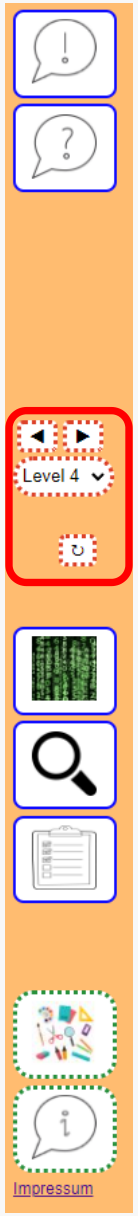
1.7 Restliche Buttons - Levelbuttons

- Auswahl des Levels
  - *Pfeilbutton (nextLevel() bzw. previousLevel())*
  - *Dropdown*
  - → *Aufruf* switchLevel(newLevel)-Funktion

- Reload-Button ↻
  - → *Endzustand des vorherigen Levels*

```
function loadState(){
  workspace.clear();
  let xmlHTTP;
  let xml;
  switch (level) {
    case 2:
      xmlHTTP= new XMLHttpRequest();
      xmlHTTP.open("GET", "state2.xml", false);
      xmlHTTP.send(null);
      xml = xmlHTTP.responseText;
      break;
          ...
  }

  let xml_new = Blockly["Xml"].textToDom(xml);
  Blockly["Xml"].domToWorkspace(xml_new,workspace);
  showCode();
}
```



1.7 Restliche Buttons

- Bastelecke
  - → *bastelecke.html wird aufgerufen*
  - → *Didaktische Komponenten entfernt*

- Impressum
  - → impressum.html wird aufgerufen

2. Ausblick und Verbesserungsmöglichkeiten

- ansprechenderes Design allgemein
- einfacherer Weg zur gewünschten Hilfeseite
- zu erstellende Website aktuell sehr schlicht
- Ansprechendere Gestaltung der Hilfeseiten
  - *Hilfevideos*
  - *Aufgaben-/Hilfestellungen einsprechen*
- Exportfunktion: Download des .html-Files