10.3 Delayed Evaluation

(require racket/promise)  

The bindings documented in this section are provided by the racket/promise and racket libraries, but not racket/base.

A promise encapsulates an expression to be evaluated on demand via force. After a promise has been forced, every later force of the promise produces the same result.

(promise? v) → boolean?  

v : any/c

Returns #t if v is a promise, #f otherwise.

(delay body ...+)  

Creates a promise that, when forced, evaluates the bodys to produce its value. The result is then cached, so further uses of force produce the cached value immediately. This includes multiple values and exceptions.

(lazy body ...+)  

Like delay, if the last body produces a promise when forced, then this promise is forced, too, to obtain a value. In other words, this form creates a composable promise, where the computation of its body is “attached” to the computation of the following promise, and a single force iterates through the whole chain, tail-calling each step.

Note that the last body of this form must produce a single value, but the value can itself be a delay promise that returns multiple values.

The lazy form is useful for implementing lazy libraries and languages, where tail calls can be wrapped in a promise.
If \( v \) is a promise, then the promise is forced to obtain a value. If the promise has not been forced before, then the result is recorded in the promise so that future \texttt{force} on the promise produce the same value (or values). If forcing the promise raises an exception, then the exception is similarly recorded so that forcing the promise will raise the same exception every time.

If \( v \) is \texttt{forced} again before the original call to \texttt{force} returns, then the \texttt{exn:fail} exception is raised.

If \( v \) is not a promise, then it is returned as the result.

\begin{verbatim}
(promise-forced? promise)  →  boolean?
promise : promise?
\end{verbatim}

Returns \#t if \texttt{promise} has been forced.

\begin{verbatim}
(promise-running? promise)  →  boolean?
promise : promise?
\end{verbatim}

Returns \#t if \texttt{promise} is currently being forced. (Note that a promise can be either running or forced but not both.)

### 10.3.1 Additional Promise Kinds

\begin{verbatim}
(delay/name body ...+)
\end{verbatim}

Creates a “call-by-name” promise that is similar to delay-promises, except that the resulting value is not cached. This kind of promise is essentially a thunk that is wrapped in a way that \texttt{force} recognizes.

If a \texttt{delay/name} promise forces itself, no exception is raised, the promise is never considered “running” or “forced” in the sense of \texttt{promise-running?} and \texttt{promise-forced?}.

\begin{verbatim}
(delay/strict body ...+)
\end{verbatim}

Creates a “strict” promise: it is evaluated immediately, and the result is wrapped in a promise value. Note that the body can evaluate to multiple values, and forcing the resulting promise will return these values.

\begin{verbatim}
(delay/sync body ...+)
\end{verbatim}