Client-side Modelica powered by Python or JavaScript

Rüdiger Franke, ABB, Germany – Ruediger.Franke@de.abb.com

Modelica is primarily supported by simulation environments for the treatment of equation based models and model libraries. As of today Modelica is rarely used for the exchange of engineering data, visualization or interactive computing, even though the Modelica language offers a lot of interesting features for such applications.

This paper investigates the potential of lightweight Modelica tools that run directly in scripting or web clients. Two Modelica parsers have been implemented in the popular client-side languages Python and JavaScript.

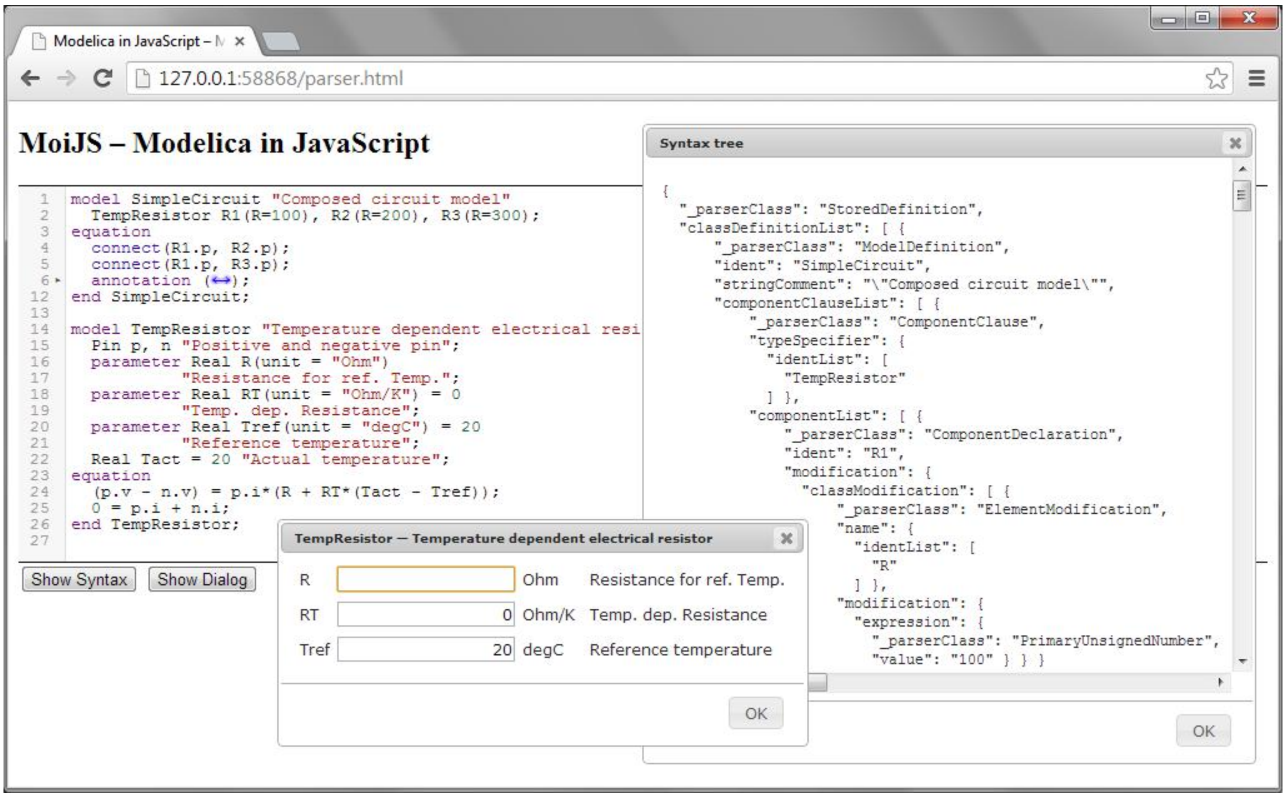
The Modelica parser in Python is extended with a backend translating algorithmic Modelica definitions to Python. This gives access to existing Python packages from scripted Modelica. It also enables the interactive debugging of algorithmic Modelica code.

The Modelica parser in JavaScript offers a generic backend interface. The paper demonstrates two applications. First a simple analysis tool for Modelica packages running from the command line is demonstrated. The true potential of JavaScript is the embedding of engineering data as Modelica code with HTML5 documents and their processing on the client side, e.g. in Web browsers. The paper shows a Modelica text editor and parameter GUI generator running in a web browser.

Client-side Modelica enables the use of the Modelica language itself as interface format in client/server architectures and for model exchange, instead of e.g. XML. Besides a significantly more compact syntax, this offers the advantage of having the semantics already standardized. The price to pay is a Modelica parser on the client side. This price turns out to be affordable in modern scripting or Web environments.

The availability of compatible JavaScript implementations by multiple vendors and fast just-in-time compilers being preinstalled on virtually any device make JavaScript attractive for more applications. Examples are HTML5 pages containing verbatim Modelica code that is evaluated on the client side, e.g. in a Web browser or in a modern mobile device.

MoiJS is extensible with new backends, exploiting the prototype based inheritance of JavaScript. MoiJS is available under the MIT license at http://omuses.github.io/moijs.


Modelica in JavaScript – M
127.0.0.1:58868/parser.html
MoiJS – Modelica in JavaScript
Show Syntax
Show Dialog
Syntax tree
OK
OK

```
model SimpleCircuit "Composed circuit model"
  TempResistor R1(R=100), R2(R=200), R3(R=300);
equation
  connect(R1.p, R2.p);
  connect(R1.p, R3.p);
  annotation (⟷);
end SimpleCircuit;

model TempResistor "Temperature dependent electrical resi
  Pin p, n "Positive and negative pin";
  parameter Real R(unit = "Ohm")
            "Resistance for ref. Temp.";
  parameter Real RT(unit = "Ohm/K") = 0
            "Temp. dep. Resistance";
  parameter Real Tref(unit = "degC") = 20
            "Reference temperature";
  Real Tact = 20 "Actual temperature";
equation
  (p.v - n.v) = p.i*(R + RT*(Tact - Tref));
  0 = p.i + n.i;
end TempResistor;
```

TempResistor — Temperature dependent electrical resistor

R		Ohm	Resistance for ref. Temp.
RT	0	Ohm/K	Temp. dep. Resistance
Tref	20	degC	Reference temperature

```
{
  "_parserClass": "StoredDefinition",
  "classDefinitionList": [ {
      "_parserClass": "ModelDefinition",
      "ident": "SimpleCircuit",
      "stringComment": "\"Composed circuit model\"",
      "componentClauseList": [ {
          "_parserClass": "ComponentClause",
          "typeSpecifier": {
            "identList": [
              "TempResistor"
            ] },
          "componentList": [ {
              "_parserClass": "ComponentDeclaration",
              "ident": "R1",
              "modification": {
                "classModification": [ {
                    "_parserClass": "ElementModification",
                    "name": {
                      "identList": [
                        "R"
                      ] },
                    "modification": {
                      "expression": {
                        "_parserClass": "PrimaryUnsignedNumber",
                        "value": "100" } } }
```