1344 lines (1240 sloc) | 44.8 KB ...

```
'use strict';

var CleanCSS = require('clean-css');
var decode = require('he').decode;
var HTMLParser = require('./htmlparser').HTMLParser;
var RelateUrl = require('relateurl');
var TokenChain = require('./tokenchain');
var UglifyJS = require('uglify-js');
var utils = require('./utils');

function trimWhitespace(str) {
  return str && str.replace(/^[ \n\r\t\f]+/, '').replace(/[ \n\r\t\f]+$/, '');
}

function collapseWhitespaceAll(str) {
  // Non-breaking space is specifically handled inside the replacer function here:
  return str && str.replace(/[ \n\r\t\f\xA0]+/g, function(spaces) {
    return spaces === '\t' ? '\t' : spaces.replace(/(^|\xA0+)[^\xA0]+/g, '$1 ');
  });
}

function collapseWhitespace(str, options, trimLeft, trimRight, collapseAll) {
  var lineBreakBefore = '', lineBreakAfter = '';

  if (options.preserveLineBreaks) {
    str = str.replace(/^[ \n\r\t\f]*?[\n\r][ \n\r\t\f]*/, function() {
      lineBreakBefore = '\n';
      return '';
    }).replace(/[ \n\r\t\f]*?[\n\r][ \n\r\t\f]*$/, function() {
```

```
      lineBreakAfter = '\n';
      return '';
    });
  }

  if (trimLeft) {
    // Non-breaking space is specifically handled inside the replacer function here:
    str = str.replace(/^[ \n\r\t\f\xA0]+/, function(spaces) {
      var conservative = !lineBreakBefore && options.conservativeCollapse;
      if (conservative && spaces === '\t') {
        return '\t';
      }
      return spaces.replace(/^[^\xA0]+/, '').replace(/(\xA0+)[^\xA0]+/g, '$1 ') || (conservative ?
    });
  }

  if (trimRight) {
    // Non-breaking space is specifically handled inside the replacer function here:
    str = str.replace(/[ \n\r\t\f\xA0]+$/, function(spaces) {
      var conservative = !lineBreakAfter && options.conservativeCollapse;
      if (conservative && spaces === '\t') {
        return '\t';
      }
      return spaces.replace(/[^\xA0]+(\xA0+)/g, ' $1').replace(/[^\xA0]+$/, '') || (conservative ?
    });
  }

  if (collapseAll) {
    // strip non space whitespace then compress spaces to one
    str = collapseWhitespaceAll(str);
  }

  return lineBreakBefore + str + lineBreakAfter;
}

var createMapFromString = utils.createMapFromString;
// non-empty tags that will maintain whitespace around them
var inlineTags = createMapFromString('a,abbr,acronym,b,bdi,bdo,big,button,cite,code,del,dfn,em,fon
// non-empty tags that will maintain whitespace within them
var inlineTextTags = createMapFromString('a,abbr,acronym,b,big,del,em,font,i,ins,kbd,mark,nobr,rp,
// self-closing tags that will maintain whitespace around them
var selfClosingInlineTags = createMapFromString('comment,img,input,wbr');

function collapseWhitespaceSmart(str, prevTag, nextTag, options) {
  var trimLeft = prevTag && !selfClosingInlineTags(prevTag);
  if (trimLeft && !options.collapseInlineTagWhitespace) {
    trimLeft = prevTag.charAt(0) === '/' ? !inlineTags(prevTag.slice(1)) : !inlineTextTags(prevTag
  }
  var trimRight = nextTag && !selfClosingInlineTags(nextTag);
```

```
  if (trimRight && !options.collapseInlineTagWhitespace) {
    trimRight = nextTag.charAt(0) === '/' ? !inlineTextTags(nextTag.slice(1)) : !inlineTags(nextTa
  }
  return collapseWhitespace(str, options, trimLeft, trimRight, prevTag && nextTag);
}

function isConditionalComment(text) {
  return /^\[if\s[^\]]+]|\[endif]$/.test(text);
}

function isIgnoredComment(text, options) {
  for (var i = 0, len = options.ignoreCustomComments.length; i < len; i++) {
    if (options.ignoreCustomComments[i].test(text)) {
      return true;
    }
  }
  return false;
}

function isEventAttribute(attrName, options) {
  var patterns = options.customEventAttributes;
  if (patterns) {
    for (var i = patterns.length; i--;) {
      if (patterns[i].test(attrName)) {
        return true;
      }
    }
    return false;
  }
  return /^on[a-z]{3,}$/.test(attrName);
}

function canRemoveAttributeQuotes(value) {
  // https://mathiasbynens.be/notes/unquoted-attribute-values
  return /^[^ \t\n\f\r"'`=<>]+$/.test(value);
}

function attributesInclude(attributes, attribute) {
  for (var i = attributes.length; i--;) {
    if (attributes[i].name.toLowerCase() === attribute) {
      return true;
    }
  }
  return false;
}

function isAttributeRedundant(tag, attrName, attrValue, attrs) {
  attrValue = attrValue ? trimWhitespace(attrValue.toLowerCase()) : '';

```

```
  return (
    tag === 'script' &&
    attrName === 'language' &&
    attrValue === 'javascript' ||

    tag === 'form' &&
    attrName === 'method' &&
    attrValue === 'get' ||

    tag === 'input' &&
    attrName === 'type' &&
    attrValue === 'text' ||

    tag === 'script' &&
    attrName === 'charset' &&
    !attributesInclude(attrs, 'src') ||

    tag === 'a' &&
    attrName === 'name' &&
    attributesInclude(attrs, 'id') ||

    tag === 'area' &&
    attrName === 'shape' &&
    attrValue === 'rect'
  );
}

// https://mathiasbynens.be/demo/javascript-mime-type
// https://developer.mozilla.org/en/docs/Web/HTML/Element/script#attr-type
var executableScriptsMimetypes = utils.createMap([
  'text/javascript',
  'text/ecmascript',
  'text/jscript',
  'application/javascript',
  'application/x-javascript',
  'application/ecmascript'
]);

function isScriptTypeAttribute(attrValue) {
  attrValue = trimWhitespace(attrValue.split(/;/, 2)[0]).toLowerCase();
  return attrValue === '' || executableScriptsMimetypes(attrValue);
}

function isExecutableScript(tag, attrs) {
  if (tag !== 'script') {
    return false;
  }
  for (var i = 0, len = attrs.length; i < len; i++) {
    var attrName = attrs[i].name.toLowerCase();
```

```
    if (attrName === 'type') {
      return isScriptTypeAttribute(attrs[i].value);
    }
  }
  return true;
}

function isStyleLinkTypeAttribute(attrValue) {
  attrValue = trimWhitespace(attrValue).toLowerCase();
  return attrValue === '' || attrValue === 'text/css';
}

function isStyleSheet(tag, attrs) {
  if (tag !== 'style') {
    return false;
  }
  for (var i = 0, len = attrs.length; i < len; i++) {
    var attrName = attrs[i].name.toLowerCase();
    if (attrName === 'type') {
      return isStyleLinkTypeAttribute(attrs[i].value);
    }
  }
  return true;
}

var isSimpleBoolean = createMapFromString('allowfullscreen,async,autofocus,autoplay,checked,compac
var isBooleanValue = createMapFromString('true,false');

function isBooleanAttribute(attrName, attrValue) {
  return isSimpleBoolean(attrName) || attrName === 'draggable' && !isBooleanValue(attrValue);
}

function isUriTypeAttribute(attrName, tag) {
  return (
    /^(?:a|area|link|base)$/.test(tag) && attrName === 'href' ||
    tag === 'img' && /^(?:src|longdesc|usemap)$/.test(attrName) ||
    tag === 'object' && /^(?:classid|codebase|data|usemap)$/.test(attrName) ||
    tag === 'q' && attrName === 'cite' ||
    tag === 'blockquote' && attrName === 'cite' ||
    (tag === 'ins' || tag === 'del') && attrName === 'cite' ||
    tag === 'form' && attrName === 'action' ||
    tag === 'input' && (attrName === 'src' || attrName === 'usemap') ||
    tag === 'head' && attrName === 'profile' ||
    tag === 'script' && (attrName === 'src' || attrName === 'for')
  );
}

function isNumberTypeAttribute(attrName, tag) {
  return (
```

```
    /^(?:a|area|object|button)$/.test(tag) && attrName === 'tabindex' ||
    tag === 'input' && (attrName === 'maxlength' || attrName === 'tabindex') ||
    tag === 'select' && (attrName === 'size' || attrName === 'tabindex') ||
    tag === 'textarea' && /^(?:rows|cols|tabindex)$/.test(attrName) ||
    tag === 'colgroup' && attrName === 'span' ||
    tag === 'col' && attrName === 'span' ||
    (tag === 'th' || tag === 'td') && (attrName === 'rowspan' || attrName === 'colspan')
  );
}

function isLinkType(tag, attrs, value) {
  if (tag !== 'link') {
    return false;
  }
  for (var i = 0, len = attrs.length; i < len; i++) {
    if (attrs[i].name === 'rel' && attrs[i].value === value) {
      return true;
    }
  }
}

function isMediaQuery(tag, attrs, attrName) {
  return attrName === 'media' && (isLinkType(tag, attrs, 'stylesheet') || isStyleSheet(tag, attrs));
}

var srcsetTags = createMapFromString('img,source');

function isSrcset(attrName, tag) {
  return attrName === 'srcset' && srcsetTags(tag);
}

function cleanAttributeValue(tag, attrName, attrValue, options, attrs) {
  if (isEventAttribute(attrName, options)) {
    attrValue = trimWhitespace(attrValue).replace(/^javascript:\s*/i, '');
    return options.minifyJS(attrValue, true);
  }
  else if (attrName === 'class') {
    attrValue = trimWhitespace(attrValue);
    if (options.sortClassName) {
      attrValue = options.sortClassName(attrValue);
    }
    else {
      attrValue = collapseWhitespaceAll(attrValue);
    }
    return attrValue;
  }
  else if (isUriTypeAttribute(attrName, tag)) {
    attrValue = trimWhitespace(attrValue);
    return isLinkType(tag, attrs, 'canonical') ? attrValue : options.minifyURLs(attrValue);
```

```
  }
  else if (isNumberTypeAttribute(attrName, tag)) {
    return trimWhitespace(attrValue);
  }
  else if (attrName === 'style') {
    attrValue = trimWhitespace(attrValue);
    if (attrValue) {
      if (/;$/.test(attrValue) && !/&#?[0-9a-zA-Z]+;$/.test(attrValue)) {
        attrValue = attrValue.replace(/\s*;$/, ';');
      }
      attrValue = options.minifyCSS(attrValue, 'inline');
    }
    return attrValue;
  }
  else if (isSrcset(attrName, tag)) {
    // https://html.spec.whatwg.org/multipage/embedded-content.html#attr-img-srcset
    attrValue = trimWhitespace(attrValue).split(/\s+,\s*|\s*,\s+/).map(function(candidate) {
      var url = candidate;
      var descriptor = '';
      var match = candidate.match(/\s+([1-9][0-9]*w|[0-9]+(?:\.[0-9]+)?x)$/);
      if (match) {
        url = url.slice(0, -match[0].length);
        var num = +match[1].slice(0, -1);
        var suffix = match[1].slice(-1);
        if (num !== 1 || suffix !== 'x') {
          descriptor = ' ' + num + suffix;
        }
      }
      return options.minifyURLs(url) + descriptor;
    }).join(', ');
  }
  else if (isMetaViewport(tag, attrs) && attrName === 'content') {
    attrValue = attrValue.replace(/\s+/g, '').replace(/[0-9]+\.[0-9]+/g, function(numString) {
      // "0.90000" -> "0.9"
      // "1.0" -> "1"
      // "1.0001" -> "1.0001" (unchanged)
      return (+numString).toString();
    });
  }
  else if (isContentSecurityPolicy(tag, attrs) && attrName.toLowerCase() === 'content') {
    return collapseWhitespaceAll(attrValue);
  }
  else if (options.customAttrCollapse && options.customAttrCollapse.test(attrName)) {
    attrValue = attrValue.replace(/\n+|\r+|\s{2,}/g, '');
  }
  else if (tag === 'script' && attrName === 'type') {
    attrValue = trimWhitespace(attrValue.replace(/\s*;\s*/g, ';'));
  }
  else if (isMediaQuery(tag, attrs, attrName)) {
```

```
    attrValue = trimWhitespace(attrValue);
    return options.minifyCSS(attrValue, 'media');
  }
  return attrValue;
}

function isMetaViewport(tag, attrs) {
  if (tag !== 'meta') {
    return false;
  }
  for (var i = 0, len = attrs.length; i < len; i++) {
    if (attrs[i].name === 'name' && attrs[i].value === 'viewport') {
      return true;
    }
  }
}

function isContentSecurityPolicy(tag, attrs) {
  if (tag !== 'meta') {
    return false;
  }
  for (var i = 0, len = attrs.length; i < len; i++) {
    if (attrs[i].name.toLowerCase() === 'http-equiv' && attrs[i].value.toLowerCase() === 'content-
      return true;
    }
  }
}

function ignoreCSS(id) {
  return '/* clean-css ignore:start */' + id + '/* clean-css ignore:end */';
}

// Wrap CSS declarations for CleanCSS > 3.x
// See https://github.com/jakubpawlowicz/clean-css/issues/418
function wrapCSS(text, type) {
  switch (type) {
    case 'inline':
      return '*{' + text + '}';
    case 'media':
      return '@media ' + text + '{a{top:0}}';
    default:
      return text;
  }
}

function unwrapCSS(text, type) {
  var matches;
  switch (type) {
    case 'inline':
```

```
      matches = text.match(/^\*\{([\s\S]*)\}$/);
      break;
    case 'media':
      matches = text.match(/^@media ([\s\S]*?)\s*{[\s\S]*}$/);
      break;
  }
  return matches ? matches[1] : text;
}

function cleanConditionalComment(comment, options) {
  return options.processConditionalComments ? comment.replace(/^(\[if\s[^\]]+]>)([\s\S]*?)(<![end
    return prefix + minify(text, options, true) + suffix;
  }) : comment;
}

function processScript(text, options, currentAttrs) {
  for (var i = 0, len = currentAttrs.length; i < len; i++) {
    if (currentAttrs[i].name.toLowerCase() === 'type' &&
        options.processScripts.indexOf(currentAttrs[i].value) > -1) {
      return minify(text, options);
    }
  }
  return text;
}

// Tag omission rules from https://html.spec.whatwg.org/multipage/syntax.html#optional-tags
// with the following deviations:
// - retain <body> if followed by <noscript>
// - </rb>, </rt>, </rtc>, </rp> & </tfoot> follow https://www.w3.org/TR/html5/syntax.html#optiona
// - retain all tags which are adjacent to non-standard HTML tags
var optionalStartTags = createMapFromString('html,head,body,colgroup,tbody');
var optionalEndTags = createMapFromString('html,head,body,li,dt,dd,p,rb,rt,rtc,rp,optgroup,option,
var headerTags = createMapFromString('meta,link,script,style,template,noscript');
var descriptionTags = createMapFromString('dt,dd');
var pBlockTags = createMapFromString('address,article,aside,blockquote,details,div,dl,fieldset,fig
var pInlineTags = createMapFromString('a,audio,del,ins,map,noscript,video');
var rubyTags = createMapFromString('rb,rt,rtc,rp');
var rtcTag = createMapFromString('rb,rtc,rp');
var optionTag = createMapFromString('option,optgroup');
var tableContentTags = createMapFromString('tbody,tfoot');
var tableSectionTags = createMapFromString('thead,tbody,tfoot');
var cellTags = createMapFromString('td,th');
var topLevelTags = createMapFromString('html,head,body');
var compactTags = createMapFromString('html,body');
var looseTags = createMapFromString('head,colgroup,caption');
var trailingTags = createMapFromString('dt,thead');
var htmlTags = createMapFromString('a,abbr,acronym,address,applet,area,article,aside,audio,b,base,

function canRemoveParentTag(optionalStartTag, tag) {
```

```
  switch (optionalStartTag) {
    case 'html':
    case 'head':
      return true;
    case 'body':
      return !headerTags(tag);
    case 'colgroup':
      return tag === 'col';
    case 'tbody':
      return tag === 'tr';
  }
  return false;
}

function isStartTagMandatory(optionalEndTag, tag) {
  switch (tag) {
    case 'colgroup':
      return optionalEndTag === 'colgroup';
    case 'tbody':
      return tableSectionTags(optionalEndTag);
  }
  return false;
}

function canRemovePrecedingTag(optionalEndTag, tag) {
  switch (optionalEndTag) {
    case 'html':
    case 'head':
    case 'body':
    case 'colgroup':
    case 'caption':
      return true;
    case 'li':
    case 'optgroup':
    case 'tr':
      return tag === optionalEndTag;
    case 'dt':
    case 'dd':
      return descriptionTags(tag);
    case 'p':
      return pBlockTags(tag);
    case 'rb':
    case 'rt':
    case 'rp':
      return rubyTags(tag);
    case 'rtc':
      return rtcTag(tag);
    case 'option':
      return optionTag(tag);
```

```
    case 'thead':
    case 'tbody':
      return tableContentTags(tag);
    case 'tfoot':
      return tag === 'tbody';
    case 'td':
    case 'th':
      return cellTags(tag);
  }
  return false;
}

var reEmptyAttribute = new RegExp(
  '^(?:class|id|style|title|lang|dir|on(?:focus|blur|change|click|dblclick|mouse(' +
    '?:down|up|over|move|out)|key(?:press|down|up)))$');

function canDeleteEmptyAttribute(tag, attrName, attrValue, options) {
  var isValueEmpty = !attrValue || /^\s*$/.test(attrValue);
  if (!isValueEmpty) {
    return false;
  }
  if (typeof options.removeEmptyAttributes === 'function') {
    return options.removeEmptyAttributes(attrName, tag);
  }
  return tag === 'input' && attrName === 'value' || reEmptyAttribute.test(attrName);
}

function hasAttrName(name, attrs) {
  for (var i = attrs.length - 1; i >= 0; i--) {
    if (attrs[i].name === name) {
      return true;
    }
  }
  return false;
}

function canRemoveElement(tag, attrs) {
  switch (tag) {
    case 'textarea':
      return false;
    case 'audio':
    case 'script':
    case 'video':
      if (hasAttrName('src', attrs)) {
        return false;
      }
      break;
    case 'iframe':
      if (hasAttrName('src', attrs) || hasAttrName('srcdoc', attrs)) {
```

```
        return false;
      }
      break;
    case 'object':
      if (hasAttrName('data', attrs)) {
        return false;
      }
      break;
    case 'applet':
      if (hasAttrName('code', attrs)) {
        return false;
      }
      break;
  }
  return true;
}

function canCollapseWhitespace(tag) {
  return !/^(?:script|style|pre|textarea)$/.test(tag);
}

function canTrimWhitespace(tag) {
  return !/^(?:pre|textarea)$/.test(tag);
}

function normalizeAttr(attr, attrs, tag, options) {
  var attrName = options.name(attr.name),
      attrValue = attr.value;

  if (options.decodeEntities && attrValue) {
    attrValue = decode(attrValue, { isAttributeValue: true });
  }

  if (options.removeRedundantAttributes &&
    isAttributeRedundant(tag, attrName, attrValue, attrs) ||
    options.removeScriptTypeAttributes && tag === 'script' &&
    attrName === 'type' && isScriptTypeAttribute(attrValue) ||
    options.removeStyleLinkTypeAttributes && (tag === 'style' || tag === 'link') &&
    attrName === 'type' && isStyleLinkTypeAttribute(attrValue)) {
    return;
  }

  if (attrValue) {
    attrValue = cleanAttributeValue(tag, attrName, attrValue, options, attrs);
  }

  if (options.removeEmptyAttributes &&
      canDeleteEmptyAttribute(tag, attrName, attrValue, options)) {
    return;
```

```
569     }
570
571     if (options.decodeEntities && attrValue) {
572       attrValue = attrValue.replace(/&(#?[0-9a-zA-Z]+;)/g, '&amp;$1');
573     }
574
575     return {
576       attr: attr,
577       name: attrName,
578       value: attrValue
579     };
580   }
581
582   function buildAttr(normalized, hasUnarySlash, options, isLast, uidAttr) {
583     var attrName = normalized.name,
584         attrValue = normalized.value,
585         attr = normalized.attr,
586         attrQuote = attr.quote,
587         attrFragment,
588         emittedAttrValue;
589
590     if (typeof attrValue !== 'undefined' && (!options.removeAttributeQuotes ||
591         ~attrValue.indexOf(uidAttr) || !canRemoveAttributeQuotes(attrValue))) {
592       if (!options.preventAttributesEscaping) {
593         if (typeof options.quoteCharacter === 'undefined') {
594           var apos = (attrValue.match(/'/g) || []).length;
595           var quot = (attrValue.match(/"/g) || []).length;
596           attrQuote = apos < quot ? '\'' : '"';
597         }
598         else {
599           attrQuote = options.quoteCharacter === '\'' ? '\'' : '"';
600         }
601         if (attrQuote === '"') {
602           attrValue = attrValue.replace(/"/g, '&#34;');
603         }
604         else {
605           attrValue = attrValue.replace(/'/g, '&#39;');
606         }
607       }
608       emittedAttrValue = attrQuote + attrValue + attrQuote;
609       if (!isLast && !options.removeTagWhitespace) {
610         emittedAttrValue += ' ';
611       }
612     }
613     // make sure trailing slash is not interpreted as HTML self-closing tag
614     else if (isLast && !hasUnarySlash && !/\/$/.test(attrValue)) {
615       emittedAttrValue = attrValue;
616     }
617     else {
```

```
    emittedAttrValue = attrValue + ' ';
  }

  if (typeof attrValue === 'undefined' || options.collapseBooleanAttributes &&
      isBooleanAttribute(attrName.toLowerCase(), attrValue.toLowerCase())) {
    attrFragment = attrName;
    if (!isLast) {
      attrFragment += ' ';
    }
  }
  else {
    attrFragment = attrName + attr.customAssign + emittedAttrValue;
  }

  return attr.customOpen + attrFragment + attr.customClose;
}

function identity(value) {
  return value;
}

function processOptions(values) {
  var options = {
    name: function(name) {
      return name.toLowerCase();
    },
    canCollapseWhitespace: canCollapseWhitespace,
    canTrimWhitespace: canTrimWhitespace,
    html5: true,
    ignoreCustomComments: [/^!/],
    ignoreCustomFragments: [
      /<%[\s\S]*?%>/,
      /<\?[\s\S]*?\?>/
    ],
    includeAutoGeneratedTags: true,
    log: identity,
    minifyCSS: identity,
    minifyJS: identity,
    minifyURLs: identity
  };
  Object.keys(values).forEach(function(key) {
    var value = values[key];
    if (key === 'caseSensitive') {
      if (value) {
        options.name = identity;
      }
    }
    else if (key === 'log') {
      if (typeof value === 'function') {
```

```
667         options.log = value;
668       }
669     }
670     else if (key === 'minifyCSS' && typeof value !== 'function') {
671       if (!value) {
672         return;
673       }
674       if (typeof value !== 'object') {
675         value = {};
676       }
677       options.minifyCSS = function(text, type) {
678         text = text.replace(/(url\s*\(\s*)("|'|)(.*?)\2(\s*\))/ig, function(match, prefix, quote,
679           return prefix + quote + options.minifyURLs(url) + quote + suffix;
680         });
681         var cleanCssOutput = new CleanCSS(value).minify(wrapCSS(text, type));
682         if (cleanCssOutput.errors.length > 0) {
683           cleanCssOutput.errors.forEach(options.log);
684           return text;
685         }
686         return unwrapCSS(cleanCssOutput.styles, type);
687       };
688     }
689     else if (key === 'minifyJS' && typeof value !== 'function') {
690       if (!value) {
691         return;
692       }
693       if (typeof value !== 'object') {
694         value = {};
695       }
696       (value.parse || (value.parse = {})).bare_returns = false;
697       options.minifyJS = function(text, inline) {
698         var start = text.match(/^\s*<!--.*/);
699         var code = start ? text.slice(start[0].length).replace(/\n\s*-->\s*$/, '') : text;
700         value.parse.bare_returns = inline;
701         var result = UglifyJS.minify(code, value);
702         if (result.error) {
703           options.log(result.error);
704           return text;
705         }
706         return result.code.replace(/;$/, '');
707       };
708     }
709     else if (key === 'minifyURLs' && typeof value !== 'function') {
710       if (!value) {
711         return;
712       }
713       if (typeof value === 'string') {
714         value = { site: value };
715       }
```

```
      else if (typeof value !== 'object') {
        value = {};
      }
      options.minifyURLs = function(text) {
        try {
          return RelateUrl.relate(text, value);
        }
        catch (err) {
          options.log(err);
          return text;
        }
      };
    }
    else {
      options[key] = value;
    }
  });
  return options;
}

function uniqueId(value) {
  var id;
  do {
    id = Math.random().toString(36).replace(/^0\.[0-9]*/, '');
  } while (~value.indexOf(id));
  return id;
}

var specialContentTags = createMapFromString('script,style');

function createSortFns(value, options, uidIgnore, uidAttr) {
  var attrChains = options.sortAttributes && Object.create(null);
  var classChain = options.sortClassName && new TokenChain();

  function attrNames(attrs) {
    return attrs.map(function(attr) {
      return options.name(attr.name);
    });
  }

  function shouldSkipUID(token, uid) {
    return !uid || token.indexOf(uid) === -1;
  }

  function shouldSkipUIDs(token) {
    return shouldSkipUID(token, uidIgnore) && shouldSkipUID(token, uidAttr);
  }

  function scan(input) {
```

```
765     var currentTag, currentType;
766     new HTMLParser(input, {
767       start: function(tag, attrs) {
768         if (attrChains) {
769           if (!attrChains[tag]) {
770             attrChains[tag] = new TokenChain();
771           }
772           attrChains[tag].add(attrNames(attrs).filter(shouldSkipUIDs));
773         }
774         for (var i = 0, len = attrs.length; i < len; i++) {
775           var attr = attrs[i];
776           if (classChain && attr.value && options.name(attr.name) === 'class') {
777             classChain.add(trimWhitespace(attr.value).split(/[ \t\n\f\r]+/).filter(shouldSkipUIDs)
778           }
779           else if (options.processScripts && attr.name.toLowerCase() === 'type') {
780             currentTag = tag;
781             currentType = attr.value;
782           }
783         }
784       },
785       end: function() {
786         currentTag = '';
787       },
788       chars: function(text) {
789         if (options.processScripts && specialContentTags(currentTag) &&
790             options.processScripts.indexOf(currentType) > -1) {
791           scan(text);
792         }
793       }
794     });
795   }
796
797   var log = options.log;
798   options.log = identity;
799   options.sortAttributes = false;
800   options.sortClassName = false;
801   scan(minify(value, options));
802   options.log = log;
803   if (attrChains) {
804     var attrSorters = Object.create(null);
805     for (var tag in attrChains) {
806       attrSorters[tag] = attrChains[tag].createSorter();
807     }
808     options.sortAttributes = function(tag, attrs) {
809       var sorter = attrSorters[tag];
810       if (sorter) {
811         var attrMap = Object.create(null);
812         var names = attrNames(attrs);
813         names.forEach(function(name, index) {
```

```
          (attrMap[name] || (attrMap[name] = [])).push(attrs[index]);
        });
        sorter.sort(names).forEach(function(name, index) {
          attrs[index] = attrMap[name].shift();
        });
      }
    };
  }
  if (classChain) {
    var sorter = classChain.createSorter();
    options.sortClassName = function(value) {
      return sorter.sort(value.split(/[ \n\f\r]+/)).join(' ');
    };
  }
}

function minify(value, options, partialMarkup) {
  if (options.collapseWhitespace) {
    value = collapseWhitespace(value, options, true, true);
  }

  var buffer = [],
      charsPrevTag,
      currentChars = '',
      hasChars,
      currentTag = '',
      currentAttrs = [],
      stackNoTrimWhitespace = [],
      stackNoCollapseWhitespace = [],
      optionalStartTag = '',
      optionalEndTag = '',
      ignoredMarkupChunks = [],
      ignoredCustomMarkupChunks = [],
      uidIgnore,
      uidAttr,
      uidPattern;

  // temporarily replace ignored chunks with comments,
  // so that we don't have to worry what's there.
  // for all we care there might be
  // completely-horribly-broken-alien-non-html-emoj-cthulhu-filled content
  value = value.replace(/<!-- htmlmin:ignore -->([\s\S]*?)<!-- htmlmin:ignore -->/g, function(matc
    if (!uidIgnore) {
      uidIgnore = uniqueId(value);
      var pattern = new RegExp('^' + uidIgnore + '([0-9]+)$');
      if (options.ignoreCustomComments) {
        options.ignoreCustomComments = options.ignoreCustomComments.slice();
      }
      else {
```

```
        options.ignoreCustomComments = [];
      }
      options.ignoreCustomComments.push(pattern);
    }
    var token = '<!--' + uidIgnore + ignoredMarkupChunks.length + '-->';
    ignoredMarkupChunks.push(group1);
    return token;
  });

  var customFragments = options.ignoreCustomFragments.map(function(re) {
    return re.source;
  });
  if (customFragments.length) {
    var reCustomIgnore = new RegExp('\\s*(?:' + customFragments.join('|') + ')+\\s*', 'g');
    // temporarily replace custom ignored fragments with unique attributes
    value = value.replace(reCustomIgnore, function(match) {
      if (!uidAttr) {
        uidAttr = uniqueId(value);
        uidPattern = new RegExp('(\\s*)' + uidAttr + '([0-9]+)' + uidAttr + '(\\s*)', 'g');
        if (options.minifyCSS) {
          options.minifyCSS = (function(fn) {
            return function(text, type) {
              text = text.replace(uidPattern, function(match, prefix, index) {
                var chunks = ignoredCustomMarkupChunks[+index];
                return chunks[1] + uidAttr + index + uidAttr + chunks[2];
              });
              var ids = [];
              new CleanCSS().minify(wrapCSS(text, type)).warnings.forEach(function(warning) {
                var match = uidPattern.exec(warning);
                if (match) {
                  var id = uidAttr + match[2] + uidAttr;
                  text = text.replace(id, ignoreCSS(id));
                  ids.push(id);
                }
              });
              text = fn(text, type);
              ids.forEach(function(id) {
                text = text.replace(ignoreCSS(id), id);
              });
              return text;
            };
          })(options.minifyCSS);
        }
        if (options.minifyJS) {
          options.minifyJS = (function(fn) {
            return function(text, type) {
              return fn(text.replace(uidPattern, function(match, prefix, index) {
                var chunks = ignoredCustomMarkupChunks[+index];
                return chunks[1] + uidAttr + index + uidAttr + chunks[2];
```

```
            }), type);
          };
        })(options.minifyJS);
      }
    }
    var token = uidAttr + ignoredCustomMarkupChunks.length + uidAttr;
    ignoredCustomMarkupChunks.push(/^(\s*)[\s\S]*?(\s*)$/.exec(match));
    return '\t' + token + '\t';
  });
}

if (options.sortAttributes && typeof options.sortAttributes !== 'function' ||
    options.sortClassName && typeof options.sortClassName !== 'function') {
  createSortFns(value, options, uidIgnore, uidAttr);
}

function _canCollapseWhitespace(tag, attrs) {
  return options.canCollapseWhitespace(tag, attrs, canCollapseWhitespace);
}

function _canTrimWhitespace(tag, attrs) {
  return options.canTrimWhitespace(tag, attrs, canTrimWhitespace);
}

function removeStartTag() {
  var index = buffer.length - 1;
  while (index > 0 && !/^<[^/!]/.test(buffer[index])) {
    index--;
  }
  buffer.length = Math.max(0, index);
}

function removeEndTag() {
  var index = buffer.length - 1;
  while (index > 0 && !/^<\//.test(buffer[index])) {
    index--;
  }
  buffer.length = Math.max(0, index);
}

// look for trailing whitespaces, bypass any inline tags
function trimTrailingWhitespace(index, nextTag) {
  for (var endTag = null; index >= 0 && _canTrimWhitespace(endTag); index--) {
    var str = buffer[index];
    var match = str.match(/^<\/([\w:-]+)>$/);
    if (match) {
      endTag = match[1];
    }
    else if (/>$/.test(str) || (buffer[index] = collapseWhitespaceSmart(str, null, nextTag, opti
```

```
961           break;
962         }
963       }
964     }
965
966     // look for trailing whitespaces from previously processed text
967     // which may not be trimmed due to a following comment or an empty
968     // element which has now been removed
969     function squashTrailingWhitespace(nextTag) {
970       var charsIndex = buffer.length - 1;
971       if (buffer.length > 1) {
972         var item = buffer[buffer.length - 1];
973         if (/^(?:<!|$)/.test(item) && item.indexOf(uidIgnore) === -1) {
974           charsIndex--;
975         }
976       }
977       trimTrailingWhitespace(charsIndex, nextTag);
978     }
979
980     new HTMLParser(value, {
981       partialMarkup: partialMarkup,
982       continueOnParseError: options.continueOnParseError,
983       customAttrAssign: options.customAttrAssign,
984       customAttrSurround: options.customAttrSurround,
985       html5: options.html5,
986
987       start: function(tag, attrs, unary, unarySlash, autoGenerated) {
988         if (tag.toLowerCase() === 'svg') {
989           options = Object.create(options);
990           options.caseSensitive = true;
991           options.keepClosingSlash = true;
992           options.name = identity;
993         }
994         tag = options.name(tag);
995         currentTag = tag;
996         charsPrevTag = tag;
997         if (!inlineTextTags(tag)) {
998           currentChars = '';
999         }
1000        hasChars = false;
1001        currentAttrs = attrs;
1002
1003        var optional = options.removeOptionalTags;
1004        if (optional) {
1005          var htmlTag = htmlTags(tag);
1006          // <html> may be omitted if first thing inside is not comment
1007          // <head> may be omitted if first thing inside is an element
1008          // <body> may be omitted if first thing inside is not space, comment, <meta>, <link>, <scr
1009          // <colgroup> may be omitted if first thing inside is <col>
```

```
1010          // <tbody> may be omitted if first thing inside is <tr>
1011          if (htmlTag && canRemoveParentTag(optionalStartTag, tag)) {
1012            removeStartTag();
1013          }
1014          optionalStartTag = '';
1015          // end-tag-followed-by-start-tag omission rules
1016          if (htmlTag && canRemovePrecedingTag(optionalEndTag, tag)) {
1017            removeEndTag();
1018            // <colgroup> cannot be omitted if preceding </colgroup> is omitted
1019            // <tbody> cannot be omitted if preceding </tbody>, </thead> or </tfoot> is omitted
1020            optional = !isStartTagMandatory(optionalEndTag, tag);
1021          }
1022          optionalEndTag = '';
1023        }
1024
1025        // set whitespace flags for nested tags (eg. <code> within a <pre>)
1026        if (options.collapseWhitespace) {
1027          if (!stackNoTrimWhitespace.length) {
1028            squashTrailingWhitespace(tag);
1029          }
1030          if (!unary) {
1031            if (!_canTrimWhitespace(tag, attrs) || stackNoTrimWhitespace.length) {
1032              stackNoTrimWhitespace.push(tag);
1033            }
1034            if (!_canCollapseWhitespace(tag, attrs) || stackNoCollapseWhitespace.length) {
1035              stackNoCollapseWhitespace.push(tag);
1036            }
1037          }
1038        }
1039
1040        var openTag = '<' + tag;
1041        var hasUnarySlash = unarySlash && options.keepClosingSlash;
1042
1043        buffer.push(openTag);
1044
1045        if (options.sortAttributes) {
1046          options.sortAttributes(tag, attrs);
1047        }
1048
1049        var parts = [];
1050        for (var i = attrs.length, isLast = true; --i >= 0;) {
1051          var normalized = normalizeAttr(attrs[i], attrs, tag, options);
1052          if (normalized) {
1053            parts.unshift(buildAttr(normalized, hasUnarySlash, options, isLast, uidAttr));
1054            isLast = false;
1055          }
1056        }
1057        if (parts.length > 0) {
1058          buffer.push(' ');
```

```
1059          buffer.push.apply(buffer, parts);
1060        }
1061        // start tag must never be omitted if it has any attributes
1062        else if (optional && optionalStartTags(tag)) {
1063          optionalStartTag = tag;
1064        }
1065
1066        buffer.push(buffer.pop() + (hasUnarySlash ? '/' : '') + '>');
1067
1068        if (autoGenerated && !options.includeAutoGeneratedTags) {
1069          removeStartTag();
1070          optionalStartTag = '';
1071        }
1072      },
1073      end: function(tag, attrs, autoGenerated) {
1074        if (tag.toLowerCase() === 'svg') {
1075          options = Object.getPrototypeOf(options);
1076        }
1077        tag = options.name(tag);
1078
1079        // check if current tag is in a whitespace stack
1080        if (options.collapseWhitespace) {
1081          if (stackNoTrimWhitespace.length) {
1082            if (tag === stackNoTrimWhitespace[stackNoTrimWhitespace.length - 1]) {
1083              stackNoTrimWhitespace.pop();
1084            }
1085          }
1086          else {
1087            squashTrailingWhitespace('/' + tag);
1088          }
1089          if (stackNoCollapseWhitespace.length &&
1090            tag === stackNoCollapseWhitespace[stackNoCollapseWhitespace.length - 1]) {
1091            stackNoCollapseWhitespace.pop();
1092          }
1093        }
1094
1095        var isElementEmpty = false;
1096        if (tag === currentTag) {
1097          currentTag = '';
1098          isElementEmpty = !hasChars;
1099        }
1100
1101        if (options.removeOptionalTags) {
1102          // <html>, <head> or <body> may be omitted if the element is empty
1103          if (isElementEmpty && topLevelTags(optionalStartTag)) {
1104            removeStartTag();
1105          }
1106          optionalStartTag = '';
1107          // </html> or </body> may be omitted if not followed by comment
```

```
1108          // </head> may be omitted if not followed by space or comment
1109          // </p> may be omitted if no more content in non-</a> parent
1110          // except for </dt> or </thead>, end tags may be omitted if no more content in parent elem
1111          if (htmlTags(tag) && optionalEndTag && !trailingTags(optionalEndTag) && (optionalEndTag !=
1112            removeEndTag();
1113          }
1114          optionalEndTag = optionalEndTags(tag) ? tag : '';
1115        }
1116
1117        if (options.removeEmptyElements && isElementEmpty && canRemoveElement(tag, attrs)) {
1118          // remove last "element" from buffer
1119          removeStartTag();
1120          optionalStartTag = '';
1121          optionalEndTag = '';
1122        }
1123        else {
1124          if (autoGenerated && !options.includeAutoGeneratedTags) {
1125            optionalEndTag = '';
1126          }
1127          else {
1128            buffer.push('</' + tag + '>');
1129          }
1130          charsPrevTag = '/' + tag;
1131          if (!inlineTags(tag)) {
1132            currentChars = '';
1133          }
1134          else if (isElementEmpty) {
1135            currentChars += '|';
1136          }
1137        }
1138      },
1139      chars: function(text, prevTag, nextTag) {
1140        prevTag = prevTag === '' ? 'comment' : prevTag;
1141        nextTag = nextTag === '' ? 'comment' : nextTag;
1142        if (options.decodeEntities && text && !specialContentTags(currentTag)) {
1143          text = decode(text);
1144        }
1145        if (options.collapseWhitespace) {
1146          if (!stackNoTrimWhitespace.length) {
1147            if (prevTag === 'comment') {
1148              var prevComment = buffer[buffer.length - 1];
1149              if (prevComment.indexOf(uidIgnore) === -1) {
1150                if (!prevComment) {
1151                  prevTag = charsPrevTag;
1152                }
1153                if (buffer.length > 1 && (!prevComment || !options.conservativeCollapse && / $/.test
1154                  var charsIndex = buffer.length - 2;
1155                  buffer[charsIndex] = buffer[charsIndex].replace(/\s+$/, function(trailingSpaces) {
1156                    text = trailingSpaces + text;
```

```
1157                  return '';
1158                });
1159              }
1160            }
1161          }
1162          if (prevTag) {
1163            if (prevTag === '/nobr' || prevTag === 'wbr') {
1164              if (/^\s/.test(text)) {
1165                var tagIndex = buffer.length - 1;
1166                while (tagIndex > 0 && buffer[tagIndex].lastIndexOf('<' + prevTag) !== 0) {
1167                  tagIndex--;
1168                }
1169                trimTrailingWhitespace(tagIndex - 1, 'br');
1170              }
1171            }
1172            else if (inlineTextTags(prevTag.charAt(0) === '/' ? prevTag.slice(1) : prevTag)) {
1173              text = collapseWhitespace(text, options, /(?:^|\s)$/.test(currentChars));
1174            }
1175          }
1176          if (prevTag || nextTag) {
1177            text = collapseWhitespaceSmart(text, prevTag, nextTag, options);
1178          }
1179          else {
1180            text = collapseWhitespace(text, options, true, true);
1181          }
1182          if (!text && /\s$/.test(currentChars) && prevTag && prevTag.charAt(0) === '/') {
1183            trimTrailingWhitespace(buffer.length - 1, nextTag);
1184          }
1185        }
1186        if (!stackNoCollapseWhitespace.length && nextTag !== 'html' && !(prevTag && nextTag)) {
1187          text = collapseWhitespace(text, options, false, false, true);
1188        }
1189      }
1190      if (options.processScripts && specialContentTags(currentTag)) {
1191        text = processScript(text, options, currentAttrs);
1192      }
1193      if (isExecutableScript(currentTag, currentAttrs)) {
1194        text = options.minifyJS(text);
1195      }
1196      if (isStyleSheet(currentTag, currentAttrs)) {
1197        text = options.minifyCSS(text);
1198      }
1199      if (options.removeOptionalTags && text) {
1200        // <html> may be omitted if first thing inside is not comment
1201        // <body> may be omitted if first thing inside is not space, comment, <meta>, <link>, <scr
1202        if (optionalStartTag === 'html' || optionalStartTag === 'body' && !/^\s/.test(text)) {
1203          removeStartTag();
1204        }
1205        optionalStartTag = '';
```

```
1206          // </html> or </body> may be omitted if not followed by comment
1207          // </head>, </colgroup> or </caption> may be omitted if not followed by space or comment
1208          if (compactTags(optionalEndTag) || looseTags(optionalEndTag) && !/^\s/.test(text)) {
1209            removeEndTag();
1210          }
1211          optionalEndTag = '';
1212        }
1213        charsPrevTag = /^\s*$/.test(text) ? prevTag : 'comment';
1214        if (options.decodeEntities && text && !specialContentTags(currentTag)) {
1215          // Escape any `&` symbols that start either:
1216          // 1) a legacy named character reference (i.e. one that doesn't end with `;`)
1217          // 2) or any other character reference (i.e. one that does end with `;`)
1218          // Note that `&` can be escaped as `&amp`, without the semi-colon.
1219          // https://mathiasbynens.be/notes/ambiguous-ampersands
1220          text = text.replace(/&((?:Iacute|aacute|uacute|plusmn|Otilde|otilde|agrave|Agrave|Yacute|y
1221        }
1222        if (uidPattern && options.collapseWhitespace && stackNoTrimWhitespace.length) {
1223          text = text.replace(uidPattern, function(match, prefix, index) {
1224            return ignoredCustomMarkupChunks[+index][0];
1225          });
1226        }
1227        currentChars += text;
1228        if (text) {
1229          hasChars = true;
1230        }
1231        buffer.push(text);
1232      },
1233      comment: function(text, nonStandard) {
1234        var prefix = nonStandard ? '<!' : '<!--';
1235        var suffix = nonStandard ? '>' : '-->';
1236        if (isConditionalComment(text)) {
1237          text = prefix + cleanConditionalComment(text, options) + suffix;
1238        }
1239        else if (options.removeComments) {
1240          if (isIgnoredComment(text, options)) {
1241            text = '<!--' + text + '-->';
1242          }
1243          else {
1244            text = '';
1245          }
1246        }
1247        else {
1248          text = prefix + text + suffix;
1249        }
1250        if (options.removeOptionalTags && text) {
1251          // preceding comments suppress tag omissions
1252          optionalStartTag = '';
1253          optionalEndTag = '';
1254        }
```

```
      buffer.push(text);
    },
    doctype: function(doctype) {
      buffer.push(options.useShortDoctype ? '<!doctype' +
        (options.removeTagWhitespace ? '' : ' ') + 'html>' :
        collapseWhitespaceAll(doctype));
    }
  });

  if (options.removeOptionalTags) {
    // <html> may be omitted if first thing inside is not comment
    // <head> or <body> may be omitted if empty
    if (topLevelTags(optionalStartTag)) {
      removeStartTag();
    }
    // except for </dt> or </thead>, end tags may be omitted if no more content in parent element
    if (optionalEndTag && !trailingTags(optionalEndTag)) {
      removeEndTag();
    }
  }
  if (options.collapseWhitespace) {
    squashTrailingWhitespace('br');
  }

  return joinResultSegments(buffer, options, uidPattern ? function(str) {
    return str.replace(uidPattern, function(match, prefix, index, suffix) {
      var chunk = ignoredCustomMarkupChunks[+index][0];
      if (options.collapseWhitespace) {
        if (prefix !== '\t') {
          chunk = prefix + chunk;
        }
        if (suffix !== '\t') {
          chunk += suffix;
        }
        return collapseWhitespace(chunk, {
          preserveLineBreaks: options.preserveLineBreaks,
          conservativeCollapse: !options.trimCustomFragments
        }, /^[ \n\r\t\f]/.test(chunk), /[ \n\r\t\f]$/.test(chunk));
      }
      return chunk;
    });
  } : identity, uidIgnore ? function(str) {
    return str.replace(new RegExp('<!--' + uidIgnore + '([0-9]+)-->', 'g'), function(match, index)
      return ignoredMarkupChunks[+index];
    });
  } : identity);
}

function joinResultSegments(results, options, restoreCustom, restoreIgnore) {
```

```
1304    var str;
1305    var maxLineLength = options.maxLineLength;
1306    if (maxLineLength) {
1307      var line = '', lines = [];
1308      while (results.length) {
1309        var len = line.length;
1310        var end = results[0].indexOf('\n');
1311        if (end < 0) {
1312          line += restoreIgnore(restoreCustom(results.shift()));
1313        }
1314        else {
1315          line += restoreIgnore(restoreCustom(results[0].slice(0, end)));
1316          results[0] = results[0].slice(end + 1);
1317        }
1318        if (len > 0 && line.length > maxLineLength) {
1319          lines.push(line.slice(0, len));
1320          line = line.slice(len);
1321        }
1322        else if (end >= 0) {
1323          lines.push(line);
1324          line = '';
1325        }
1326      }
1327      if (line) {
1328        lines.push(line);
1329      }
1330      str = lines.join('\n');
1331    }
1332    else {
1333      str = restoreIgnore(restoreCustom(results.join('')));
1334    }
1335    return options.collapseWhitespace ? collapseWhitespace(str, options, true, true) : str;
1336  }
1337
1338  exports.minify = function(value, options) {
1339    var start = Date.now();
1340    options = processOptions(options || {});
1341    var result = minify(value, options);
1342    options.log('minified in: ' + (Date.now() - start) + 'ms');
1343    return result;
1344  };
```