LambdaJS quick reference

Marek Materzok

December 7, 2015

1 Syntax

$$
\begin{array}{rcl}
e & ::= & x \mid l \mid e\,(e, \dots) \mid \textbf{func}\,(x, \dots)\,e \mid un\,e \mid e\;bin\;e \mid e\,;\,e \mid e\,;;\,e \mid \\
 & & \textbf{let}\,(x = e)\,e \mid \textbf{rec}\,(x = e)\,e \mid \textbf{if}\,(e)\,e\,\textbf{else}\,e \mid \\
 & & \textbf{label}\,i : e \mid \textbf{break}\,i\,e \mid \textbf{throw}\,e \mid \textbf{try}\,e\,\textbf{catch}\,(x)\,e \mid \textbf{try}\,e\,\textbf{finally}\,e \mid \\
 & & e\,[e\langle pa\rangle] \mid e\,[e\langle pa\rangle = e] \mid e\,[\textbf{delete}\,e] \mid e\,[\langle oa\rangle] \mid e\,[\langle oa\rangle = e] \mid \\
 & & \{[oa : e, \dots]\,s : pe, \dots\} \\
pe & ::= & \{\textbf{value} : e, \textbf{writable} : e, \textbf{enumerable} : e, \textbf{configurable} : e\} \mid \\
 & & \{\textbf{getter} : e, \textbf{setter} : e, \textbf{enumerable} : e, \textbf{configurable} : e\} \\
l & ::= & b \mid n \mid s \mid \textbf{undef} \mid \textbf{null} \mid \textbf{empty} \\
b & ::= & \textbf{true} \mid \textbf{false} \\
n & ::= & \textit{IEEE floating-point numbers} \\
s & ::= & \textit{UTF-16 encoded strings} \\
un & ::= & \textbf{typeof} \mid \textbf{strlen} \mid \textbf{is-primitive} \mid \textbf{is-closure} \mid \dots \\
bin & ::= & + \mid - \mid * \mid / \mid \% \mid < \mid == \mid === \mid \textbf{has-own-property} \mid \\
 & & +_s \mid <_s \mid \dots \\
pa & ::= & \textbf{value} \mid \textbf{writable} \mid \textbf{getter} \mid \textbf{setter} \mid \textbf{enumerable} \mid \textbf{configurable} \\
oa & ::= & \textbf{proto} \mid \textbf{class} \mid \textbf{extensible} \mid \textbf{code} \mid \dots
\end{array}
$$

2 Semantics

Presented in big-step style; the formalized pretty-big-step semantics in Coq can be obtained from it. Abort-handling rules (for throws and breaks) are not presented.

$$
\begin{array}{rcl}
v & ::= & l \mid (\Delta; x, \dots; e) \mid ptr \\
r & ::= & v \mid \textbf{throw}\,v \mid \textbf{break}\,i\,v \\
ptr & ::= & \textit{heap pointers} \\
o & ::= & \{[\textbf{proto} : v, \textbf{class} : s, \textbf{extensible} : b, \textbf{code} : v, \dots]\,s : p, \dots\} \\
p & ::= & \{\textbf{value} : v, \textbf{writable} : v, \textbf{enumerable} : b, \textbf{configurable} : b\} \mid \\
 & & \{\textbf{getter} : v, \textbf{setter} : v, \textbf{enumerable} : b, \textbf{configurable} : b\}
\end{array}
$$

$$
\begin{array}{rcll}
v + \textbf{empty} & = & v & \\
v + v' & = & v' & v' \neq \textbf{empty} \\
v + \textbf{throw}\,v' & = & \textbf{throw}\,v & \\
v + \textbf{break}\,i\,v' & = & \textbf{break}\,i\,(v + v') &
\end{array}
$$

$$\frac{}{\text{abort}(\textbf{throw}\,v)} \qquad \frac{}{\text{abort}(\textbf{break}\,i\,v)} \qquad \frac{oa \in \{\textbf{proto}, \textbf{extensible}\}}{oa \text{ writable}}$$

$$\frac{}{pa \text{ writable in } \{\textbf{configurable} : \textbf{true}, \dots\}} \qquad \frac{pa \in \{\textbf{value}, \textbf{writable}\}}{pa \text{ writable in } \{\textbf{writable} : \textbf{true}, \dots\}}$$

$$\frac{}{\textbf{null} \text{ is valid } \textbf{proto}} \qquad \frac{}{ptr \text{ is valid } \textbf{proto}} \qquad \frac{}{b \text{ is valid } \textbf{extensible}} \qquad \frac{}{s \text{ is valid } \textbf{class}}$$

$$\frac{}{\textbf{undef} \text{ is valid } \textbf{code}} \qquad \frac{}{(\Delta; x, \dots; e) \text{ is valid } \textbf{code}} \qquad \frac{}{v \text{ is valid } \textbf{value}} \qquad \frac{}{v \text{ is valid } \textbf{getter}}$$

$$\frac{}{v \text{ is valid } \textbf{setter}} \qquad \frac{}{b \text{ is valid } \textbf{writable}} \qquad \frac{}{b \text{ is valid } \textbf{enumerable}} \qquad \frac{}{b \text{ is valid } \textbf{configurable}}$$

$$\frac{}{\Delta; \Phi; l \Downarrow \Phi; l} \qquad \frac{}{\Delta; \Phi; x \Downarrow \Phi; \Delta(x)} \qquad \frac{}{\Delta; \Phi; \mathbf{func}\,(x_1, \ldots, x_n)\, e \Downarrow \Phi; (\Delta; x_1, \ldots, x_n; e)}$$

$$\frac{\Delta; \Phi; e \Downarrow \Phi_0; (\Delta'; x_1, \ldots, x_n; e') \qquad \forall i, \Delta; \Phi_{i-1}; e_i \Downarrow \Phi_i; v_i \qquad \Delta'(x_1, \ldots, x_n = v_1, \ldots, v_n); \Phi_n; e' \Downarrow \Phi'; r}{\Delta; \Phi; e\,(e_1, \ldots, e_n) \Downarrow \Phi'; r}$$

$$\frac{\Delta; \Phi; e \Downarrow \Phi'; v \qquad un(\Phi', v) = v'}{\Delta; \Phi; un\, e \Downarrow \Phi'; v'} \qquad \frac{\Delta; \Phi; e_1 \Downarrow \Phi'; v_1 \qquad \Delta; \Phi'; e_1 \Downarrow \Phi''; v_2 \qquad bin(\Phi'', v_1, v_2) = v}{\Delta; \Phi; e_1\, bin\, e_2 \Downarrow \Phi''; v}$$

$$\frac{\Delta; \Phi; e_1 \Downarrow \Phi'; v \qquad \Delta; \Phi'; e_2 \Downarrow \Phi''; r}{\Delta; \Phi; e_1\,;\, e_2 \Downarrow \Phi''; r} \qquad \frac{\Delta; \Phi; e_1 \Downarrow \Phi'; v \qquad \Delta; \Phi'; e_2 \Downarrow \Phi''; r \qquad r' = v + r}{\Delta; \Phi; e_1\,;;\, e_2 \Downarrow \Phi''; r'}$$

$$\frac{\Delta; \Phi; e_1 \Downarrow \Phi'; v \qquad \Delta(x = v); \Phi'; e_2 \Downarrow \Phi''; r}{\Delta; \Phi; \mathbf{let}\,(x = e_1)\, e_2 \Downarrow \Phi''; r} \qquad \frac{\Delta' = \Delta(x = (\Delta'; x_1, \ldots, x_n; e)) \qquad \Delta'; \Phi; e' \Downarrow \Phi'; r}{\Delta; \Phi; \mathbf{rec}\,(x = \mathbf{func}\,(x_1, \ldots, x_n)\, e)\, e' \Downarrow \Phi'; r}$$

$$\frac{\Delta; \Phi; e \Downarrow \Phi'; \mathbf{true} \qquad \Delta; \Phi'; e_1 \Downarrow \Phi''; r}{\Delta; \Phi; \mathbf{if}\,(e)\, e_1\, \mathbf{else}\, e_2 \Downarrow \Phi''; r} \qquad \frac{\Delta; \Phi; e \Downarrow \Phi'; \mathbf{false} \qquad \Delta; \Phi'; e_2 \Downarrow \Phi''; r}{\Delta; \Phi; \mathbf{if}\,(e)\, e_1\, \mathbf{else}\, e_2 \Downarrow \Phi''; r}$$

$$\frac{\Delta; \Phi; e \Downarrow \Phi'; \mathbf{break}\ i\ v}{\Delta; \Phi; \mathbf{label}\ i : e \Downarrow \Phi'; v} \qquad \frac{\Delta; \Phi; e \Downarrow \Phi'; r \qquad \forall v, r \neq \mathbf{break}\ i\ v}{\Delta; \Phi; \mathbf{label}\ i : e \Downarrow \Phi'; r} \qquad \frac{\Delta; \Phi; e \Downarrow \Phi'; v}{\Delta; \Phi; \mathbf{break}\ i\ e \Downarrow \Phi'; \mathbf{break}\ i\ v}$$

$$\frac{\Delta; \Phi; e \Downarrow \Phi'; v}{\Delta; \Phi; \mathbf{throw}\ e \Downarrow \Phi'; \mathbf{throw}\ v} \qquad \frac{\Delta; \Phi; e_1 \Downarrow \Phi'; r \qquad \forall v, r \neq \mathbf{throw}\ v}{\Delta; \Phi; \mathbf{try}\ e_1\ \mathbf{catch}\,(x)\, e_2 \Downarrow \Phi'; r}$$

$$\frac{\Delta; \Phi; e_1 \Downarrow \Phi'; r \qquad \Delta; \Phi'; e_2 \Downarrow \Phi''; v}{\Delta; \Phi; \mathbf{try}\ e_1\ \mathbf{finally}\ e_2 \Downarrow \Phi''; r} \qquad \frac{\Delta; \Phi; e_1 \Downarrow \Phi'; r \qquad \Delta; \Phi'; e_2 \Downarrow \Phi''; r' \qquad \mathrm{abort}(r')}{\Delta; \Phi; \mathbf{try}\ e_1\ \mathbf{finally}\ e_2 \Downarrow \Phi''; r'}$$

$$\frac{\Delta; \Phi; e_1 \Downarrow \Phi'; ptr \qquad \Delta; \Phi'; e_2 \Downarrow \Phi''; s \qquad \Phi''(ptr) = \{[\ldots]\, s : \{pa : v\}, \ldots\}}{\Delta; \Phi; e_1\, [e_2\langle pa\rangle] \Downarrow v; \Phi''}$$

$$\frac{\begin{array}{c}\Delta; \Phi; e_1 \Downarrow \Phi_1; ptr \\ \Delta; \Phi_1; e_2 \Downarrow \Phi_2; s \qquad \Delta; \Phi_2; e_3 \Downarrow \Phi_3; v \qquad \Phi_3(ptr) = o \qquad o = \{[\ldots]\ \ldots\} \qquad s \text{ not a property in } o \\ o \text{ extensible} \qquad v \text{ is valid } pa \qquad \Phi' = \Phi_3(ptr = \{[\ldots]\, s : \text{defaultprop}(pa = v), \ldots\})\end{array}}{\Delta; \Phi; e_1\, [e_2\langle pa\rangle = e_3] \Downarrow \Phi'; v}$$

$$\frac{\begin{array}{c}\Delta; \Phi; e_1 \Downarrow \Phi_1; ptr \qquad \Delta; \Phi_1; e_2 \Downarrow \Phi_2; s \qquad \Delta; \Phi_2; e_3 \Downarrow \Phi_3; v \qquad \Phi_3(ptr) = o \\ o = \{[\ldots]\, s : p, \ldots\} \qquad v \text{ is valid } pa \qquad pa \text{ writable in } p \qquad \Phi' = \Phi_3(ptr = \{[\ldots]\, s : p(pa = v), \ldots\})\end{array}}{\Delta; \Phi; e_1\, [e_2\langle pa\rangle = e_3] \Downarrow \Phi'; v}$$

$$\frac{\begin{array}{c}\Delta; \Phi; e_1 \Downarrow \Phi_1; ptr \qquad \Delta; \Phi_1; e_2 \Downarrow \Phi_2; s \\ \Phi_2(ptr) = \{[\ldots]\, s : \{\mathbf{configurable} : \mathbf{true}, \ldots\}, \ldots\} \qquad \Phi' = \Phi_2(ptr = \{[\ldots]\ \ldots\})\end{array}}{\Delta; \Phi; e_1\, [\mathbf{delete}\ e_2] \Downarrow v; \Phi'}$$

$$\frac{\Delta; \Phi; e \Downarrow ptr; \Phi' \qquad \Phi'(ptr) = \{[oa : v, \ldots]\ \ldots\}}{\Delta; \Phi; e\, [\langle oa\rangle] \Downarrow v; \Phi'}$$

$$\frac{\begin{array}{c}\Delta; \Phi; e_1 \Downarrow \Phi_1; ptr \qquad \Delta; \Phi_1; e_2 \Downarrow \Phi_2; v \qquad \Phi_2(ptr) = o \qquad o = \{[oa : v', \ldots]\ \ldots\} \\ v \text{ is valid } oa \qquad o \text{ extensible} \qquad oa \text{ writable} \qquad \Phi' = \Phi_2(ptr = \{[oa : v, \ldots]\ \ldots\})\end{array}}{\Delta; \Phi; e_1\, [\langle oa\rangle = e_2] \Downarrow \Phi'; v}$$

$$\frac{\forall i, \Delta; \Phi_{i-1}; e_i \Downarrow \Phi_i; v_i \qquad v_3 \text{ and } v_4 \text{ are bools}}{\begin{array}{c}\Delta; \Phi_0; \{\mathbf{getter} : e_1, \mathbf{setter} : e_2, \mathbf{enumerable} : e_3, \mathbf{configurable} : e_4\} \Downarrow \\ \Phi_4; \{\mathbf{getter} : v_1, \mathbf{setter} : v_2, \mathbf{enumerable} : v_3, \mathbf{configurable} : v_4\}\end{array}}$$

$$\frac{\forall i, \Delta; \Phi_{i-1}; e_i \Downarrow \Phi_i; v_i \qquad v_2,\ v_3 \text{ and } v_4 \text{ are bools}}{\begin{array}{c}\Delta; \Phi_0; \{\mathbf{value} : e_1, \mathbf{writable} : e_2, \mathbf{enumerable} : e_3, \mathbf{configurable} : e_4\} \Downarrow \\ \Phi_4; \{\mathbf{value} : v_1, \mathbf{writable} : v_2, \mathbf{enumerable} : v_3, \mathbf{configurable} : v_4\}\end{array}}$$

$$\frac{\begin{array}{c}\forall i, \Delta; \Phi_{i-1}; e_i \Downarrow \Phi_i; v_1 \qquad \forall i, \Delta; \Phi_{n+i-1}; pe_i \Downarrow \Phi_{n+i}; p_i \qquad oa_1, \ldots, oa_n \text{ distinct} \qquad s1, \ldots, s_m \text{ distinct} \\ \{\mathbf{proto}, \mathbf{class}, \mathbf{extensible}, \mathbf{code}\} \subseteq \{oa_i : i \in \{1, \ldots, n\}\} \qquad \forall i, v_i \text{ is valid } oa_i \qquad ptr \notin \Phi_{n+m}\end{array}}{\begin{array}{c}\Delta; \Phi; \{[oa_1 : e_1, \ldots, oa_n : e_n]\, s_1 : pe_1, \ldots, s_m : pe_m\} \Downarrow \\ \Phi_{n+m}(ptr = \{[oa_1 : v_1, \ldots, oa_n : v_n]\, s_1 : p_1, \ldots, s_m : p_m\}); ptr\end{array}}$$