Language Syntax

Webpage

We first have the syntax of the body of the webpage `<body>...</body>`,

```
bodyexp: anyHtmlCode> (<{exp}> anyHtmlCode)*
```

Expressions

General expressions:

```
exp: e, e', e'', ... ::=
    let <identifier> = e in e'
  | fun <identifier> -> e
  | fixfun <identifier> <identifier> -> e
  | e e'
  | if e then e' else e''
  | e;e'
  | <identifier>
  | <aexp>
  | <bexp>
  | <sexp>
  | <texp>
  | <uexp>
  | <html>
  | (e)
  | begin e end
```

Arithmetic expressions:

```
aexp:
    <exp> + <exp>
  | <exp> - <exp>
  | <exp> * <exp>
  | <exp> / <exp>
  | <exp> ^ <exp>
  | <int literal>
```

Boolean expressions:

```
bexp:
    <exp> < <exp>
  | <exp> > <exp>
  | <exp> <= <exp>
  | <exp> >= <exp>
  | <exp> = <exp>
  | <exp> <> <exp>
  | <exp> && <exp>
  | <exp> || <exp>
  | not <exp>
  | <boolean literal>
```

String expressions:

```
sexp: <exp> ++ <exp> | <fstring literal>
```

Tuple expressions:

```
texp: fst <exp> | snd <exp> | <exp>, <exp>
```

Unit expression:

`uexp: ()`

HTML:

`html: <[ anyHtmlCode ]>`

For now, only couples are allowed, and `(x1, x2, x3, x4)` is parsed as `(x1, (x2, (x3, x4)))`.

Identifiers (variable and function names)

`(_|[a-z])(_|'|[0-9]|[a-z]|[A-Z])*`

Examples:
- variable
- my_function
- _MyFunction
- myVariable

But not:
- MyFunction
- 01var

Literals

Integers

For readability for the programmer, we allow underscores in numbers.

`[0-9]([0-9]|_)*`

Examples:
- 123
- 100_000
- 1_2_____3____

Strings

Strings are delimited by quotes: `"..."`.

Format strings

Format strings are delimited by: `f"..."`. A formatter can be inserted in a format string with `%(value)`

Booleans

`true`, `false`

Type system

Types

`<tlit> : int | bool | string | unit | html`

$\alpha, \beta, \ldots$ `::=` $\alpha \to \beta$ `|` $\alpha \times \beta$

Typing rules

$$\frac{\Gamma \vdash \texttt{e} : \alpha \qquad \Gamma, \texttt{x} : \alpha \vdash \texttt{e'} : \beta}{\Gamma \vdash \texttt{let x = e in e'} : \beta} \qquad \frac{\Gamma, \texttt{x} : \alpha \vdash \texttt{e} : \beta}{\Gamma \vdash \texttt{fun x -> e} : \alpha \to \beta} \qquad \frac{\Gamma, \texttt{f} : \alpha \to \beta, \texttt{x} : \alpha \vdash \texttt{e} : \beta}{\Gamma \vdash \texttt{fixfun f x -> e} : \alpha \to \beta}$$

$$\frac{\Gamma \vdash \texttt{e} : \alpha \to \beta \qquad \Gamma \vdash \texttt{e'} : \alpha}{\Gamma \vdash \texttt{e e'} : \beta} \qquad \frac{\Gamma \vdash \texttt{e} : \texttt{bool} \qquad \Gamma \vdash \texttt{e'} : \alpha \qquad \Gamma \vdash \texttt{e''} : \alpha}{\Gamma \vdash \texttt{if e then e' else e''} : \alpha} \qquad \frac{\Gamma \vdash \texttt{e} : \texttt{unit} \qquad \Gamma \vdash \texttt{e'} : \alpha}{\Gamma \vdash \texttt{e;e'} : \alpha}$$

$$\circledast\text{: }\texttt{+, -, *, /}\text{, or }\texttt{\^{}}\ \frac{\Gamma \vdash \texttt{e} : \texttt{int} \qquad \Gamma \vdash \texttt{e'} : \texttt{int}}{\Gamma \vdash \texttt{e} \circledast \texttt{e'} : \texttt{int}} \qquad \circledast\text{: }\texttt{>, <, >=, <=, =}\text{ or }\texttt{<>}\ \frac{\Gamma \vdash \texttt{e} : \alpha \qquad \Gamma \vdash \texttt{e'} : \alpha}{\Gamma \vdash \texttt{e} \circledast \texttt{e'} : \texttt{bool}}$$

$$\circledast\text{: \&\& or }\texttt{||}\ \frac{\Gamma \vdash \texttt{e} : \texttt{bool} \qquad \Gamma \vdash \texttt{e'} : \texttt{bool}}{\Gamma \vdash \texttt{e} \circledast \texttt{e'} : \texttt{bool}} \qquad \frac{\Gamma \vdash \texttt{e} : \texttt{bool}}{\Gamma \vdash \texttt{not e} : \texttt{bool}} \qquad \Gamma(x) = \alpha\ \frac{}{\Gamma \vdash x : \alpha}$$

$$\frac{}{\Gamma \vdash \texttt{b} : \texttt{bool}} \qquad \frac{}{\Gamma \vdash \texttt{n} : \texttt{int}} \qquad \frac{}{\Gamma \vdash \texttt{<(f)string literal>} : \texttt{string}} \qquad \frac{}{\Gamma \vdash \texttt{<[html code]>} : \texttt{html}}$$

$$\frac{\Gamma \vdash \texttt{e} : \alpha \qquad \Gamma \vdash \texttt{e'} : \beta}{\Gamma \vdash \texttt{(e, e')} : \alpha \times \beta} \qquad \frac{\Gamma \vdash \texttt{e} : \alpha \times \beta}{\Gamma \vdash \texttt{fst e} : \alpha} \qquad \frac{\Gamma \vdash \texttt{e} : \alpha \times \beta}{\Gamma \vdash \texttt{snd e} : \beta} \qquad \frac{}{\Gamma \vdash \texttt{()} : \texttt{unit}}$$

Program semantics

Values

```
values: v, v', ... ::= ⟨E, <function>⟩ | n | true | false | <string literal> | (v, v')
```

```
function: fun x -> e | fixfun x -> e
```

Evaluation rules

We implement a big-step call-by-value semantics.

$$\frac{}{\texttt{E} \vdash \texttt{n} \Downarrow \texttt{n}} \qquad \frac{}{\texttt{E} \vdash \texttt{true} \Downarrow \texttt{true}} \qquad \frac{}{\texttt{E} \vdash \texttt{false} \Downarrow \texttt{false}} \qquad \frac{}{\texttt{E} \vdash \texttt{"<string>"} \Downarrow \texttt{"<string>"}}$$

$$\frac{}{\texttt{E} \vdash \texttt{fun x -> e} \Downarrow \langle \texttt{E}, \texttt{fixfun f x -> e} \rangle} \qquad \frac{}{\texttt{E} \vdash \texttt{fixfun f x -> e} \Downarrow \langle \texttt{E}, \texttt{fun x -> e} \rangle}$$

$$\frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{v} \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{v'}}{\texttt{E} \vdash \texttt{(e, e')} \Downarrow \texttt{(v, v')}} \qquad \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{n} \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{m}}{\texttt{E} \vdash \texttt{e} \circledast \texttt{e'} \Downarrow n \mathbin{\overline{\circledast}} n'} \qquad \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{n}}{\texttt{E} \vdash \texttt{-e} \Downarrow -n} \qquad \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{b} \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{b'}}{\texttt{E} \vdash \texttt{e} \circledast \texttt{e'} \Downarrow b \mathbin{\overline{\circledast}} b'}$$

$$\frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{b}}{\texttt{E} \vdash \texttt{not e} \Downarrow \neg b} \qquad \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{s} \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{s'}}{\texttt{E} \vdash \texttt{e} \mathbin{+\!\!+} \texttt{e'} \Downarrow s \mathbin{\overline{+\!\!+}} s'}$$

$$\frac{\texttt{E} \vdash \texttt{e} \Downarrow \langle \texttt{E'}, \texttt{fun x -> e\_f} \rangle \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{v} \qquad \texttt{E', x} \mapsto \texttt{v} \vdash \texttt{e\_f} \Downarrow \texttt{v'}}{\texttt{E} \vdash \texttt{e e'} \Downarrow \texttt{v'}} \qquad \frac{\texttt{E} \vdash \texttt{e'} \Downarrow \texttt{v'} \qquad \texttt{E, x} \mapsto \texttt{v'} \vdash \texttt{e'} \Downarrow \texttt{v}}{\texttt{E} \vdash \texttt{let x = e in e'} \Downarrow \texttt{v}}$$

$$\frac{\texttt{E} \vdash \texttt{e} \Downarrow \langle \texttt{E', fixfun f x -> e\_f} \rangle \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{v} \qquad \texttt{E, f} \mapsto \texttt{fixfun f x -> e\_f, x} \mapsto \texttt{v} \vdash \texttt{e\_f} \Downarrow \texttt{v'}}{\texttt{E} \vdash \texttt{e e'} \Downarrow \texttt{v'}}$$

$$\frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{v} \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{v'}}{\texttt{E} \vdash \texttt{e;e'} \Downarrow \texttt{v'}} \qquad \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{true} \qquad \texttt{E} \vdash \texttt{e'} \Downarrow \texttt{v'}}{\texttt{E} \vdash \texttt{if e then e' else e''} \Downarrow \texttt{v'}} \qquad \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{false} \qquad \texttt{E} \vdash \texttt{e''} \Downarrow \texttt{v''}}{\texttt{E} \vdash \texttt{if e then e' else e''} \Downarrow \texttt{v''}}$$

$$\frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{(v, v')}}{\texttt{E} \vdash \texttt{fst e} \Downarrow \texttt{v}} \qquad \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{(v, v')}}{\texttt{E} \vdash \texttt{snd e} \Downarrow \texttt{v'}} \qquad \texttt{E(x) = v}\ \frac{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{x}}{\texttt{E} \vdash \texttt{e} \Downarrow \texttt{v}}$$

TODO

- ☐ Don’t lex ml located in html comment
- ☐ Add syntactic sugar for multiple variables functions.
- ☐ Add t-uples
- ☐ Add pattern-matching
- ☐ Add superglobal variables (e.g. given in argument of the interpreter in a yaml format)
- ☐ Add user-defined global variables

☐ Add user-defined types

☐ Once it's done, implement basic types such as list directly within the language.

☐ Maybe revisit sequence's semantics. We may want <{"<tag>";"hey</tag>"}> to produce the html code `<tag>hey</tag>` .

☐ Allow type annotations from the user

☐ Allow importing other ml files (as modules ?)

☐ Keep line number information on parsed term for better typing error messages (?)