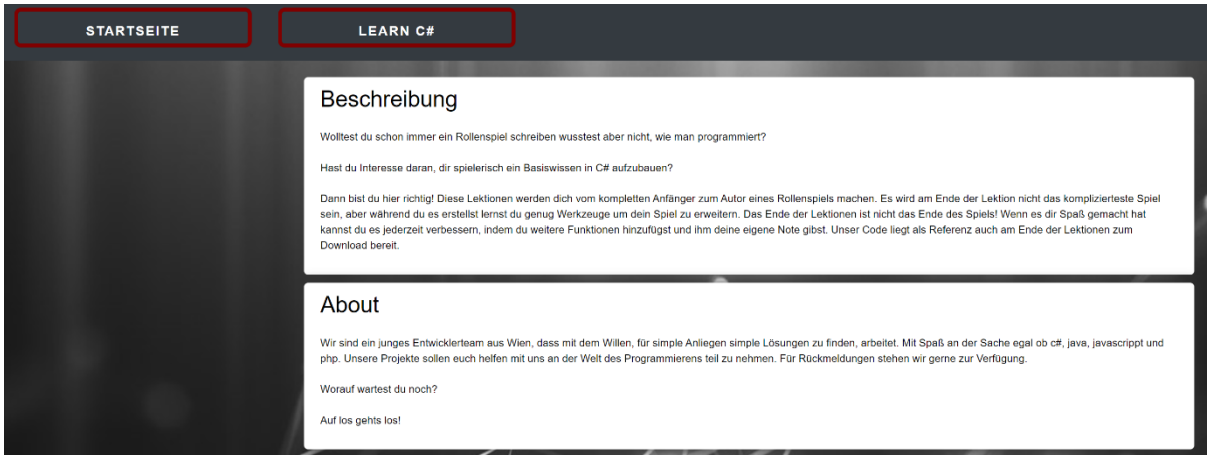
Let’s Code - Developers Guide

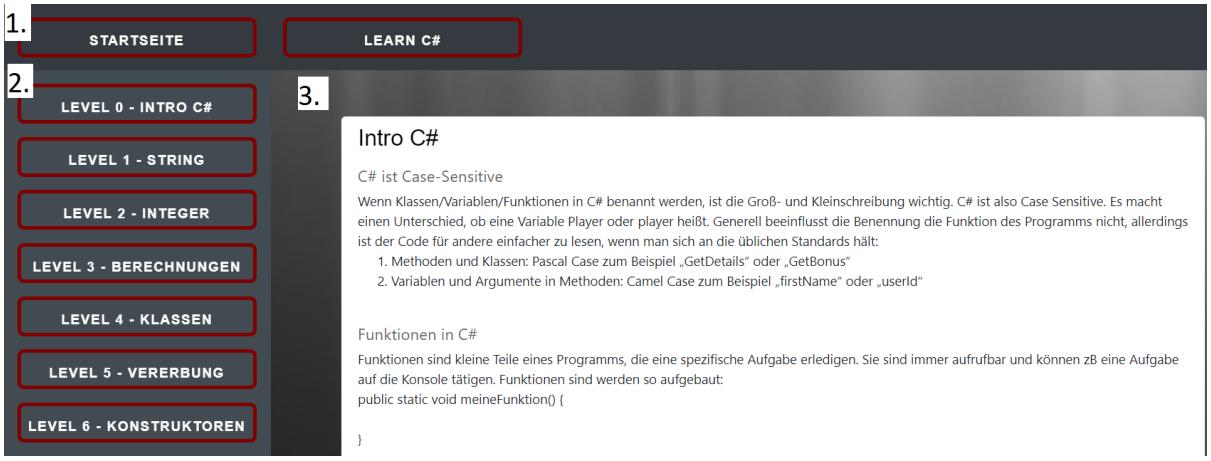
Die Seite Let’s Code besteht aus 2 großen Blöcken, der Startseite und der Subpage, auf der die einzelnen Level zu finden sind.

Auf der Startseite kann der Text in den einzelnen Div-Containern ganz normal erweitert werden. Um einen neuen Container, für Informationen zu erstellen, muss in der **“form.html”** in den divs (bild, wrap, grad, content) ein div class=”beitrag” erstellt werden. In diesem “beitrag” kann dann neue Informationen hinzugefügt werden.

Über die Buttons “Learn C#” und “Get Started” gelangt man zum Intro des C#-Kurses. Von dort aus gelangt man auch zu den einzelnen Leveln des Kurses und zurück zur Startseite.



Die Subpage kann leicht erweitert werden.



1. Ist der Header der Webseite, hier kann der User momentan zwischen der Startseite und der Subpage wechseln. In späteren Versionen könnte man hier verschiedene Sprachen einbauen.
   Diesen Teil als Code befindet sich in der **LearnC.html** Datei auf den Zeilen 50-59.
   Um eine neue Verlinkung hinzuzufügen, erstelle ein neues <li> Element in der <ul>.

2. Ist die Navigation der Webseite, hier kann der User zwischen den einzelnen Lektionen, hier als Level angegeben, wechseln. Die Levels werden durch das Drücken der Liste in die Subpage reingeladen.
   Um die Liste der Levels zu erweitern, füge in der **LearnC.html** Datei neues Listen Element zu

den Levels hinzu (Zeile 65-126). Als nächster füge innerhalb von (Zeile 132 – 286) einen neuen Aufbau für das Level hinzu. Benutze dafür die Vorlage eines anderen Levels.
Für das Aufrufen der richtigen Subpage, werden die Methoden "toggleDiv(x)" und "load_home(x)" verwendet.
Die "toggleDiv(x)" Methode, wird dazu verwendet den Animationseffekt auf der Webseite, der beim Wechsel der einzelnen Level stattfindet, ausführen zu lassen.
Die "load_home(x)" Methode, ladet die einzelnen Level in den dafür vorbereiteten div-Container.
Der Inhalt der Level befindet sich in den entsprechenden Level Webseiten im Ordner Subpages. Dafür wird der Prefix verwendet, dass jedes Level mit LevelX (X steht für eine Zahl) abgespeichert ist.
Die Methoden toggleDiv(x) und load_home(x) befinden sich in der Datei **js/load_animation.js**.

3. Hier wird der Inhalt des Levels angezeigt. Um die Levels zu erweitern, wird folgendes benötigt:
Das X in divX und InhaltX steht für die nächste freie Nummer, divX wird für die "toggle_Div(x)" Methode verwendet.
InhaltX wird für die load_home(x) Methode verwendet. Damit das richtige Level geladen wird.

Der Javascriptteil schickt den Code aus dem div mit der id "codeeditor" und evaluiert es.

Der Rückgabewert kann dann abgefragt werden.

An erster Stelle wird die Variable "editor" mit einem value befüllt. Dies ist der Text, der dem Benutzer beim Laden der Subpage angezeigt wird.

Zeilenumbrüche werden mit \n erzeugt. Ein Zeilenumbruch durch die Eingabetaste funktioniert innerhalb des Codes nicht.

Ans Ende jedes Editors wurde ein Button gehängt.

Nach betätigen wird der Inhalt des Editors zur Evaluierung geschickt. LanguageChoice steht für die Programmiersprache. Unsere Seite evaluiert momentan c#. Weitere Sprachen sind aber auch möglich.

```
C# = 1   , VB.NET = 2,    F# = 3,    Java = 4,    Python = 5,    C (gcc) =
6,    C++ (gcc) = 7  ,  Php = 8,    Pascal = 9,    Objective-C = 10,
Haskell = 11,    Ruby = 12,    Perl = 13,    Lua = 14,    Nasm = 15,
Sql Server = 16,    Javascript = 17,    Lisp = 18,    Prolog = 19,    Go =
20,    Scala = 21,    Scheme = 22,    Node.js = 23,    Python 3 = 24,
Octave = 25,    C (clang) = 26,    C++ (clang) = 27,        C++ (vc++) =
28,    C (vc) = 29,    D = 30,    R = 31,    Tcl = 32,    MySQL = 33,
PostgreSQL = 34,    Oracle = 35,    Swift = 37,    Bash = 38,    Ada = 39,
Erlang = 40,    Elixir = 41,    Ocaml = 42,    Kotlin = 43,    Brainf*** =
44,    Fortran = 45,    Rust = 46,    Clojure = 47
```

Im AJAX call wird der Code compiled. Im Erfolgsfall wird die Funktion “whatIWantToDoWithMyResponseWhenItSucceeds(data);”, im Fehlerfall wird “whatIWantToDoWithMyResponseWhenItFails(data);” aufgerufen.

Die Werte können dann mit response.Result.includes (für den output des compilers) oder editor.getValue (für den geschriebenen code) abgeprüft werden.

Wir haben editor.getValue in der variable areaContent abgespeichert.

Zum Abschluss wird abhängig von dem Ergebnis der if-Abfragen ein alert ausgegeben. Hier können aber auch mit Javascript andere Funktionen ausgeführt werden. Beispielsweise könnte man die Farbe eines Buttons von rot auf grün ändern oder der im Editor hinterlegte Code geändert werden. Der Fantasie sind keine Grenzen gesetzt.