Verbessern des look-and-feel des MEXLE Wikis

Sinnvolle Vorkenntnisse: CSS, Web-UI, PHP, JS

1. Erstellen einer besseren Grundstruktur (ggf. über Nutzerbefragung)
2. ~~Anpassung von CSS und Vorlagen (näher an die Hochschul-CCS)~~
3. ~~Optimierung des Plugins~~ quizlib
    1. ~~Tipps bei falscher Auswahl~~
    2. ~~Sprachauswahl (für kombiniertes Englisch- und Deutschsprachiges Wiki)~~
    3. ~~Einbinden von Bildern~~
4. ~~Erstellen eines Falstad-Plugins~~
    1. ~~Ablegen separater Dateien für die Simulation (als Text, statt bisher als Link)~~
    2. ~~Ein-/Ausblenden von Menü und Slider~~
    3. ~~optimal wäre zstzl.: Einbinden des~~ CircuitJS ~~über Plugin-Installation~~
5. ~~Optimieren des Plugins~~ imagereference
    1. ~~Möglichkeit auch Simulationen und andere Textbausteine als Bild / Tabelle einzubinden~~
6. Optimieren des Plugins draw.io
    1. Bildgröße änderbar
    2. Ausgabe als SVG statt png
7. Entwicklung eines Lehr-Wiki-Plugins
    1. Verzeichnismanagement: Möglichkeit für Verzeichnisse, welche über einzelne Seiten hinaus gehen, z.B. für einen Namespace
        1. Abbildungsverzeichnis
        2. Tabellenverzeichnis
        3. Medienverzeichnis
        4. Formelverzeichnis
        5. Abkürzungsverzeichnis
        6. Aufgabenverzeichnis
        7. Literaturverzeichnis
    2. Fortschrittsanzeige nach Anmeldung
        1. Gekoppelt an Übungsaufgaben und Klicks
8. Entwicklung eines Plugins für responsivevoice.org, am besten in Kombination mit mit dem responsive.js Plugin
9. Entwicklung einer bessere Möglichkeit der Erklärungen zu Code (vgl. Code unter "III. Eingabe in Atmel Studio"). Basis könnte z.B. das Codedoc Plugin und CodePrettifier sein.
10. Verbessern des Plugins Image Map Plugin. Dieses zeigt, wenn aktiviert, beim Öffnen des Editor eine Fehlermeldung „unknown toolbar type: imagemap addBtnActionImagemap“ (Workaround vorhanden)
11. ~~Einbinden eines Online C Compilers auf die Mexle Seite: z.B.~~ https://slugelisp.ahungry.com/package/wandbox

far-off vision

1. ~~Erweiterung der Falstad Simulation mit AVR Core (z.B.~~ simavr.js~~)~~

Prio für MEXLE eLearning Plattform

1. Primär (in unsortierter Reihenfolge)
   notwendig für first Deploy
   1. Einbindung von geogebra in iFrames (ohne dass der geogebra-iFrame den Fokus schnappt), verwendet z.B. hier
   2. In spannungsfolger wurde reveal.js genutzt, um die Herleitung einer Formel nacheinander anzuzeigen. Hierfür wäre eine (alternative) Lösung gut
   3. In nichtinvertierender_verstaerker wurden Teile des Textes und der Bilder versteckt (Klick auf in der Tabelle und bei Abbildung 11).
   4. Einbindung von diagrams.net o.ä. (kompatibel dazu wäre gut, da inzwischen etliche Bilder in diagrams.net vorliegen)
   5. Bild-, Simulations- und Tabellen-Referenzen („in Abbildung x sehen Sie …“, Abbildung x als Link)

1. Sekundär (in unsortierter Reihenfolge)
   Sinnvoll für „2. Version“
   1. Lizenzangabe der eigenentwickelten Lerninhalte
   2. Verzeichnismanagement: Möglichkeit für Verzeichnisse, welche über einzelne Seiten hinaus gehen, z.B. für einen Namespace
      1. Abbildungsverzeichnis
      2. Tabellenverzeichnis
      3. Medienverzeichnis
      4. Formelverzeichnis
      5. Abkürzungsverzeichnis
      6. Aufgabenverzeichnis
      7. Literaturverzeichnis
      8. (Lizenzenverzeichnis für eingebundene Komponenten --> vermutlich in Lit.verz., Medienverz. und Abb.verz. integriert)
   3. Fortschrittsanzeige nach Anmeldung
      1. Gekoppelt an Übungsaufgaben und Klicks
   4. Darstellung der diagrams.net Diagramme als SVG, statt als PNG
   5. Anbindung an Hochschul-LDAP-Server
   6. Aufgaben
      1. Formelaufgaben
      2. Hilfen für häufige, falsche Antworten

1. Tertiär (in unsortierter Reihenfolge)
   1. Support für multilinguale Kurse:
      1. optimal wäre, dass jeder Absatz in deutsch oder <insert-Language-here> anlegbar zu machen, dann könnte die Kapitel- und Schritt-Struktur übernommen in die andere Sprache werden
      2. Alternative wäre Einbindung eines Übersetzungsprogramms
      3. Problematisch könnten Text in Bildern, Videos und Simulationen werden
   2. Kommentare für Textbereiche (Als Rückmeldung)
   3. Falstad
      1. auf dem eigenen Server (CircuitJS)
      2. Ablegen separater Dateien für die Simulation (als Text, statt bisher als Link)
      3. Ein-/Ausblenden von Menü und Slider
   4. Einbinden eines C-Compilers und ggf. eine Microcontroller-Emulation des AVR-Cores (ist

verfügbar)

5. Verwenden von Open source TTS. Damit können Lernvideos - in Kombination mit dem responsive.js - als geskriptete Präsentationen laufen. Das verbessert die Wartbarkeit, da Fehler im Video nicht zu Sprüngen in Sprache (Lautstärke, Tonlage) und Bildern führen. - Einbinden eines C-Compilers und ggf. eine Microcontroller-Emulation des AVR-Cores (ist verfügbar)
6. Verwenden von Open source TTS. Damit können Lernvideos - in Kombination mit dem responsive.js - als geskriptete Präsentationen laufen. Das verbessert die Wartbarkeit, da Fehler im Video nicht zu Sprüngen in Sprache (Lautstärke, Tonlage) und Bildern führen.