```
/*
*******************************************************************
**********************
 * GLOBAL OBJECTS > OBJECT
 * https://developer.mozilla.org/en-US/docs/Web/JavaScript/
Reference/Global_Objects/Object
 *
*******************************************************************
********************** */

// Global object: properties
Object.length                                        // length is a
property of a function object, and indicates how many arguments the
function expects, i.e. the number of formal parameters. This number
does not include the rest parameter. Has a value of 1.
Object.prototype                                     // Represents
the Object prototype object and allows to add new properties and
methods to all objects of type Object.

// Methods of the Object constructor
Object.assign(target, ...sources)                    // Copies the
values of all enumerable own properties from one or more source
objects to a target object. method is used to copy the values of all
enumerable own properties from one or more source objects to a
target object. It will return the target object
Object.create(MyObject)                              // Creates a
new object with the specified prototype object and properties. The
object which should be the prototype of the newly-created object.
Object.defineProperty(obj, prop, descriptor)         // Adds the
named property described by a given descriptor to an object.
Object.defineProperties(obj, props)                  // Adds the
named properties described by the given descriptors to an object.
Object.entries(obj)                                  // Returns an
array containing all of the [key, value] pairs of a given object's
own enumerable string properties.
Object.freeze(obj)                                   // Freezes an
object: other code can't delete or change any properties.
Object.getOwnPropertyDescriptor(obj, prop)           // Returns a
property descriptor for a named property on an object.
Object.getOwnPropertyDescriptors(obj)                // Returns an
object containing all own property descriptors for an object.
Object.getOwnPropertyNames(obj)                      // Returns an
array containing the names of all of the given object's own
enumerable and non-enumerable properties.
Object.getOwnPropertySymbols(obj)                    // Returns an
array of all symbol properties found directly upon a given object.
Object.getPrototypeOf(obj)                           // Returns the
prototype of the specified object.
Object.is(value1, value2);                           // Compares if
two values are the same value. Equates all NaN values (which differs
from both Abstract Equality Comparison and Strict Equality
Comparison).
Object.isExtensible(obj)                             // Determines
if extending of an object is allowed.
```

```
Object.isFrozen(obj)                                // Determines
if an object was frozen.
Object.isSealed(obj)                                // Determines
if an object is sealed.
Object.keys(obj)                                    // Returns an
array containing the names of all of the given object's own
enumerable string properties.
Object.preventExtensions(obj)                       // Prevents any
extensions of an object.
Object.seal(obj)                                    // Prevents
other code from deleting properties of an object.
Object.setPrototypeOf(obj, prototype)               // Sets the
prototype (i.e., the internal [[Prototype]] property).
Object.values(obj)                                  // Returns an
array containing the values that correspond to all of a given
object's own enumerable string properties.

// Object instances and Object prototype object
(Object.prototype.property or Object.prototype.method())
// Properties
obj.constructor                                     // Specifies
the function that creates an object's prototype.
obj.__proto__                                       // Points to
the object which was used as prototype when the object was
instantiated.

// Methods
obj.hasOwnProperty(prop)                            // Returns a
boolean indicating whether an object contains the specified property
as a direct property of that object and not inherited through the
prototype chain.
prototypeObj.isPrototypeOf(object)                  // Returns a
boolean indicating whether the object this method is called upon is
in the prototype chain of the specified object.
obj.propertyIsEnumerable(prop)                      // Returns a
boolean indicating if the internal ECMAScript [[Enumerable]]
attribute is set.
obj.toLocaleString()                                // Calls
toString().
obj.toString()                                      // Returns a
string representation of the object.
object.valueOf()                                    // Returns the
primitive value of the specified object.

/*
********************************************************************
***********************
 * GLOBAL OBJECTS > ARRAY
 * https://developer.mozilla.org/en-US/docs/Web/JavaScript/
Reference/Global_Objects/Array
 *
********************************************************************
*********************** */
```

```
// Global object: properties
Array.length                                      // Reflects the
number of elements in an array.
Array.prototype                                   // Represents
the prototype for the Array constructor and allows to add new
properties and methods to all Array objects.

// Global object: methods
Array.from(arrayLike[, mapFn[, thisArg]])         // Creates a
new Array instance from an array-like or iterable object.
Array.isArray(obj)                                // Returns true
if a variable is an array, if not false.
Array.of(element0[, element1[, ...[, elementN]]])    // Creates a
new Array instance with a variable number of arguments, regardless
of number or type of the arguments.

// Instance: properties
arr.length                                        // Reflects the
number of elements in an array.

// Instance: mutator methods
arr.copyWithin(target, start, end)                // Copies a
sequence of array elements within the array.
arr.fill(value, start, end)                       // Fills all
the elements of an array from a start index to an end index with a
static value.
arr.pop()                                         // Removes the
last element from an array and returns that element.
arr.flat()                                        // merges
nested array into one single array
arr.push([element1[, ...[, elementN]]])           // Adds one or
more elements to the end of an array and returns the new length of
the array.
arr.reverse()                                     // Reverses the
order of the elements of an array in place — the first becomes the
last, and the last becomes the first.
arr.shift()                                       // Removes the
first element from an array and returns that element.
arr.sort()                                        // Sorts the
elements of an array in place and returns the array.
array.splice(start, deleteCount, item1, item2, ...)  // Adds and/or
removes elements from an array.
arr.unshift([element1[, ...[, elementN]]])        // Adds one or
more elements to the front of an array and returns the new length of
the array.

// Instance: accessor methods
arr.concat(value1[, value2[, ...[, valueN]]])     // Returns a
new array comprised of this array joined with other array(s) and/or
value(s).
arr.includes(searchElement, fromIndex)            // Determines
whether an array contains a certain element, returning true or false
as appropriate.
arr.indexOf(searchElement[, fromIndex])           // Returns the
```

```
first (least) index of an element within the array equal to the
specified value, or -1 if none is found.
arr.join(separator)                                 // Joins all
elements of an array into a string.
arr.lastIndexOf(searchElement, fromIndex)           // Returns the
last (greatest) index of an element within the array equal to the
specified value, or -1 if none is found.
arr.slice(begin, end)                               // Extracts a
section of an array and returns a new array.
arr.toString()                                      // Returns a
string representing the array and its elements. Overrides the
Object.prototype.toString() method.
arr.toLocaleString(locales, options)                // Returns a
localized string representing the array and its elements. Overrides
the Object.prototype.toLocaleString() method.

// Instance: iteration methods
arr.entries()                                       // Returns a
new Array Iterator object that contains the key/value pairs for each
index in the array.
arr.every(callback[, thisArg])                      // Returns true
if every element in this array satisfies the provided testing
function.
arr.filter(callback[, thisArg])                     // Creates a
new array with all of the elements of this array for which the
provided filtering function returns true.
arr.find(callback[, thisArg])                       // Returns the
found value in the array, if an element in the array satisfies the
provided testing function or undefined if not found.
arr.findIndex(callback[, thisArg])                  // Returns the
found index in the array, if an element in the array satisfies the
provided testing function or -1 if not found.
arr.forEach(callback[, thisArg])                    // Calls a
function for each element in the array.
arr.keys()                                          // Returns a
new Array Iterator that contains the keys for each index in the
array.
arr.map(callback[, initialValue])                   // Creates a
new array with the results of calling a provided function on every
element in this array.
arr.reduce(callback[, initialValue])                // Apply a
function against an accumulator and each value of the array (from
left-to-right) as to reduce it to a single value.
arr.reduceRight(callback[, initialValue])           // Apply a
function against an accumulator and each value of the array (from
right-to-left) as to reduce it to a single value.
arr.some(callback[, initialValue])                  // Returns true
if at least one element in this array satisfies the provided testing
function.
arr.values()                                        // Returns a
new Array Iterator object that contains the values for each index in
the array.
```

```
<!-- *
*******************************************************************
**********************
 * HTML5 Cheat sheet by Hackr.io
 * Source: https://websitesetup.org/wp-content/uploads/2014/02/HTML-
CHEAT-SHEET-768x8555.png
 *
*******************************************************************
********************** * -->


<!-- Document Summary -->

<!DOCTYPE html>                                      <!-- Tells the
browser that HTML5 version of HTML to be recognized by the browser
-->
<html lang="en"></html>                              <!-- The HTML lang
attribute is used to identify the language of text content on the
web. This information helps search engines return language specific
results, -->
<head></head>                                        <!-- Contains
Information specific to the page like title, styles and scripts -->
<title></title>                                      <!-- Title for the
page that shows up in the browser title bar -->
<body></body>                                        <!-- Content that
the user will see -->


<!-- Document Information -->


<base/>                                              <!-- Usefull for
specifying relative links in a document -->
<style></style>                                      <!-- Contains
styles for the html document -->
<meta/>                                              <!-- Contains
additional information about the page, author, page description and
other hidden page info -->
<script></script>                                    <!-- Contains all
scripts internal or external -->
<link/>                                              <!-- Used to
create relationships with external pages and stylesheets -->


<!-- Document Structure -->


<h1></h1> ... <h6></h6>                              <!-- All six
levels of heading with 1 being the most promiment and 6 being the
least prominent -->
<p></p>                                              <!-- Used to
organize paragraph text -->
```

```
<div></div>                                         <!-- A generic
container used to denote a page section -->
<span></span>                                       <!-- Inline
section or block container used for creating inline style elements
-->
<br/>                                               <!-- Creates a
line-break -->
<hr>                                                <!-- Creates a
sectional break into HTML -->


<!-- Text Formatting -->


<strong></strong> and <b></b>                       <!-- Makes text
contained in the tag as bold -->
<em></em> and <i></i>                               <!-- Alternative
way to make the text contained in the tag as italic -->
<del></del>                                   <!-- Creates a strike
through the text element -->
<pre></pre>                                         <!-- Preformatted
monospace text block with some spacing intact -->
<blockquote></blockquote>                           <!-- Contains long
paragraphs of quotations often cited -->
<abbr></abbr>                                       <!-- Contains
abbreviations while also making the full form avaialable -->
<address></address>                                 <!-- Used to
display contact information -->
<code></code>                                       <!-- Used to
display inline code snippets -->
<q></q>                                             <!-- Defines a
short inline quotation -->
<sub></sub>                                         <!-- Defines
subscripted text -->
<sup></sup>                                         <!-- Defines
superscripted text -->
<kbd></kbd>                                         <!-- Specifies
text as keyboard input -->
<small></small>                                     <!-- Specifies
small text -->
<ins></ins>                                         <!-- Defines a
text that has been inserted into the document -->


<!-- Links Formatting -->


<a href="url"></a>                                  <!-- Used to link
to external or internal pages of a wbesite -->
<a href="mailto:email@example.com"></a>             <!-- Used to link
to an email address -->
<a href="name"></a>                                 <!-- Used to link
to a document element -->
<a href="#name"></a>                                <!-- Used to link
```

```
to specific div element -->
<a href="tel://####-####-##"></a>                <!-- Used to
display phone numbers and make them clickable -->


<!-- Image Formatting -->


<img src="url" alt="text">                       <!-- Used to
display images in a webpage where src="url" contains the link to the
image source and alt="" contains an alternative text to display when
the image is not displayed -->


<!-- List Formatting -->


<ol></ol>                                        <!-- Used to
create ordered lists with numbers in the items -->
<ul></ul>                                        <!-- Used to
display unordered lists with numbers in the items -->
<li></li>                                        <!-- Contains list
items inside ordered and unordered lists -->
<dl></dl>                                        <!-- Contains list
item definitions -->
<dt></dt>                                        <!-- Definition of
single term inline with body content -->
<dd></dd>                                        <!-- The
descrpition of the defined term -->


<!-- Forms Formatting and Attributes -->


<form action="url"></form>                       <!-- Form element
creates a form and action="" specifies where the data is to be sent
to when the visitor submits the form -->

<!-- Supported attributes -->
method="somefunction()"                          <!-- Contains the
type of request (GET, POST... etc) which dictates how to send the
data of the form -->
enctype=""                                       <!-- Dictates how
the data is to be encoded when the data is sent to the web server.
-->
autocomplete=""                                  <!-- Specifies if
the autocomplete functionality is enabled or not -->
novalidate                                       <!-- Dictates if
the form will be validated or not -->
accept-charset=""                                <!-- Identifies
the character encoding upon form submission -->
target=""                                        <!-- Tell where to
display the information upon form submission. Possible values:
'_blank', '_self', '_parent', '_top' -->
```

```
<fieldset disabled="disabled"></fieldset>         <!-- Identifies
the group of all fields in the form -->
<label for=""></label>                            <!-- A simple
field label telling the user what to type in the field -->
<legend></legend>                                 <!-- The form
legend acts as a caption for the fieldset element -->

<input type="text/email/number/color/date">       <!-- Input is the
input field where the user can input various types of data -->

<!-- Supported attributes -->
name=""                                           <!-- Describes the
name of the form -->
width=""                                          <!-- Specifies the
width of an input field -->
value=""                                          <!-- Describes the
value of the input information field -->
size=""                                           <!-- Specifies the
input element width in characters -->
maxlength=""                                      <!-- Specifies the
maximum input character numbers -->
required=""                                       <!-- Specifies if
the input field is required to fill in before submitting the form --
>
step=""                                           <!-- Identifies
the legal number intervals of the input field -->

<textarea name="" id="" cols="30" rows="10">      <!-- Specifies a
large input text field for longer messages -->
</textarea>

<select name=""></select>                         <!-- Describes a
dropdown box for users to select from variety of choices -->

<!-- Supported attributes -->
name=""                                           <!-- The name for
a dropdown combination box -->
size=""                                           <!-- Specifies the
number of available options -->
multiple                                          <!-- Allows for
multiple option selections -->
required                                          <!-- Requires that
a value is selected before submitting the form -->
autofocus                                         <!-- Specifies
that the dropdown automatically comes to focus once the page loads
-->
<optgroup></optgroup>                             <!-- Specifies the
entire grouping of available options -->
<option value=""></option>                        <!-- Defines one
of the avaialble option from the dropdown list -->
<button></button>                                 <!-- A clickable
button to submit the form -->
```

```
<!-- Tables Formatting -->


<table></table>                                  <!-- Defines and
contains all table related content -->
<caption></caption>                              <!-- A description
of what table is and what it contains -->
<thead></thead>                                  <!-- The table
headers contain the type of information defined in each column
underneath -->
<tbody></tbody>                                  <!-- Contains the
tables data or information -->
<tfoot></tfoot>                                  <!-- Defines table
footer -->
<tr></tr>                                        <!-- Contains the
information to be included in a table row -->
<th></th>                                        <!-- Contains the
information to be included in a single table header -->
<td></td>                                        <!-- Contains
actual information in a table cell -->
<colgroup></colgroup>                            <!-- Groups a
single or multiple columns for formatting purposes -->
<col>                                            <!-- Defines a
single column of information inside a table -->


<!-- Objects and iFrames -->


<object data=""></object>                        <!-- Describes and
embed file type including audio, video, PDF's, images -->

<!-- Supported attributes -->
type=""                                          <!-- Describes the
type of media embedded -->
height=""                                        <!-- Describes the
height of the object in pixels -->
width=""                                         <!-- Describes the
width of the object in pixels -->
usemap=""                                        <!-- This is the
name of the client-side image map in the object -->

<iframe src="" frameborder="0"></iframe>         <!-- Contains an
inline frame that allows to embed external information -->
<embed src="" type="">                           <!-- Acts as a
container for external application or plug-in -->
src=""                                           <!-- The source of
the external file you're embedding -->
width=""                                         <!-- Describes the
width of the iframe in pixels -->


<!-- HTML5 New Tags -->
```

```
<header></header>                                 <!-- Defines the
header block for a document or a section -->
<footer></footer>                                 <!-- Defines the
footer block for a document or a section -->
<main></main>                                     <!-- Describes the
main content of a document -->
<article></article>                               <!-- Identifies an
article inside a document -->
<aside></aside>                                   <!-- Specifies
content contained in a document sidebar -->
<section></section>                               <!-- Defines a
section of a document -->
<details></details>                               <!-- Describes
additonal information that user can view or hide -->
<dialog></dialog>                                 <!-- A dialog box
or a window -->
<figure></figure>                                 <!-- An
independent content block featuring images, diagrams or
illustrations -->
<figcaption></figcaption>                         <!-- Caption that
describe a figure -->
<mark></mark>                                     <!-- Displays a
portion of highlighted text with in a page content -->
<nav></nav>                                       <!-- Navigation
links for the user in a document -->
<menuitem></menuitem>                             <!-- The specific
menu item that a user can raise from a pop up menu -->
<meter></meter>                                   <!-- Describes the
scalar measurement with in a known array -->
<progress></progress>                             <!-- Displays the
progress of a task usually a progress bar -->
<rp></rp>                                         <!-- Describes
text within the browsers that do not support ruby notations -->
<rt></rt>                                         <!-- Displays east
asian typography character details -->
<ruby></ruby>                                     <!-- Describes
annotations for east asian typography -->
<summary></summary>                               <!-- Contains a
visible heading for details element -->
<bdi></bdi>                                       <!-- Helps you
format parts of text in a different direction than other text -->
<time></time>                                     <!-- Identifies
the time and date -->
<wbr>                                             <!-- A line break
within the content -->

<!-- Some other useful tags -->

<canvas></canvas>                                 <!-- Allows to
draw 2D shapes on the web page with the help of javascript -->
<keygen>                                          <!-- Represents a
control for generating a public-private key pair -->
```

```
<map></map>                                       <!-- Specifies an
image map -->

<!-- Collective Character Obejcts -->


&#34; &quot; Quotation Marks - "
&#38; &amp; Ampersand - &
&#60; &lt; Less than sign - <
&#62; &gt; Greater than sign - >
&#160; &nbsp; Non-breaking space
&#169; &copy; Copyright Symbol - ©
&#64; &Uuml; @ symbol - @
&#149; &ouml; Small bullet - .
&#153; &ucirc; Trademark Symbol - ™
```

```
/****************************
 * CSS3 CHEATSHEET - Beginner Friendly
 * Learn more: https://web.dev/learn/css/
 * Documentation: https://developer.mozilla.org/en-US/docs/Web/CSS
 * PDF for Better Readability: https://github.com/LeCoupa/awesome-
cheatsheets/files/8880372/CSS_Cheatsheet.pdf
 * Brief overview about CSS: http://jonkinney.com/assets/21/
advanced_css.pdf
 * (Advance) Know more about each topic in-depth: https://
www.ciwcertified.com/resources/documents/sample-chapter/
CCT02CDHTCSCK1405.PDF
 *
 *
 *
 *  Table of contents
 *  -------------------
 *  01 | Font
 *  02 | Text
 *  03 | Background
 *  04 | Border
 *  05 | Box Model
 *  06 | Colors
 *  07 | Template Layout
 *  08 | Table
 *  09 | Columns
 *  10 | List & Markers
 *  11 | Animations
 *  12 | Transitions
 *  13 | CSS Flexbox (Important)
 *        - Parent Properties (flex container)
 *        - Child Properties (flex items)
 *  14 | CSS Grid (Useful in Complex Web Designs)
 *        - Parent Properties (Grid container)
 *        - Child Properties (Grid items)
 *  15 | Media Queries
 *
 *
 *
 ****************************/

/***************************

------------ 01: Font -----------

There are many properties related to the font, such as the face,
weight, style, etc. These
properties allow you to change the style or complete look of your
text.

*******************************/

/** Body Selector which applies properties to whole body <body></
body> */
body {
  /* Font-Family */
  font-family: "Segoe UI", Tahoma, sans-serif; /* You can declare
multiple fonts. */
  /*if first font doesn't exists other ones will be declared serial
wise */

  /* Font-Style */
  font-style: italic;

  /* Font-Variant */
  font-variant: small-caps;

  /* Font-Weight */
  font-weight: bold;

  /* Font-Size */
  font-size: larger;

  /* Font */
  font: style variant weight size family;
}

/***************************

------------ 02: Text -----------

Text properties allow one to manipulate alignment, spacing,
decoration, indentation, etc., in the
document.

*******************************/
```

```
/* Applies to all tags with class 'container' ex: <div
class="container"></div> */
.container {
  /* Text-Align */
  text-align: justify;

  /* Letter-Spacing */
  letter-spacing: 0.15em;

  /* Text-Decoration */
  text-decoration: underline;

  /* Word-Spacing */
  word-spacing: 0.25em;

  /* Text-Transform */
  text-transform: uppercase;

  /* Text-Indent */
  text-indent: 0.5cm;

  /* Line-Height */
  line-height: normal;
}

/***************************

------------ 03: Background -----------

As the name suggests, these properties are related to background,
i.e., you can change the color,
image, position, size, etc., of the document.

*******************************/

/* Applies to all tags with id 'wrapper' ex: <div id="wrapper"></
div> */
#wrapper {
  /* Background-Image */
  background-image: url("Path");

  /* Background-Position */
  background-position: right top;

  /* Background-Size */
  background-size: cover;

  /* Background-Repeat */
  background-repeat: no-repeat;

  /* Background-Attachment */
  background-attachment: scroll;
  /* Background-Color */
  background-color: fuchsia;

  /* Background */
  background: color image repeat attachment position;
}

/****************************

------------ 04: Border -----------

Border properties are used to change the style, radius, color, etc.,
of buttons or other items of
the document.

********************************/

/* You can also select multiple items */
div,
.container {
  /* Border-Width */
  border-width: 5px;

  /* Border-Style */
  border-style: solid;

  /* Border-Color */
  border-color: aqua;

  /* Border-Radius */
  border-radius: 15px;

  /* Border */
  border: width style color;
}

/****************************

------------ 05: Box Model -----------

In laymen's terms, the CSS box model is a container that wraps
around every HTML element. It
consists of margins, borders, padding, and the actual content.
It is used to create the design and layout of web pages.

********************************/

.wrapper {
  /* Float */
  float: none;
  /* Clear */
  clear: both;
  /* Display */
  display: block;
  /* Height */
  height: fit-content;
  /* Width */
  width: auto;
  /* Margin */
  margin: top right bottom left;
  /* Padding */
  padding: top right bottom left;
  /* Overflow */
  overflow: hidden;
  /* Visibility */
  visibility: visible;
  /* z-index */
  z-index: 1;
  /* position */
  position: static | relative | fixed | absolute | sticky;

}

/***************************

------------ 06: Colors -----------

With the help of the color property, one can give color to text,
shape, or any other object.

******************************/

p,
span,
.text {
  /* Color - 1 */
  color: cornsilk;
  /* Color - 2 */
  color: #fff8dc;
  /* Color - 3 */
  color: rgba(255, 248, 220, 1);
  /* Color - 4 */
  color: hsl(48, 100%, 93%);
  /* Opacity */
  opacity: 1;
}

/***************************

------------ 07: Template Layout -----------

Specifies the visual look of the content inside a template

******************************/

/* '*' selects all elements on a page */
* {
  /* Box-Align */
  box-align: start;

  /* Box-Direction */
  box-direction: normal;

  /* Box-Flex */
  box-flex: normal;

  /* Box-Flex-Group */
  box-flex-group: 2;

  /* Box-Orient */
  box-orient: inline;

  /* Box-Pack */
  box-pack: justify;

  /* Box-Sizing */
  box-sizing: margin-box;

  /* max-width */
  max-width: 800px;

  /* min-width */
  min-width: 500px;

  /* max-height */
  max-height: 100px;

  /* min-height */
  min-height: 80px;
}

/****************************

------------ 08: Table -----------

Table properties are used to give style to the tables in the
document. You can change many
things like border spacing, table layout, caption, etc.

*******************************/

table {
  /* Border-Collapse */
  border-collapse: separate;

  /* Empty-Cells */
  empty-cells: show;

  /* Border-Spacing */
  border-spacing: 2px;

  /* Table-Layout */
  table-layout: auto;

  /* Caption-Side */
  caption-side: bottom;
}

/****************************

------------ 09: Columns -----------

These properties are used explicitly with columns of the tables, and
they are used to give the
table an incredible look.

********************************/

/* Applies to <table class="nice-table"></table> */
/* Not <table></table> */
table.nice-table {
  /* Column-Count */
  column-count: 10;

  /* Column-Gap */
  column-gap: 5px;

  /* Column-rule-width */
  column-rule-width: medium;

  /* Column-rule-style */
  column-rule-style: dotted;

  /* Column-rule-color */
  column-rule-color: black;

  /* Column-width */
  column-width: 10px;

  /* Column-span */
  column-span: all;
}

/****************************

------ 10: List & Markers -------

List and marker properties are used to customize lists in the
document.

********************************/

li,
ul,
ol {
  /* List-style-type */
  list-style-type: square;

  /* List-style-position */
  list-style-position: 20px;

  /* List-style-image */
  list-style-image: url("image.gif");

  /* Marker-offset */
  marker-offset: auto;
}

/****************************

------------ 11: Animations -----------

CSS animations allow one to animate transitions or other media files
on the web page.

********************************/

svg,
.loader {
  /* Animation-name */
  animation-name: myanimation;

  /* Animation-duration */
  animation-duration: 10s;

  /* Animation-timing-function */
  animation-timing-function: ease;

  /* Animation-delay */
  animation-delay: 5ms;

  /* Animation-iteration-count */
  animation-iteration-count: 3;

  /* Animation-direction */
  animation-direction: normal;

  /* Animation-play-state */
  animation-play-state: running;

  /* Animation-fill-mode */
  animation-fill-mode: both;
}

/****************************

------------ 12: Transitions -----------

Transitions let you define the transition between two states of an
element.
```

```
******************************/

a,
button {
  /* Transition-property */
  transition-property: none;

  /* Transition-duration */
  transition-duration: 2s;

  /* Transition-timing-function */
  transition-timing-function: ease-in-out;

  /* Transition-delay */
  transition-delay: 20ms;
}

/**************************

------------ 13: CSS Flexbox (Important) -----------

Flexbox is a layout of CSS that lets you format HTML easily. Flexbox
makes it simple to align
items vertically and horizontally using rows and columns. Items will
"flex" to different sizes to fill
the space. And overall, it makes the responsive design more
manageable.

******************************/

/* ---------------------- Parent Properties (flex container)
------------ */

section,
div#wrapper {
  /* display */
  display: flex;

  /* flex-direction */
  flex-direction: row | row-reverse | column | column-reverse;

  /* flex-wrap */
  flex-wrap: nowrap | wrap | wrap-reverse;

  /* flex-flow */
  flex-flow: column wrap;

  /* justify-content */
  justify-content: flex-start | flex-end | center | space-between |
space-around;

  /* align-items */
  align-items: stretch | flex-start | flex-end | center | baseline;
  /* align-content */
  align-content: flex-start | flex-end | center | space-between |
space-around;
}


/* ---------------------- Child Properties (flex items) ------------
*/

p,
span,
h1,
h2,
h3,
h4,
h5,
h6,
a {
    /* order */
    order: 5; /* default is 0 */

    /* flex-grow */
    flex-grow: 4; /* default 0 */

    /* flex-shrink */
    flex-shrink: 3; /* default 1 */

    /* flex-basis */
    flex-basis: | auto; /* default auto */

    /* flex shorthand */
    flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-
basis'> ]

    /* align-self */
    align-self: auto | flex-start | flex-end | center | baseline |
stretch;
}

/****************************

------------ 14: CSS Grid (Useful in Complex Web Designs)
-----------

Grid layout is a 2-Dimensional grid system to CSS that creates
complex responsive web design
layouts more easily and consistently across browsers.

********************************/


/* ---------------------- Parent Properties (Grid container)
------------ */
```

```
section,
div#wrapper {
    /* display */
    display: grid | inline-grid;

    /* grid-template-columns */
    grid-template-columns: 12px 12px 12px;

    /* grid-template-rows */
    grid-template-rows: 8px auto 12px;

    /* grid-template */
    grid-template: none | <grid-template-rows> / <grid-template-
columns>;

    /* column-gap */
    column-gap: <line-size>;

    /* row-gap */
    row-gap: <line-size>;

    /* grid-column-gap */
    grid-column-gap: <line-size>;

    /* grid-row-gap */
    grid-row-gap: <line-size>;

    /* gap shorthand */
    gap: <grid-row-gap> <grid-column-gap>;

    /* grid-gap shorthand */
    grid-gap: <grid-row-gap> <grid-column-gap>;

    /* justify-items */
    justify-items: start | end | center | stretch;

    /* align-items */
    align-items: start | end | center | stretch;

    /* place-items */
    place-items: center;

    /* justify-content */
    justify-content: start | end | center | stretch | space-around |
space-between;

    /* align-content */
    align-content: start | end | center | stretch | space-around |
space-between;

    /* place-content */
    place-content: <align-content> / <justify-content>;
    /* grid-auto-columns */
    grid-auto-columns: <track-size> ...;

    /* grid-auto-rows */
    grid-auto-rows: <track-size> ...;

    /* grid-auto-flow */
    grid-auto-flow: row | column | row dense | column dense;

}


/* ---------------------- Child Properties (Grid items) ------------
*/

p,
span,
h1,
h2,
h3,
h4,
h5,
h6,
a {
    /* grid-column-start */
    grid-column-start: <number> | <name> | span <number> | span
<name> | auto;

    /* grid-column-end */
    grid-column-end: <number> | <name> | span <number> | span <name>
| auto;

    /* grid-row-start */
    grid-row-start: <number> | <name> | span <number> | span <name>
| auto;

    /* grid-row-end */
    grid-row-end: <number> | <name> | span <number> | span <name> |
auto;

    /* grid-column shorthand */
    grid-column: <start-line> / <end-line> | <start-line> / span
<value>;

    /* grid-row shorthand */
    grid-row: <start-line> / <end-line> | <start-line> / span
<value>;

    /* grid-area */
    grid-area: <name> | <row-start> / <column-start> / <row-end> /
<column-end>;

    /* justify-self */
    justify-self: start | end | center | stretch;
    /* align-self */
    align-self: start | end | center | stretch;

    /* place-self */
    place-self: center;
}


/***************************

------------ 15: MEDIA QUERIES -----------

Using media queries are a popular technique for delivering a
tailored style sheet to
desktops, laptops, tablets, and mobile phones (such as iPhone and
Android phones).

|----------------------------------------------------------|
|  Responsive Grid Media Queries - 1280, 1024, 768, 480    |
|   1280-1024   - desktop (default grid)                   |
|   1024-768    - tablet landscape                         |
|   768-480     - tablet                                   |
|   480-less    - phone landscape & smaller                |
|----------------------------------------------------------|

*******************************/


@media all and (min-width: 1024px) and (max-width: 1280px) { }

@media all and (min-width: 768px) and (max-width: 1024px) { }

@media all and (min-width: 480px) and (max-width: 768px) { }

@media all and (max-width: 480px) { }

/* Small screens - MOBILE */
@media only screen { } /* Define mobile styles - Mobile First */

@media only screen and (max-width: 40em) { } /* max-width 640px,
mobile-only styles, use when QAing mobile issues */

/* Medium screens - TABLET */
@media only screen and (min-width: 40.063em) { } /* min-width 641px,
medium screens */

@media only screen and (min-width: 40.063em) and (max-width: 64em)
{ } /* min-width 641px and max-width 1024px, use when QAing tablet-
only issues */

/* Large screens - DESKTOP */
@media only screen and (min-width: 64.063em) { } /* min-width
1025px, large screens */
```

```
@media only screen and (min-width: 64.063em) and (max-width: 90em)
{ } /* min-width 1024px and max-width 1440px, use when QAing large
screen-only issues */

/* XLarge screens */
@media only screen and (min-width: 90.063em) { } /* min-width
1441px, xlarge screens */

@media only screen and (min-width: 90.063em) and (max-width: 120em)
{ } /* min-width 1441px and max-width 1920px, use when QAing xlarge
screen-only issues */

/* XXLarge screens */
@media only screen and (min-width: 120.063em) { } /* min-width
1921px, xlarge screens */

/*------------------------------------------*/



/* Portrait */
@media screen and (orientation:portrait) { /* Portrait styles here
*/ }
/* Landscape */
@media screen and (orientation:landscape) { /* Landscape styles here
*/ }


/* CSS for iPhone, iPad, and Retina Displays */

/* Non-Retina */
@media screen and (-webkit-max-device-pixel-ratio: 1) {
}

/* Retina */
@media only screen and (-webkit-min-device-pixel-ratio: 1.5),
only screen and (-o-min-device-pixel-ratio: 3/2),
only screen and (min--moz-device-pixel-ratio: 1.5),
only screen and (min-device-pixel-ratio: 1.5) {
}

/* iPhone Portrait */
@media screen and (max-device-width: 480px) and
(orientation:portrait) {
}

/* iPhone Landscape */
@media screen and (max-device-width: 480px) and
(orientation:landscape) {
}

/* iPad Portrait */
@media screen and (min-device-width: 481px) and
```

~~~~
(orientation:portrait) {
}

/* iPad Landscape */
@media screen and (min-device-width: 481px) and
(orientation:landscape) {
}

/* Make Sure you don't forgot to add */
<meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no" /> /* within <head> tag  */
~~~~

~~~~
/*
********************************************************************
***********************
 * REACT.JS CHEATSHEET
 * DOCUMENTATION: https://reactjs.org/docs/
 * FILE STRUCTURE: https://reactjs.org/docs/faq-structure.html
 *
********************************************************************
*********************** */


```
npm install --save react       // declarative and flexible
JavaScript library for building UI
npm install --save react-dom   // serves as the entry point of the
DOM-related rendering paths
npm install --save prop-types  // runtime type checking for React
props and similar objects
```

// notes: don't forget the command lines


/*
********************************************************************
***********************
 * REACT
 * https://reactjs.org/docs/react-api.html
 *
~~~~

```
********************************************************************
*********************** */


// Create and return a new React element of the given type.
// Code written with JSX will be converted to use
React.createElement().
// You will not typically invoke React.createElement() directly if
you are using JSX.
React.createElement(
  type,
  [props],
  [...children]
)

// Clone and return a new React element using element as the
starting point.
// The resulting element will have the original element’s props with
the new props merged in shallowly.
React.cloneElement(
  element,
  [props],
  [...children]
)

// Verifies the object is a React element. Returns true or false.
React.isValidElement(object)

React.Children  // provides utilities for dealing with the
this.props.children opaque data structure.

// Invokes a function on every immediate child contained within
children with this set to thisArg.
React.Children.map(children, function[(thisArg)])

// Like React.Children.map() but does not return an array.
React.Children.forEach(children, function[(thisArg)])

// Returns the total number of components in children,
// equal to the number of times that a callback passed to map or
forEach would be invoked.
React.Children.count(children)

// Verifies that children has only one child (a React element) and
returns it.
// Otherwise this method throws an error.
React.Children.only(children)

// Returns the children opaque data structure as a flat array with
keys assigned to each child.
// Useful if you want to manipulate collections of children in your
render methods,
// especially if you want to reorder or slice this.props.children
before passing it down.
```

```
React.Children.toArray(children)

// The React.Fragment component lets you return multiple elements in
a render() method without creating an additional DOM element
// You can also use it with the shorthand <></> syntax.
React.Fragment


/*
*********************************************************************
***********************
 * REACT.COMPONENT
 * React.Component is an abstract base class, so it rarely makes
sense to refer to React.Component
 * directly. Instead, you will typically subclass it, and define at
least a render() method.
 * https://reactjs.org/docs/react-component.html
 *
*********************************************************************
*********************** */


class Component extends React.Component {
  // Will be called before it is mounted
  constructor(props) {
    // Call this method before any other statement
    // or this.props will be undefined in the constructor
    super(props);

    // The constructor is also often used to bind event handlers to
the class instance.
    // Binding makes sure the method has access to component
attributes like this.props and this.state
    this.method = this.method.bind(this);

    // The constructor is the right place to initialize state.
    this.state = {
      active: true,

      // In rare cases, it’s okay to initialize state based on
props.
      // This effectively “forks” the props and sets the state with
the initial props.
      // If you “fork” props by using them for state, you might also
want to implement componentWillReceiveProps(nextProps)
      // to keep the state up-to-date with them. But lifting state
up is often easier and less bug-prone.
      color: props.initialColor
    };
  }

  // Enqueues changes to the component state and
  // tells React that this component and its children need to be re-
rendered with the updated state.
```

```
  // setState() does not always immediately update the component. It
may batch or defer the update until later.
  // This makes reading this.state right after calling setState() a
potential pitfall.
  // Instead, use componentDidUpdate or a setState callback.
  // You may optionally pass an object as the first argument to
setState() instead of a function.
  setState(updater[, callback]) { }

  // Invoked just before mounting occurs (before render())
  // This is the only lifecycle hook called on server rendering.
  componentWillMount() { }

  // Invoked immediately after a component is mounted.
  // Initialization that requires DOM nodes should go here.
  // If you need to load data from a remote endpoint, this is a good
place to instantiate the network request.
  // This method is a good place to set up any subscriptions. If you
do that, don’t forget to unsubscribe in componentWillUnmount().
  componentDidMount() { }

  // Invoked before a mounted component receives new props.
  // If you need to update the state in response to prop changes
(for example, to reset it),
  // you may compare this.props and nextProps and perform state
transitions using this.setState() in this method.
  componentWillReceiveProps(nextProps) { }

  // Let React know if a component’s output is not affected by the
current change in state or props.
  // The default behavior is to re-render on every state change, and
in the vast majority of cases you should rely on the default
behavior.
  // shouldComponentUpdate() is invoked before rendering when new
props or state are being received. Defaults to true.
  // This method is not called for the initial render or when
forceUpdate() is used.
  // Returning false does not prevent child components from re-
rendering when their state changes.
  shouldComponentUpdate(nextProps, nextState) { }

  // Invoked just before rendering when new props or state are being
received.
  // Use this as an opportunity to perform preparation before an
update occurs. This method is not called for the initial render.
  // Note that you cannot call this.setState() here; nor should you
do anything else
  // (e.g. dispatch a Redux action) that would trigger an update to
a React component before componentWillUpdate() returns.
  // If you need to update state in response to props changes, use
componentWillReceiveProps() instead.
  componentWillUpdate(nextProps, nextState) { }

  // Invoked immediately after updating occurs. This method is not
```

```
called for the initial render.
  // Use this as an opportunity to operate on the DOM when the
component has been updated.
  // This is also a good place to do network requests as long as you
compare the current props to previous props (e.g. a network request
may not be necessary if the props have not changed).
  componentDidUpdate(prevProps, prevState) { }

  // Invoked immediately before a component is unmounted and
destroyed.
  // Perform any necessary cleanup in this method, such as
invalidating timers, canceling network requests,
  // or cleaning up any subscriptions that were created in
componentDidMount().
  componentWillUnmount() { }

  // Error boundaries are React components that catch JavaScript
errors anywhere in their child component tree,
  // log those errors, and display a fallback UI instead of the
component tree that crashed.
  // Error boundaries catch errors during rendering, in lifecycle
methods, and in constructors of the whole tree below them.
  componentDidCatch() { }

  // This method is required.
  // It should be pure, meaning that it does not modify component
state,
  // it returns the same result each time it’s invoked, and
  // it does not directly interact with the browser (use lifecycle
methods for this)
  // It must return one of the following types: react elements,
string and numbers, portals, null or booleans.
  render() {
    // Contains the props that were defined by the caller of this
component.
    console.log(this.props);

    // Contains data specific to this component that may change over
time.
    // The state is user-defined, and it should be a plain
JavaScript object.
    // If you don’t use it in render(), it shouldn’t be in the
state.
    // For example, you can put timer IDs directly on the instance.
    // Never mutate this.state directly, as calling setState()
afterwards may replace the mutation you made.
    // Treat this.state as if it were immutable.
    console.log(this.state);

    return (
      <div>
        {/* Comment goes here */}
        Hello, {this.props.name}!
      </div>
    );
  }
}

// Can be defined as a property on the component class itself, to
set the default props for the class.
// This is used for undefined props, but not for null props.
Component.defaultProps = {
  color: 'blue'
};

component = new Component();

// By default, when your component’s state or props change, your
component will re-render.
// If your render() method depends on some other data, you can tell
React that the component needs re-rendering by calling
forceUpdate().
// Normally you should try to avoid all uses of forceUpdate() and
only read from this.props and this.state in render().
component.forceUpdate(callback)


/*
*******************************************************************
***********************
 * REACT.DOM
 * The react-dom package provides DOM-specific methods that can be
used at the top level of
 * your app and as an escape hatch to get outside of the React model
if you need to.
 * Most of your components should not need to use this module.
 * https://reactjs.org/docs/react-dom.html
 *
*******************************************************************
*********************** */


// Render a React element into the DOM in the supplied container and
return a reference
// to the component (or returns null for stateless components).
ReactDOM.render(element, container[, callback])

// Same as render(), but is used to hydrate a container whose HTML
contents were rendered
// by ReactDOMServer. React will attempt to attach event listeners
to the existing markup.
ReactDOM.hydrate(element, container[, callback])

// Remove a mounted React component from the DOM and clean up its
event handlers and state.
// If no component was mounted in the container, calling this
function does nothing.
// Returns true if a component was unmounted and false if there was
```

```
no component to unmount.
ReactDOM.unmountComponentAtNode(container)

// If this component has been mounted into the DOM, this returns the
corresponding native browser
// DOM element. This method is useful for reading values out of the
DOM, such as form field values
// and performing DOM measurements. In most cases, you can attach a
ref to the DOM node and avoid
// using findDOMNode at all.
ReactDOM.findDOMNode(component)

// Creates a portal. Portals provide a way to render children into a
DOM node that exists outside
// the hierarchy of the DOM component.
ReactDOM.createPortal(child, container)


/*
*******************************************************************
***********************
 * REACTDOMSERVER
 * The ReactDOMServer object enables you to render components to
static markup.
 * https://reactjs.org/docs/react-dom.html
 *
*******************************************************************
*********************** */


// Render a React element to its initial HTML. React will return an
HTML string.
// You can use this method to generate HTML on the server and send
the markup down on the initial
// request for faster page loads and to allow search engines to
crawl your pages for SEO purposes.
ReactDOMServer.renderToString(element)

// Similar to renderToString, except this doesn't create extra DOM
attributes that React uses
// internally, such as data-reactroot. This is useful if you want to
use React as a simple static
// page generator, as stripping away the extra attributes can save
some bytes.
ReactDOMServer.renderToStaticMarkup(element)

// Render a React element to its initial HTML. Returns a Readable
stream that outputs an HTML string.
// The HTML output by this stream is exactly equal to what
ReactDOMServer.renderToString would return.
// You can use this method to generate HTML on the server and send
the markup down on the initial
// request for faster page loads and to allow search engines to
crawl your pages for SEO purposes.
```

```
ReactDOMServer.renderToNodeStream(element)

// Similar to renderToNodeStream, except this doesn’t create extra
DOM attributes that React uses
// internally, such as data-reactroot. This is useful if you want to
use React as a simple static
// page generator, as stripping away the extra attributes can save
some bytes.
ReactDOMServer.renderToStaticNodeStream(element)


/*
*********************************************************************
***********************
 * TYPECHECKING WITH PROPTYPES
 * https://reactjs.org/docs/typechecking-with-proptypes.html
 *
*********************************************************************
*********************** */


import PropTypes from 'prop-types';

MyComponent.propTypes = {
  // You can declare that a prop is a specific JS type. By default,
these
  // are all optional.
  optionalArray: PropTypes.array,
  optionalBool: PropTypes.bool,
  optionalFunc: PropTypes.func,
  optionalNumber: PropTypes.number,
  optionalObject: PropTypes.object,
  optionalString: PropTypes.string,
  optionalSymbol: PropTypes.symbol,

  // Anything that can be rendered: numbers, strings, elements or an
array
  // (or fragment) containing these types.
  optionalNode: PropTypes.node,

  // A React element.
  optionalElement: PropTypes.element,

  // You can also declare that a prop is an instance of a class.
This uses
  // JS's instanceof operator.
  optionalMessage: PropTypes.instanceOf(Message),

  // You can ensure that your prop is limited to specific values by
treating
  // it as an enum.
  optionalEnum: PropTypes.oneOf(['News', 'Photos']),

  // An object that could be one of many types
  optionalUnion: PropTypes.oneOfType([
    PropTypes.string,
    PropTypes.number,
    PropTypes.instanceOf(Message)
  ]),

  // An array of a certain type
  optionalArrayOf: PropTypes.arrayOf(PropTypes.number),

  // An object with property values of a certain type
  optionalObjectOf: PropTypes.objectOf(PropTypes.number),

  // An object taking on a particular shape
  optionalObjectWithShape: PropTypes.shape({
    color: PropTypes.string,
    fontSize: PropTypes.number
  }),

  // You can chain any of the above with `isRequired` to make sure a
warning
  // is shown if the prop isn't provided.
  requiredFunc: PropTypes.func.isRequired,

  // A value of any data type
  requiredAny: PropTypes.any.isRequired,

  // You can also specify a custom validator. It should return an
Error
  // object if the validation fails. Don't `console.warn` or throw,
as this
  // won't work inside `oneOfType`.
  customProp: function(props, propName, componentName) {
    if (!/matchme/.test(props[propName])) {
      return new Error(
        'Invalid prop `' + propName + '` supplied to' +
        ' `' + componentName + '`. Validation failed.'
      );
    }
  },

  // You can also supply a custom validator to `arrayOf` and
`objectOf`.
  // It should return an Error object if the validation fails. The
validator
  // will be called for each key in the array or object. The first
two
  // arguments of the validator are the array or object itself, and
the
  // current item's key.
  customArrayProp: PropTypes.arrayOf(function(propValue, key,
componentName, location, propFullName) {
    if (!/matchme/.test(propValue[key])) {
      return new Error(
        'Invalid prop `' + propFullName + '` supplied to' +
        ' `' + componentName + '`. Validation failed.'
      );
    }
  })
};



-------------------------------------------------------------------
---




/*
*******************************************************************
***********************
 * GLOBAL OBJECTS > OBJECT
 * https://developer.mozilla.org/en-US/docs/Web/JavaScript/
Reference/Global_Objects/Object
 *
*******************************************************************
*********************** */

// Global object: properties
Object.length                                       // length is a
property of a function object, and indicates how many arguments the
function expects, i.e. the number of formal parameters. This number
does not include the rest parameter. Has a value of 1.
Object.prototype                                    // Represents
the Object prototype object and allows to add new properties and
methods to all objects of type Object.

// Methods of the Object constructor
Object.assign(target, ...sources)                   // Copies the
values of all enumerable own properties from one or more source
objects to a target object. method is used to copy the values of all
enumerable own properties from one or more source objects to a
target object. It will return the target object
Object.create(MyObject)                             // Creates a
new object with the specified prototype object and properties. The
object which should be the prototype of the newly-created object.
Object.defineProperty(obj, prop, descriptor)        // Adds the
named property described by a given descriptor to an object.
Object.defineProperties(obj, props)                 // Adds the
named properties described by the given descriptors to an object.
Object.entries(obj)                                 // Returns an
array containing all of the [key, value] pairs of a given object's
own enumerable string properties.
Object.freeze(obj)                                  // Freezes an
object: other code can't delete or change any properties.
Object.getOwnPropertyDescriptor(obj, prop)          // Returns a
property descriptor for a named property on an object.
```

```
Object.getOwnPropertyDescriptors(obj)               // Returns an
object containing all own property descriptors for an object.
Object.getOwnPropertyNames(obj)                     // Returns an
array containing the names of all of the given object's own
enumerable and non-enumerable properties.
Object.getOwnPropertySymbols(obj)                   // Returns an
array of all symbol properties found directly upon a given object.
Object.getPrototypeOf(obj)                          // Returns the
prototype of the specified object.
Object.is(value1, value2);                          // Compares if
two values are the same value. Equates all NaN values (which differs
from both Abstract Equality Comparison and Strict Equality
Comparison).
Object.isExtensible(obj)                            // Determines
if extending of an object is allowed.
Object.isFrozen(obj)                                // Determines
if an object was frozen.
Object.isSealed(obj)                                // Determines
if an object is sealed.
Object.keys(obj)                                    // Returns an
array containing the names of all of the given object's own
enumerable string properties.
Object.preventExtensions(obj)                       // Prevents any
extensions of an object.
Object.seal(obj)                                    // Prevents
other code from deleting properties of an object.
Object.setPrototypeOf(obj, prototype)               // Sets the
prototype (i.e., the internal [[Prototype]] property).
Object.values(obj)                                  // Returns an
array containing the values that correspond to all of a given
object's own enumerable string properties.

// Object instances and Object prototype object
(Object.prototype.property or Object.prototype.method())
// Properties
obj.constructor                                     // Specifies
the function that creates an object's prototype.
obj.__proto__                                       // Points to
the object which was used as prototype when the object was
instantiated.

// Methods
obj.hasOwnProperty(prop)                            // Returns a
boolean indicating whether an object contains the specified property
as a direct property of that object and not inherited through the
prototype chain.
prototypeObj.isPrototypeOf(object)                  // Returns a
boolean indicating whether the object this method is called upon is
in the prototype chain of the specified object.
obj.propertyIsEnumerable(prop)                      // Returns a
boolean indicating if the internal ECMAScript [[Enumerable]]
attribute is set.
obj.toLocaleString()                                // Calls
toString().
```

```
obj.toString()                                      // Returns a
string representation of the object.
object.valueOf()                                    // Returns the
primitive value of the specified object.

/*
********************************************************************
***********************
 * GLOBAL OBJECTS > ARRAY
 * https://developer.mozilla.org/en-US/docs/Web/JavaScript/
Reference/Global_Objects/Array
 *
********************************************************************
*********************** */

// Global object: properties
Array.length                                        // Reflects the
number of elements in an array.
Array.prototype                                     // Represents
the prototype for the Array constructor and allows to add new
properties and methods to all Array objects.

// Global object: methods
Array.from(arrayLike[, mapFn[, thisArg]])           // Creates a
new Array instance from an array-like or iterable object.
Array.isArray(obj)                                  // Returns true
if a variable is an array, if not false.
Array.of(element0[, element1[, ...[, elementN]]])    // Creates a
new Array instance with a variable number of arguments, regardless
of number or type of the arguments.

// Instance: properties
arr.length                                          // Reflects the
number of elements in an array.

// Instance: mutator methods
arr.copyWithin(target, start, end)                  // Copies a
sequence of array elements within the array.
arr.fill(value, start, end)                         // Fills all
the elements of an array from a start index to an end index with a
static value.
arr.pop()                                           // Removes the
last element from an array and returns that element.
arr.flat()                                          // merges
nested array into one single array
arr.push([element1[, ...[, elementN]]])             // Adds one or
more elements to the end of an array and returns the new length of
the array.
arr.reverse()                                       // Reverses the
order of the elements of an array in place — the first becomes the
last, and the last becomes the first.
arr.shift()                                         // Removes the
first element from an array and returns that element.
arr.sort()                                          // Sorts the
```

```
elements of an array in place and returns the array.
array.splice(start, deleteCount, item1, item2, ...)  // Adds and/or
removes elements from an array.
arr.unshift([element1[, ...[, elementN]]])           // Adds one or
more elements to the front of an array and returns the new length of
the array.

// Instance: accessor methods
arr.concat(value1[, value2[, ...[, valueN]]])        // Returns a
new array comprised of this array joined with other array(s) and/or
value(s).
arr.includes(searchElement, fromIndex)               // Determines
whether an array contains a certain element, returning true or false
as appropriate.
arr.indexOf(searchElement[, fromIndex])              // Returns the
first (least) index of an element within the array equal to the
specified value, or -1 if none is found.
arr.join(separator)                                  // Joins all
elements of an array into a string.
arr.lastIndexOf(searchElement, fromIndex)            // Returns the
last (greatest) index of an element within the array equal to the
specified value, or -1 if none is found.
arr.slice(begin, end)                                // Extracts a
section of an array and returns a new array.
arr.toString()                                       // Returns a
string representing the array and its elements. Overrides the
Object.prototype.toString() method.
arr.toLocaleString(locales, options)                 // Returns a
localized string representing the array and its elements. Overrides
the Object.prototype.toLocaleString() method.

// Instance: iteration methods
arr.entries()                                        // Returns a
new Array Iterator object that contains the key/value pairs for each
index in the array.
arr.every(callback[, thisArg])                       // Returns true
if every element in this array satisfies the provided testing
function.
arr.filter(callback[, thisArg])                      // Creates a
new array with all of the elements of this array for which the
provided filtering function returns true.
arr.find(callback[, thisArg])                        // Returns the
found value in the array, if an element in the array satisfies the
provided testing function or undefined if not found.
arr.findIndex(callback[, thisArg])                   // Returns the
found index in the array, if an element in the array satisfies the
provided testing function or -1 if not found.
arr.forEach(callback[, thisArg])                     // Calls a
function for each element in the array.
arr.keys()                                           // Returns a
new Array Iterator that contains the keys for each index in the
array.
arr.map(callback[, initialValue])                    // Creates a
new array with the results of calling a provided function on every
```

```
element in this array.
arr.reduce(callback[, initialValue])                 // Apply a
function against an accumulator and each value of the array (from
left-to-right) as to reduce it to a single value.
arr.reduceRight(callback[, initialValue])            // Apply a
function against an accumulator and each value of the array (from
right-to-left) as to reduce it to a single value.
arr.some(callback[, initialValue])                   // Returns true
if at least one element in this array satisfies the provided testing
function.
arr.values()                                         // Returns a
new Array Iterator object that contains the values for each index in
the array.



--------------------------------------------------------------------
--------------------------------------------------------



/*
==============================================
CSS3 ANIMATION CHEAT SHEET
==============================================

Made by Justin Aguilar

www.justinaguilar.com/animations/

Questions, comments, concerns, love letters:
justin@justinaguilar.com
==============================================
*/

/*
==============================================
slideDown
==============================================
*/


.slideDown{
	animation-name: slideDown;
	-webkit-animation-name: slideDown;

	animation-duration: 1s;
	-webkit-animation-duration: 1s;

	animation-timing-function: ease;
	-webkit-animation-timing-function: ease;

	visibility: visible !important;
```

```
}

@keyframes slideDown {
        0% {
                transform: translateY(-100%);
        }
        50%{
                transform: translateY(8%);
        }
        65%{
                transform: translateY(-4%);
        }
        80%{
                transform: translateY(4%);
        }
        95%{
                transform: translateY(-2%);
        }
        100% {
                transform: translateY(0%);
        }
}

@-webkit-keyframes slideDown {
        0% {
                -webkit-transform: translateY(-100%);
        }
        50%{
                -webkit-transform: translateY(8%);
        }
        65%{
                -webkit-transform: translateY(-4%);
        }
        80%{
                -webkit-transform: translateY(4%);
        }
        95%{
                -webkit-transform: translateY(-2%);
        }
        100% {
                -webkit-transform: translateY(0%);
        }
}

/*
=============================================
slideUp
=============================================
*/


.slideUp{
        animation-name: slideUp;
        -webkit-animation-name: slideUp;
```

```
}


        animation-duration: 1s;
        -webkit-animation-duration: 1s;

        animation-timing-function: ease;
        -webkit-animation-timing-function: ease;

        visibility: visible !important;
}

@keyframes slideUp {
        0% {
                transform: translateY(100%);
        }
        50%{
                transform: translateY(-8%);
        }
        65%{
                transform: translateY(4%);
        }
        80%{
                transform: translateY(-4%);
        }
        95%{
                transform: translateY(2%);
        }
        100% {
                transform: translateY(0%);
        }
}

@-webkit-keyframes slideUp {
        0% {
                -webkit-transform: translateY(100%);
        }
        50%{
                -webkit-transform: translateY(-8%);
        }
        65%{
                -webkit-transform: translateY(4%);
        }
        80%{
                -webkit-transform: translateY(-4%);
        }
        95%{
                -webkit-transform: translateY(2%);
        }
        100% {
                -webkit-transform: translateY(0%);
        }
}

/*
==============================================
slideLeft
==============================================
*/


.slideLeft{
        animation-name: slideLeft;
        -webkit-animation-name: slideLeft;

        animation-duration: 1s;
        -webkit-animation-duration: 1s;

        animation-timing-function: ease-in-out;
        -webkit-animation-timing-function: ease-in-out;

        visibility: visible !important;
}

@keyframes slideLeft {
        0% {
                transform: translateX(150%);
        }
        50%{
                transform: translateX(-8%);
        }
        65%{
                transform: translateX(4%);
        }
        80%{
                transform: translateX(-4%);
        }
        95%{
                transform: translateX(2%);
        }
        100% {
                transform: translateX(0%);
        }
}

@-webkit-keyframes slideLeft {
        0% {
                -webkit-transform: translateX(150%);
        }
        50%{
                -webkit-transform: translateX(-8%);
        }
        65%{
                -webkit-transform: translateX(4%);
        }
        80%{
                -webkit-transform: translateX(-4%);
        }
        95%{
                -webkit-transform: translateX(2%);
        }
        100% {
                -webkit-transform: translateX(0%);
        }
}

/*
==============================================
slideRight
==============================================
*/


.slideRight{
        animation-name: slideRight;
        -webkit-animation-name: slideRight;

        animation-duration: 1s;
        -webkit-animation-duration: 1s;

        animation-timing-function: ease-in-out;
        -webkit-animation-timing-function: ease-in-out;

        visibility: visible !important;
}

@keyframes slideRight {
        0% {
                transform: translateX(-150%);
        }
        50%{
                transform: translateX(8%);
        }
        65%{
                transform: translateX(-4%);
        }
        80%{
                transform: translateX(4%);
        }
        95%{
                transform: translateX(-2%);
        }
        100% {
                transform: translateX(0%);
        }
}

@-webkit-keyframes slideRight {
        0% {
                -webkit-transform: translateX(-150%);
        }
        50%{
                -webkit-transform: translateX(8%);
        }
        65%{
                -webkit-transform: translateX(-4%);
        }
        80%{
                -webkit-transform: translateX(4%);
        }
        95%{
                -webkit-transform: translateX(-2%);
        }
        100% {
                -webkit-transform: translateX(0%);
        }
}

/*
==============================================
slideExpandUp
==============================================
*/


.slideExpandUp{
        animation-name: slideExpandUp;
        -webkit-animation-name: slideExpandUp;

        animation-duration: 1.6s;
        -webkit-animation-duration: 1.6s;

        animation-timing-function: ease-out;
        -webkit-animation-timing-function: ease -out;

        visibility: visible !important;
}

@keyframes slideExpandUp {
        0% {
                transform: translateY(100%) scaleX(0.5);
        }
        30%{
                transform: translateY(-8%) scaleX(0.5);
        }
        40%{
                transform: translateY(2%) scaleX(0.5);
        }
        50%{
                transform: translateY(0%) scaleX(1.1);
        }
        60%{
                transform: translateY(0%) scaleX(0.9);
        }
        70% {
                transform: translateY(0%) scaleX(1.05);
        }
        80%{
```

```
			transform: translateY(0%) scaleX(0.95);
		}
		90% {
			transform: translateY(0%) scaleX(1.02);
		}
		100%{
			transform: translateY(0%) scaleX(1);
		}
}

@-webkit-keyframes slideExpandUp {
		0% {
			-webkit-transform: translateY(100%) scaleX(0.5);
		}
		30%{
			-webkit-transform: translateY(-8%) scaleX(0.5);
		}
		40%{
			-webkit-transform: translateY(2%) scaleX(0.5);
		}
		50%{
			-webkit-transform: translateY(0%) scaleX(1.1);
		}
		60%{
			-webkit-transform: translateY(0%) scaleX(0.9);

		}
		70% {
			-webkit-transform: translateY(0%) scaleX(1.05);
		}
		80%{
			-webkit-transform: translateY(0%) scaleX(0.95);

		}
		90% {
			-webkit-transform: translateY(0%) scaleX(1.02);
		}
		100%{
			-webkit-transform: translateY(0%) scaleX(1);

		}
}

/*
==============================================
expandUp
==============================================
*/


.expandUp{
		animation-name: expandUp;
		-webkit-animation-name: expandUp;
```

```
        animation-duration: 0.7s;
        -webkit-animation-duration: 0.7s;

        animation-timing-function: ease;
        -webkit-animation-timing-function: ease;

        visibility: visible !important;
}

@keyframes expandUp {
        0% {
                transform: translateY(100%) scale(0.6) scaleY(0.5);
        }
        60%{
                transform: translateY(-7%) scaleY(1.12);
        }
        75%{
                transform: translateY(3%);
        }
        100% {
                transform: translateY(0%) scale(1) scaleY(1);
        }
}

@-webkit-keyframes expandUp {
        0% {
                -webkit-transform: translateY(100%) scale(0.6)
scaleY(0.5);
        }
        60%{
                -webkit-transform: translateY(-7%) scaleY(1.12);
        }
        75%{
                -webkit-transform: translateY(3%);
        }
        100% {
                -webkit-transform: translateY(0%) scale(1)
scaleY(1);
        }
}

/*
==============================================
fadeIn
==============================================
*/

.fadeIn{
        animation-name: fadeIn;
        -webkit-animation-name: fadeIn;

        animation-duration: 1.5s;
        -webkit-animation-duration: 1.5s;
```

```css
        animation-timing-function: ease-in-out;
        -webkit-animation-timing-function: ease-in-out;

        visibility: visible !important;
}

@keyframes fadeIn {
        0% {
                transform: scale(0);
                opacity: 0.0;
        }
        60% {
                transform: scale(1.1);
        }
        80% {
                transform: scale(0.9);
                opacity: 1;
        }
        100% {
                transform: scale(1);
                opacity: 1;
        }
}

@-webkit-keyframes fadeIn {
        0% {
                -webkit-transform: scale(0);
                opacity: 0.0;
        }
        60% {
                -webkit-transform: scale(1.1);
        }
        80% {
                -webkit-transform: scale(0.9);
                opacity: 1;
        }
        100% {
                -webkit-transform: scale(1);
                opacity: 1;
        }
}

/*
==============================================
expandOpen
==============================================
*/


.expandOpen{
        animation-name: expandOpen;
        -webkit-animation-name: expandOpen;

        animation-duration: 1.2s;
        -webkit-animation-duration: 1.2s;

        animation-timing-function: ease-out;
        -webkit-animation-timing-function: ease-out;

        visibility: visible !important;
}

@keyframes expandOpen {
        0% {
                transform: scale(1.8);
        }
        50% {
                transform: scale(0.95);
        }
        80% {
                transform: scale(1.05);
        }
        90% {
                transform: scale(0.98);
        }
        100% {
                transform: scale(1);
        }
}

@-webkit-keyframes expandOpen {
        0% {
                -webkit-transform: scale(1.8);
        }
        50% {
                -webkit-transform: scale(0.95);
        }
        80% {
                -webkit-transform: scale(1.05);
        }
        90% {
                -webkit-transform: scale(0.98);
        }
        100% {
                -webkit-transform: scale(1);
        }
}

/*
==============================================
bigEntrance
==============================================
*/


.bigEntrance{
        animation-name: bigEntrance;
        -webkit-animation-name: bigEntrance;
```

```
	animation-duration: 1.6s;
	-webkit-animation-duration: 1.6s;

	animation-timing-function: ease-out;
	-webkit-animation-timing-function: ease-out;

	visibility: visible !important;
}

@keyframes bigEntrance {
	0% {
		transform: scale(0.3) rotate(6deg) translateX(-30%)
translateY(30%);
		opacity: 0.2;
	}
	30% {
		transform: scale(1.03) rotate(-2deg) translateX(2%)
translateY(-2%);
		opacity: 1;
	}
	45% {
		transform: scale(0.98) rotate(1deg) translateX(0%)
translateY(0%);
		opacity: 1;
	}
	60% {
		transform: scale(1.01) rotate(-1deg) translateX(0%)
translateY(0%);
		opacity: 1;
	}
	75% {
		transform: scale(0.99) rotate(1deg) translateX(0%)
translateY(0%);
		opacity: 1;
	}
	90% {
		transform: scale(1.01) rotate(0deg) translateX(0%)
translateY(0%);
		opacity: 1;
	}
	100% {
		transform: scale(1) rotate(0deg) translateX(0%)
translateY(0%);
		opacity: 1;
	}
}

@-webkit-keyframes bigEntrance {
	0% {
		-webkit-transform: scale(0.3) rotate(6deg)
translateX(-30%) translateY(30%);
		opacity: 0.2;
	}
```

```
        30% {
                -webkit-transform: scale(1.03) rotate(-2deg)
translateX(2%) translateY(-2%);
                opacity: 1;
        }
        45% {
                -webkit-transform: scale(0.98) rotate(1deg)
translateX(0%) translateY(0%);
                opacity: 1;
        }
        60% {
                -webkit-transform: scale(1.01) rotate(-1deg)
translateX(0%) translateY(0%);
                opacity: 1;
        }
        75% {
                -webkit-transform: scale(0.99) rotate(1deg)
translateX(0%) translateY(0%);
                opacity: 1;
        }
        90% {
                -webkit-transform: scale(1.01) rotate(0deg)
translateX(0%) translateY(0%);
                opacity: 1;
        }
        100% {
                -webkit-transform: scale(1) rotate(0deg)
translateX(0%) translateY(0%);
                opacity: 1;
        }
}

/*
==============================================
hatch
==============================================
*/

.hatch{
        animation-name: hatch;
        -webkit-animation-name: hatch;

        animation-duration: 2s;
        -webkit-animation-duration: 2s;

        animation-timing-function: ease-in-out;
        -webkit-animation-timing-function: ease-in-out;

        transform-origin: 50% 100%;
        -ms-transform-origin: 50% 100%;
        -webkit-transform-origin: 50% 100%;

        visibility: visible !important;
}
```

```
}



@keyframes hatch {
        0% {
                transform: rotate(0deg) scaleY(0.6);
        }
        20% {
                transform: rotate(-2deg) scaleY(1.05);
        }
        35% {
                transform: rotate(2deg) scaleY(1);
        }
        50% {
                transform: rotate(-2deg);
        }
        65% {
                transform: rotate(1deg);
        }
        80% {
                transform: rotate(-1deg);
        }
        100% {
                transform: rotate(0deg);
        }

}

@-webkit-keyframes hatch {
        0% {
                -webkit-transform: rotate(0deg) scaleY(0.6);
        }
        20% {
                -webkit-transform: rotate(-2deg) scaleY(1.05);
        }
        35% {
                -webkit-transform: rotate(2deg) scaleY(1);
        }
        50% {
                -webkit-transform: rotate(-2deg);
        }
        65% {
                -webkit-transform: rotate(1deg);
        }
        80% {
                -webkit-transform: rotate(-1deg);
        }
        100% {
                -webkit-transform: rotate(0deg);
        }
}


/*
==============================================
bounce
```

```
==============================================
*/


.bounce{
        animation-name: bounce;
        -webkit-animation-name: bounce;

        animation-duration: 1.6s;
        -webkit-animation-duration: 1.6s;

        animation-timing-function: ease;
        -webkit-animation-timing-function: ease;

        transform-origin: 50% 100%;
        -ms-transform-origin: 50% 100%;
        -webkit-transform-origin: 50% 100%;
}

@keyframes bounce {
        0% {
                transform: translateY(0%) scaleY(0.6);
        }
        60%{
                transform: translateY(-100%) scaleY(1.1);
        }
        70%{
                transform: translateY(0%) scaleY(0.95)
scaleX(1.05);
        }
        80%{
                transform: translateY(0%) scaleY(1.05) scaleX(1);
        }
        90%{
                transform: translateY(0%) scaleY(0.95) scaleX(1);
        }
        100%{
                transform: translateY(0%) scaleY(1) scaleX(1);
        }
}

@-webkit-keyframes bounce {
        0% {
                -webkit-transform: translateY(0%) scaleY(0.6);
        }
        60%{
                -webkit-transform: translateY(-100%) scaleY(1.1);
        }
        70%{
                -webkit-transform: translateY(0%) scaleY(0.95)
scaleX(1.05);
        }
        80%{
                -webkit-transform: translateY(0%) scaleY(1.05)
scaleX(1);
        }
        90%{
                -webkit-transform: translateY(0%) scaleY(0.95)
scaleX(1);
        }
        100%{
                -webkit-transform: translateY(0%) scaleY(1)
scaleX(1);
        }
}


/*
=============================================
pulse
=============================================
*/

.pulse{
        animation-name: pulse;
        -webkit-animation-name: pulse;

        animation-duration: 1.5s;
        -webkit-animation-duration: 1.5s;

        animation-iteration-count: infinite;
        -webkit-animation-iteration-count: infinite;
}

@keyframes pulse {
        0% {
                transform: scale(0.9);
                opacity: 0.7;
        }
        50% {
                transform: scale(1);
                opacity: 1;
        }
        100% {
                transform: scale(0.9);
                opacity: 0.7;
        }
}

@-webkit-keyframes pulse {
        0% {
                -webkit-transform: scale(0.95);
                opacity: 0.7;
        }
        50% {
                -webkit-transform: scale(1);
                opacity: 1;
        }
        100% {
                -webkit-transform: scale(0.95);
                opacity: 0.7;
        }
}

/*
=============================================
floating
=============================================
*/

.floating{
        animation-name: floating;
        -webkit-animation-name: floating;

        animation-duration: 1.5s;
        -webkit-animation-duration: 1.5s;

        animation-iteration-count: infinite;
        -webkit-animation-iteration-count: infinite;
}

@keyframes floating {
        0% {
                transform: translateY(0%);
        }
        50% {
                transform: translateY(8%);
        }
        100% {
                transform: translateY(0%);
        }
}

@-webkit-keyframes floating {
        0% {
                -webkit-transform: translateY(0%);
        }
        50% {
                -webkit-transform: translateY(8%);
        }
        100% {
                -webkit-transform: translateY(0%);
        }
}

/*
=============================================
tossing
=============================================
*/

.tossing{
        animation-name: tossing;
        -webkit-animation-name: tossing;

        animation-duration: 2.5s;
        -webkit-animation-duration: 2.5s;

        animation-iteration-count: infinite;
        -webkit-animation-iteration-count: infinite;
}

@keyframes tossing {
        0% {
                transform: rotate(-4deg);
        }
        50% {
                transform: rotate(4deg);
        }
        100% {
                transform: rotate(-4deg);
        }
}

@-webkit-keyframes tossing {
        0% {
                -webkit-transform: rotate(-4deg);
        }
        50% {
                -webkit-transform: rotate(4deg);
        }
        100% {
                -webkit-transform: rotate(-4deg);
        }
}

/*
==============================================
pullUp
==============================================
*/

.pullUp{
        animation-name: pullUp;
        -webkit-animation-name: pullUp;

        animation-duration: 1.1s;
        -webkit-animation-duration: 1.1s;

        animation-timing-function: ease-out;
        -webkit-animation-timing-function: ease-out;

        transform-origin: 50% 100%;
        -ms-transform-origin: 50% 100%;
        -webkit-transform-origin: 50% 100%;
}
```

```
@keyframes pullUp {
        0% {
                transform: scaleY(0.1);
        }
        40% {
                transform: scaleY(1.02);
        }
        60% {
                transform: scaleY(0.98);
        }
        80% {
                transform: scaleY(1.01);
        }
        100% {
                transform: scaleY(0.98);
        }
        80% {
                transform: scaleY(1.01);
        }
        100% {
                transform: scaleY(1);
        }
}

@-webkit-keyframes pullUp {
        0% {
                -webkit-transform: scaleY(0.1);
        }
        40% {
                -webkit-transform: scaleY(1.02);
        }
        60% {
                -webkit-transform: scaleY(0.98);
        }
        80% {
                -webkit-transform: scaleY(1.01);
        }
        100% {
                -webkit-transform: scaleY(0.98);
        }
        80% {
                -webkit-transform: scaleY(1.01);
        }
        100% {
                -webkit-transform: scaleY(1);
        }
}

/*
==============================================
pullDown
==============================================
*/
```

```
.pullDown{
        animation-name: pullDown;
        -webkit-animation-name: pullDown;

        animation-duration: 1.1s;
        -webkit-animation-duration: 1.1s;

        animation-timing-function: ease-out;
        -webkit-animation-timing-function: ease-out;

        transform-origin: 50% 0%;
        -ms-transform-origin: 50% 0%;
        -webkit-transform-origin: 50% 0%;
}

@keyframes pullDown {
        0% {
                transform: scaleY(0.1);
        }
        40% {
                transform: scaleY(1.02);
        }
        60% {
                transform: scaleY(0.98);
        }
        80% {
                transform: scaleY(1.01);
        }
        100% {
                transform: scaleY(0.98);
        }
        80% {
                transform: scaleY(1.01);
        }
        100% {
                transform: scaleY(1);
        }
}

@-webkit-keyframes pullDown {
        0% {
                -webkit-transform: scaleY(0.1);
        }
        40% {
                -webkit-transform: scaleY(1.02);
        }
        60% {
                -webkit-transform: scaleY(0.98);
        }
        80% {
                -webkit-transform: scaleY(1.01);
        }
        100% {
                -webkit-transform: scaleY(0.98);
        }
        80% {
                -webkit-transform: scaleY(1.01);
        }
        100% {
                -webkit-transform: scaleY(1);
        }
}

/*
==============================================
stretchLeft
==============================================
*/

.stretchLeft{
        animation-name: stretchLeft;
        -webkit-animation-name: stretchLeft;

        animation-duration: 1.5s;
        -webkit-animation-duration: 1.5s;

        animation-timing-function: ease-out;
        -webkit-animation-timing-function: ease-out;

        transform-origin: 100% 0%;
        -ms-transform-origin: 100% 0%;
        -webkit-transform-origin: 100% 0%;
}

@keyframes stretchLeft {
        0% {
                transform: scaleX(0.3);
        }
        40% {
                transform: scaleX(1.02);
        }
        60% {
                transform: scaleX(0.98);
        }
        80% {
                transform: scaleX(1.01);
        }
        100% {
                transform: scaleX(0.98);
        }
        80% {
                transform: scaleX(1.01);
        }
        100% {
                transform: scaleX(1);
        }
}
```

```
@-webkit-keyframes stretchLeft {
        0% {
                -webkit-transform: scaleX(0.3);
        }
        40% {
                -webkit-transform: scaleX(1.02);
        }
        60% {
                -webkit-transform: scaleX(0.98);
        }
        80% {
                -webkit-transform: scaleX(1.01);
        }
        100% {
                -webkit-transform: scaleX(0.98);
        }
        80% {
                -webkit-transform: scaleX(1.01);
        }
        100% {
                -webkit-transform: scaleX(1);
        }
}

/*
==============================================
stretchRight
==============================================
*/

.stretchRight{
        animation-name: stretchRight;
        -webkit-animation-name: stretchRight;

        animation-duration: 1.5s;
        -webkit-animation-duration: 1.5s;

        animation-timing-function: ease-out;
        -webkit-animation-timing-function: ease-out;

        transform-origin: 0% 0%;
        -ms-transform-origin: 0% 0%;
        -webkit-transform-origin: 0% 0%;
}

@keyframes stretchRight {
        0% {
                transform: scaleX(0.3);
        }
        40% {
                transform: scaleX(1.02);
        }
        60% {
                transform: scaleX(0.98);
        }
        80% {
                transform: scaleX(1.01);
        }
        100% {
                transform: scaleX(0.98);
        }
        80% {
                transform: scaleX(1.01);
        }
        100% {
                transform: scaleX(1);
        }
}

@-webkit-keyframes stretchRight {
        0% {
                -webkit-transform: scaleX(0.3);
        }
        40% {
                -webkit-transform: scaleX(1.02);
        }
        60% {
                -webkit-transform: scaleX(0.98);
        }
        80% {
                -webkit-transform: scaleX(1.01);
        }
        100% {
                -webkit-transform: scaleX(0.98);
        }
        80% {
                -webkit-transform: scaleX(1.01);
        }
        100% {
                -webkit-transform: scaleX(1);
        }
}

//------------------------------------------------------------------
--------------------------------------------------------------------
------------------------
```