handlebars

Using the Handlebars precompiler, you can precompile your Handlebars templates to save time on the client and reduce the required runtime size of the handlebars library.

Getting Started

First, you will need to install node and npm. On OS X:

```
$ brew install node
```

This assumes you already have Homebrew installed. If not, install it first.

Next, install the Handlebars npm package.

```
$ npm install handlebars -g
```

Using the `-g` flag installs the package globally, so it can be used in any project.

Now, you're ready to use the precompiler:

```
$ handlebars <input> -f <output>
```

The compiler will insert templates in `Handlebars.templates`. If your input file is `person.handlebars`, the compiler will insert it at `Handlebars.templates.person`. This template will be a function that may be directly executed in the same manner as templates compiled locally. I.e.

```
Handlebars.templates.person(context, options);
```

If you're working with precompiled templates, you don't need to ship the compiler with your deployed application. Instead, you can use the smaller "runtime" build.

```
<script src="/libs/handlebars.runtime.js"></script>
```

In addition to reducing the download size, eliminating client-side compilation will significantly speed up boot time, as compilation is the most expensive part of Handlebars.

Optimizations

Because you are precompiling templates, you can also apply several optimization to the compiler. The first allows you to specify a list of the known helpers to the compiler

```
handlebars <input> -f <output> -k each -k if -k unless
```

The Handlebars compiler will optimize accesses to those helpers for performance.

When all helpers are known at compile time, the `--knownOnly` option provides the smallest generated code

that also provides the fastest execution.

Usage

```
Usage: node ./bin/handlebars [template|directory]...

Options:
  -f, --output         Output File
  --map                Source Map File                                    [strin
  -a, --amd            Exports amd style (require.js)
  -c, --commonjs       Exports CommonJS style, path to Handlebars module
  -h, --handlebarPath  Path to handlebar.js (only valid for amd-style)
  -k, --known          Known helpers
  -o, --knownOnly      Known helpers only
  -m, --min            Minimize output
  -n, --namespace      Template namespace                                 [default:
  -s, --simple         Output template function only.
  -N, --name           Name of passed string templates. Optional if running in a simple mode. Re
                       on multiple templates.
  -i, --string         Generates a template from the passed CLI argument.
                       "-" is treated as a special value and causes stdin to be read for the tem
  -r, --root           Template root. Base value that will be stripped from template names.
  -p, --partial        Compiling a partial template
  -d, --data           Include data when compiling
  -e, --extension      Template extension.
  -b, --bom            Removes the BOM (Byte Order Mark) from the beginning of the templates.
  -v, --version        Prints the current compiler version
  --help               Outputs this message
```

If using the precompiler's normal mode, the resulting templates will be stored to the `Handlebars.templates` object using the relative template name sans the extension. These templates may be executed in the same manner as templates.

If using the simple mode the precompiler will generate a single javascript method. To execute this method it must be passed to the `Handlebars.template` method and the resulting object may be used as normal.