Final Code Deliverables

Java Script :- Scrollreveal.js

```
/*! @license ScrollReveal v4.0.9

    Copyright 2021 Fisssion LLC.

    Licensed under the GNU General Public License 3.0 for
    compatible open source projects and non-commercial use.

    For commercial sites, themes, projects, and applications,
    keep your source code private/proprietary by purchasing
    a commercial license from https://scrollrevealjs.org/
*/
(function (global, factory) {
    typeof exports === 'object' && typeof module !== 'undefined' ?
module.exports = factory() :
    typeof define === 'function' && define.amd ? define(factory) :
    (global = global || self, global.ScrollReveal = factory());
}(this, function () { 'use strict';

    var defaults = {
        delay: 0,
        distance: '0',
        duration: 600,
        easing: 'cubic-bezier(0.5, 0, 0, 1)',
        interval: 0,
        opacity: 0,
        origin: 'bottom',
        rotate: {
            x: 0,
            y: 0,
            z: 0
        },
        scale: 1,
        cleanup: false,
        container: document.documentElement,
        desktop: true,
        mobile: true,
        reset: false,
        useDelay: 'always',
        viewFactor: 0.0,
        viewOffset: {
            top: 0,
            right: 0,
            bottom: 0,
            left: 0
        },
        afterReset: function afterReset() {},
        afterReveal: function afterReveal() {},
        beforeReset: function beforeReset() {},
        beforeReveal: function beforeReveal() {}
    };

    function failure() {
        document.documentElement.classList.remove('sr');

        return {
            clean: function clean() {},
            destroy: function destroy() {},
            reveal: function reveal() {},
            sync: function sync() {},
            get noop() {
                return true
            }
        }
    }

    function success() {
        document.documentElement.classList.add('sr');

        if (document.body) {
            document.body.style.height = '100%';
        } else {
            document.addEventListener('DOMContentLoaded', function () {
                document.body.style.height = '100%';
            });
        }
    }

    var mount = { success: success, failure: failure };

    /*! @license is-dom-node v1.0.4

        Copyright 2018 Fisssion LLC.

        Permission is hereby granted, free of charge, to any person obtaining a copy
        of this software and associated documentation files (the "Software"), to deal
        in the Software without restriction, including without limitation the
rights
        to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
        copies of the Software, and to permit persons to whom the Software is
        furnished to do so, subject to the following conditions:

        The above copyright notice and this permission notice shall be
included in all
        copies or substantial portions of the Software.

        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
        IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
        FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
        AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
        LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
        OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE
        SOFTWARE.

    */
    function isDomNode(x) {
        return typeof window.Node === 'object'
            ? x instanceof window.Node
            : x !== null &&
                  typeof x === 'object' &&
                  typeof x.nodeType === 'number' &&
                  typeof x.nodeName === 'string'
    }

    /*! @license is-dom-node-list v1.2.1

        Copyright 2018 Fisssion LLC.

        Permission is hereby granted, free of charge, to any person obtaining
a copy
        of this software and associated documentation files (the "Software"),
to deal
        in the Software without restriction, including without limitation the
rights
        to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
        copies of the Software, and to permit persons to whom the Software is
        furnished to do so, subject to the following conditions:
```

```
        The above copyright notice and this permission notice shall be
included in all
        copies or substantial portions of the Software.

        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
        IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
        FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
        AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
        LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
        OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE
        SOFTWARE.

    */

    function isDomNodeList(x) {
        var prototypeToString = Object.prototype.toString.call(x);
        var regex = /^\[object (HTMLCollection|NodeList|Object)\]$/;

        return typeof window.NodeList === 'object'
            ? x instanceof window.NodeList
            : x !== null &&
                    typeof x === 'object' &&
                    typeof x.length === 'number' &&
                    regex.test(prototypeToString) &&
                    (x.length === 0 || isDomNode(x[0]))
    }

    /*! @license Tealight v0.3.6

        Copyright 2018 Fisssion LLC.

        Permission is hereby granted, free of charge, to any person obtaining
a copy
        of this software and associated documentation files (the "Software"),
to deal
        in the Software without restriction, including without limitation the
rights
        to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
        copies of the Software, and to permit persons to whom the Software is
        furnished to do so, subject to the following conditions:
```

```
        The above copyright notice and this permission notice shall be
included in all
        copies or substantial portions of the Software.

        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
        IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
        FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
        AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
        LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
        OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE
        SOFTWARE.

    */

    function tealight(target, context) {
      if ( context === void 0 ) { context = document; }

      if (target instanceof Array) { return target.filter(isDomNode); }
      if (isDomNode(target)) { return [target]; }
      if (isDomNodeList(target)) { return Array.prototype.slice.call(target);
}
      if (typeof target === "string") {
        try {
          var query = context.querySelectorAll(target);
          return Array.prototype.slice.call(query);
        } catch (err) {
          return [];
        }
      }
      return [];
    }

    function isObject(x) {
        return (
            x !== null &&
            x instanceof Object &&
            (x.constructor === Object ||
                Object.prototype.toString.call(x) === '[object Object]')
        )
    }

    function each(collection, callback) {
        if (isObject(collection)) {
            var keys = Object.keys(collection);
            return keys.forEach(function (key) { return
callback(collection[key], key, collection); })
        }
        if (collection instanceof Array) {
            return collection.forEach(function (item, i) { return
callback(item, i, collection); })
        }
        throw new TypeError('Expected either an array or object literal.')
    }

    function logger(message) {
        var details = [], len = arguments.length - 1;
        while ( len-- > 0 ) details[ len ] = arguments[ len + 1 ];

        if (this.constructor.debug && console) {
            var report = "%cScrollReveal: " + message;
            details.forEach(function (detail) { return (report += "\n — " +
detail); });
            console.log(report, 'color: #ea654b;'); // eslint-disable-line no-
console
        }
    }

    function rinse() {
        var this$1 = this;

        var struct = function () { return ({
            active: [],
            stale: []
        }); };

        var elementIds = struct();
        var sequenceIds = struct();
        var containerIds = struct();

        /**
         * Take stock of active element IDs.
         */
        try {
            each(tealight('[data-sr-id]'), function (node) {
                var id = parseInt(node.getAttribute('data-sr-id'));
                elementIds.active.push(id);
            });
        } catch (e) {
            throw e
        }
        /**
```

```
         * Destroy stale elements.
         */
        each(this.store.elements, function (element) {
            if (elementIds.active.indexOf(element.id) === -1) {
                elementIds.stale.push(element.id);
            }
        });

        each(elementIds.stale, function (staleId) { return delete
this$1.store.elements[staleId]; });

        /**
         * Take stock of active container and sequence IDs.
         */
        each(this.store.elements, function (element) {
            if (containerIds.active.indexOf(element.containerId) === -1) {
                containerIds.active.push(element.containerId);
            }
            if (element.hasOwnProperty('sequence')) {
                if (sequenceIds.active.indexOf(element.sequence.id) === -1) {
                    sequenceIds.active.push(element.sequence.id);
                }
            }
        });

        /**
         * Destroy stale containers.
         */
        each(this.store.containers, function (container) {
            if (containerIds.active.indexOf(container.id) === -1) {
                containerIds.stale.push(container.id);
            }
        });

        each(containerIds.stale, function (staleId) {
            var stale = this$1.store.containers[staleId].node;
            stale.removeEventListener('scroll', this$1.delegate);
            stale.removeEventListener('resize', this$1.delegate);
            delete this$1.store.containers[staleId];
        });

        /**
         * Destroy stale sequences.
         */
        each(this.store.sequences, function (sequence) {
            if (sequenceIds.active.indexOf(sequence.id) === -1) {
                sequenceIds.stale.push(sequence.id);
            }
        });

        each(sequenceIds.stale, function (staleId) { return delete
this$1.store.sequences[staleId]; });
    }

    /*! @license Rematrix v0.3.0

        Copyright 2018 Julian Lloyd.

        Permission is hereby granted, free of charge, to any person obtaining
a copy
        of this software and associated documentation files (the "Software"),
to deal
        in the Software without restriction, including without limitation the
rights
        to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
        copies of the Software, and to permit persons to whom the Software is
        furnished to do so, subject to the following conditions:

        The above copyright notice and this permission notice shall be
included in
        all copies or substantial portions of the Software.

        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
        IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
        FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
        AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
        LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
        OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN
        THE SOFTWARE.
    */
    /**
     * @module Rematrix
     */

    /**
     * Transformation matrices in the browser come in two flavors:
     *
     *  - `matrix` using 6 values (short)
     *  - `matrix3d` using 16 values (long)
     *
     * This utility follows this [conversion guide](https://goo.gl/EJlUQ1)
     * to expand short form matrices to their equivalent long form.
     *
     * @param  {array} source - Accepts both short and long form matrices.
     * @return {array}
     */
    function format(source) {
        if (source.constructor !== Array) {
            throw new TypeError('Expected array.')
        }
        if (source.length === 16) {
            return source
        }
        if (source.length === 6) {
            var matrix = identity();
            matrix[0] = source[0];
            matrix[1] = source[1];
            matrix[4] = source[2];
            matrix[5] = source[3];
            matrix[12] = source[4];
            matrix[13] = source[5];
            return matrix
        }
        throw new RangeError('Expected array with either 6 or 16 values.')
    }

    /**
     * Returns a matrix representing no transformation. The product of any
matrix
     * multiplied by the identity matrix will be the original matrix.
     *
     * > **Tip:** Similar to how `5 * 1 === 5`, where `1` is the identity.
     *
     * @return {array}
     */
    function identity() {
        var matrix = [];
        for (var i = 0; i < 16; i++) {
            i % 5 == 0 ? matrix.push(1) : matrix.push(0);
        }
        return matrix
    }

    /**
     * Returns a 4x4 matrix describing the combined transformations
     * of both arguments.
     *
     * > **Note:** Order is very important. For example, rotating 45°
     * along the Z-axis, followed by translating 500 pixels along the
     * Y-axis... is not the same as translating 500 pixels along the
     * Y-axis, followed by rotating 45° along on the Z-axis.
     *
     * @param  {array} m - Accepts both short and long form matrices.
     * @param  {array} x - Accepts both short and long form matrices.
     * @return {array}
     */
    function multiply(m, x) {
        var fm = format(m);
        var fx = format(x);
        var product = [];

        for (var i = 0; i < 4; i++) {
            var row = [fm[i], fm[i + 4], fm[i + 8], fm[i + 12]];
            for (var j = 0; j < 4; j++) {
                var k = j * 4;
                var col = [fx[k], fx[k + 1], fx[k + 2], fx[k + 3]];
                var result =
                    row[0] * col[0] + row[1] * col[1] + row[2] * col[2] +
row[3] * col[3];

                product[i + k] = result;
            }
        }

        return product
    }

    /**
     * Attempts to return a 4x4 matrix describing the CSS transform
     * matrix passed in, but will return the identity matrix as a
     * fallback.
     *
     * > **Tip:** This method is used to convert a CSS matrix (retrieved as a
     * `string` from computed styles) to its equivalent array format.
     *
     * @param  {string} source - `matrix` or `matrix3d` CSS Transform value.
     * @return {array}
     */
    function parse(source) {
        if (typeof source === 'string') {
            var match = source.match(/matrix(3d)?\(([^)]+)\)/);
            if (match) {
                var raw = match[2].split(', ').map(parseFloat);
                return format(raw)
            }
        }
        return identity()
    }

    /**
     * Returns a 4x4 matrix describing X-axis rotation.
     *
     * @param  {number} angle - Measured in degrees.
     * @return {array}
     */
    function rotateX(angle) {
        var theta = Math.PI / 180 * angle;
        var matrix = identity();

        matrix[5] = matrix[10] = Math.cos(theta);
        matrix[6] = matrix[9] = Math.sin(theta);
        matrix[9] *= -1;

        return matrix
    }

    /**
     * Returns a 4x4 matrix describing Y-axis rotation.
     *
     * @param  {number} angle - Measured in degrees.
     * @return {array}
     */
    function rotateY(angle) {
        var theta = Math.PI / 180 * angle;
        var matrix = identity();

        matrix[0] = matrix[10] = Math.cos(theta);
        matrix[2] = matrix[8] = Math.sin(theta);
        matrix[2] *= -1;

        return matrix
    }

    /**
     * Returns a 4x4 matrix describing Z-axis rotation.
     *
     * @param  {number} angle - Measured in degrees.
     * @return {array}
     */
    function rotateZ(angle) {
        var theta = Math.PI / 180 * angle;
        var matrix = identity();

        matrix[0] = matrix[5] = Math.cos(theta);
        matrix[1] = matrix[4] = Math.sin(theta);
        matrix[4] *= -1;

        return matrix
    }

    /**
     * Returns a 4x4 matrix describing 2D scaling. The first argument
     * is used for both X and Y-axis scaling, unless an optional
     * second argument is provided to explicitly define Y-axis scaling.
     *
     * @param  {number} scalar    - Decimal multiplier.
     * @param  {number} [scalarY] - Decimal multiplier.
     * @return {array}
     */
    function scale(scalar, scalarY) {
        var matrix = identity();

        matrix[0] = scalar;
        matrix[5] = typeof scalarY === 'number' ? scalarY : scalar;

        return matrix
    }

    /**
     * Returns a 4x4 matrix describing X-axis translation.
     *
     * @param  {number} distance - Measured in pixels.
     * @return {array}
     */
    function translateX(distance) {
        var matrix = identity();
        matrix[12] = distance;
        return matrix
    }

    /**
     * Returns a 4x4 matrix describing Y-axis translation.
     *
     * @param  {number} distance - Measured in pixels.
     * @return {array}
     */
    function translateY(distance) {
        var matrix = identity();
        matrix[13] = distance;
        return matrix
    }
```

```
    var getPrefixedCssProp = (function () {
        var properties = {};
        var style = document.documentElement.style;

        function getPrefixedCssProperty(name, source) {
            if ( source === void 0 ) source = style;

            if (name && typeof name === 'string') {
                if (properties[name]) {
                    return properties[name]
                }
                if (typeof source[name] === 'string') {
                    return (properties[name] = name)
                }
                if (typeof source[("-webkit-" + name)] === 'string') {
                    return (properties[name] = "-webkit-" + name)
                }
                throw new RangeError(("Unable to find \"" + name + "\" style
property."))
            }
            throw new TypeError('Expected a string.')
        }

        getPrefixedCssProperty.clearCache = function () { return (properties =
{}); };

        return getPrefixedCssProperty
    })();

    function style(element) {
        var computed = window.getComputedStyle(element.node);
        var position = computed.position;
        var config = element.config;

        /**
         * Generate inline styles
         */
        var inline = {};
        var inlineStyle = element.node.getAttribute('style') || '';
        var inlineMatch = inlineStyle.match(/[\w-]+\s*:\s*[^;]+\s*/gi) || [];

        inline.computed = inlineMatch ? inlineMatch.map(function (m) { return
m.trim(); }).join('; ') + ';' : '';

        inline.generated = inlineMatch.some(function (m) { return
m.match(/visibility\s?:\s?visible/i); })
            ? inline.computed
            : inlineMatch.concat( ['visibility: visible']).map(function (m) {
return m.trim(); }).join('; ') + ';';

        /**
         * Generate opacity styles
         */
        var computedOpacity = parseFloat(computed.opacity);
        var configOpacity = !isNaN(parseFloat(config.opacity))
            ? parseFloat(config.opacity)
            : parseFloat(computed.opacity);

        var opacity = {
            computed: computedOpacity !== configOpacity ? ("opacity: " +
computedOpacity + ";") : '',
            generated: computedOpacity !== configOpacity ? ("opacity: " +
configOpacity + ";") : ''
        };

        /**
         * Generate transformation styles
         */
        var transformations = [];

        if (parseFloat(config.distance)) {
            var axis = config.origin === 'top' || config.origin === 'bottom' ?
'Y' : 'X';

            /**
             * Let’s make sure our our pixel distances are negative for top
and left.
             * e.g. { origin: 'top', distance: '25px' } starts at `top: -25px`
in CSS.
             */
            var distance = config.distance;
            if (config.origin === 'top' || config.origin === 'left') {
                distance = /^-/.test(distance) ? distance.substr(1) : ("-" +
distance);
            }

            var ref = distance.match(/(^-?\d+\.?\d?)|(em$|px$|%$)/g);
            var value = ref[0];
            var unit = ref[1];

            switch (unit) {
                case 'em':
                    distance = parseInt(computed.fontSize) * value;
                    break
                case 'px':
                    distance = value;
                    break
                case '%':
                    /**
                     * Here we use `getBoundingClientRect` instead of
                     * the existing data attached to `element.geometry`
                     * because only the former includes any transformations
                     * current applied to the element.
                     *
                     * If that behavior ends up being unintuitive, this
                     * logic could instead utilize `element.geometry.height`
                     * and `element.geoemetry.width` for the distance
calculation
                     */
                    distance =
                        axis === 'Y'
                            ? (element.node.getBoundingClientRect().height *
value) / 100
                            : (element.node.getBoundingClientRect().width *
value) / 100;
                    break
                default:
                    throw new RangeError('Unrecognized or missing distance
unit.')
            }

            if (axis === 'Y') {
                transformations.push(translateY(distance));
            } else {
                transformations.push(translateX(distance));
            }
        }

        if (config.rotate.x) { transformations.push(rotateX(config.rotate.x));
}
        if (config.rotate.y) { transformations.push(rotateY(config.rotate.y));
}
        if (config.rotate.z) { transformations.push(rotateZ(config.rotate.z));
}
        if (config.scale !== 1) {
            if (config.scale === 0) {
                /**
                 * The CSS Transforms matrix interpolation specification
                 * basically disallows transitions of non-invertible
                 * matrixes, which means browsers won't transition
                 * elements with zero scale.
                 *
                 * That’s inconvenient for the API and developer
                 * experience, so we simply nudge their value
                 * slightly above zero; this allows browsers
                 * to transition our element as expected.
                 *
                 * `0.0002` was the smallest number
                 * that performed across browsers.
                 */
                transformations.push(scale(0.0002));
            } else {
                transformations.push(scale(config.scale));
            }
        }

        var transform = {};
        if (transformations.length) {
            transform.property = getPrefixedCssProp('transform');
            /**
             * The default computed transform value should be one of:
             * undefined || 'none' || 'matrix()' || 'matrix3d()'
             */
            transform.computed = {
                raw: computed[transform.property],
                matrix: parse(computed[transform.property])
            };

            transformations.unshift(transform.computed.matrix);
            var product = transformations.reduce(multiply);

            transform.generated = {
                initial: ((transform.property) + ": matrix3d(" +
(product.join(', ')) + ");"),
                final: ((transform.property) + ": matrix3d(" +
(transform.computed.matrix.join(', ')) + ");")
            };
        } else {
            transform.generated = {
                initial: '',
                final: ''
            };
        }

        /**
         * Generate transition styles
         */
        var transition = {};
        if (opacity.generated || transform.generated.initial) {
            transition.property = getPrefixedCssProp('transition');
            transition.computed = computed[transition.property];
            transition.fragments = [];

            var delay = config.delay;
            var duration = config.duration;
            var easing = config.easing;

            if (opacity.generated) {
                transition.fragments.push({
                    delayed: ("opacity " + (duration / 1000) + "s " + easing +
" " + (delay / 1000) + "s"),
                    instant: ("opacity " + (duration / 1000) + "s " + easing +
" 0s")
                });
            }

            if (transform.generated.initial) {
                transition.fragments.push({
                    delayed: ((transform.property) + " " + (duration / 1000) +
"s " + easing + " " + (delay / 1000) + "s"),
                    instant: ((transform.property) + " " + (duration / 1000) +
"s " + easing + " 0s")
                });
            }

            /**
             * The default computed transition property should be undefined,
or one of:
             * '' || 'none 0s ease 0s' || 'all 0s ease 0s' || 'all 0s 0s
cubic-bezier()'
             */
            var hasCustomTransition =
                transition.computed && !transition.computed.match(/all 0s|none
0s/);

            if (hasCustomTransition) {
                transition.fragments.unshift({
                    delayed: transition.computed,
                    instant: transition.computed
                });
            }

            var composed = transition.fragments.reduce(
                function (composition, fragment, i) {
                    composition.delayed += i === 0 ? fragment.delayed : (", "
+ (fragment.delayed));
                    composition.instant += i === 0 ? fragment.instant : (", "
+ (fragment.instant));
                    return composition
                },
                {
                    delayed: '',
                    instant: ''
                }
            );

            transition.generated = {
                delayed: ((transition.property) + ": " + (composed.delayed) +
";"),
                instant: ((transition.property) + ": " + (composed.instant) +
";")
            };
        } else {
            transition.generated = {
                delayed: '',
                instant: ''
            };
        }

        return {
            inline: inline,
            opacity: opacity,
            position: position,
            transform: transform,
            transition: transition
        }
    }

    /**
     * apply a CSS string to an element using the CSSOM (element.style) rather
     * than setAttribute, which may violate the content security policy.
     *
     * @param {Node}   [el]  Element to receive styles.
     * @param {string} [declaration] Styles to apply.
     */
    function applyStyle (el, declaration) {
        declaration.split(';').forEach(function (pair) {
            var ref = pair.split(':');
            var property = ref[0];
            var value = ref.slice(1);
            if (property && value) {
                el.style[property.trim()] = value.join(':');
            }
        });
    }

    function clean(target) {
        var this$1 = this;

        var dirty;
        try {
            each(tealight(target), function (node) {
                var id = node.getAttribute('data-sr-id');
                if (id !== null) {
                    dirty = true;
                    var element = this$1.store.elements[id];
                    if (element.callbackTimer) {
                        window.clearTimeout(element.callbackTimer.clock);
                    }
                    applyStyle(element.node, element.styles.inline.generated);
                    node.removeAttribute('data-sr-id');
                    delete this$1.store.elements[id];
                }
            });
        } catch (e) {
            return logger.call(this, 'Clean failed.', e.message)
        }

        if (dirty) {
            try {
                rinse.call(this);
            } catch (e) {
                return logger.call(this, 'Clean failed.', e.message)
            }
        }
    }

    function destroy() {
        var this$1 = this;

        /**
         * Remove all generated styles and element ids
         */
        each(this.store.elements, function (element) {
            applyStyle(element.node, element.styles.inline.generated);
            element.node.removeAttribute('data-sr-id');
        });

        /**
         * Remove all event listeners.
         */
        each(this.store.containers, function (container) {
            var target =
                container.node === document.documentElement ? window :
container.node;
```

```
            target.removeEventListener('scroll', this$1.delegate);
            target.removeEventListener('resize', this$1.delegate);
        });

        /**
         * Clear all data from the store
         */
        this.store = {
            containers: {},
            elements: {},
            history: [],
            sequences: {}
        };
    }

    function deepAssign(target) {
        var sources = [], len = arguments.length - 1;
        while ( len-- > 0 ) sources[ len ] = arguments[ len + 1 ];

        if (isObject(target)) {
            each(sources, function (source) {
                each(source, function (data, key) {
                    if (isObject(data)) {
                        if (!target[key] || !isObject(target[key])) {
                            target[key] = {};
                        }
                        deepAssign(target[key], data);
                    } else {
                        target[key] = data;
                    }
                });
            });
            return target
        } else {
            throw new TypeError('Target must be an object literal.')
        }
    }

    function isMobile(agent) {
        if ( agent === void 0 ) agent = navigator.userAgent;

        return /Android|iPhone|iPad|iPod/i.test(agent)
    }

    var nextUniqueId = (function () {
        var uid = 0;
        return function () { return uid++; }
    })();
```

```javascript
    function initialize() {
        var this$1 = this;

        rinse.call(this);

        each(this.store.elements, function (element) {
            var styles = [element.styles.inline.generated];

            if (element.visible) {
                styles.push(element.styles.opacity.computed);
                styles.push(element.styles.transform.generated.final);
                element.revealed = true;
            } else {
                styles.push(element.styles.opacity.generated);
                styles.push(element.styles.transform.generated.initial);
                element.revealed = false;
            }

            applyStyle(element.node, styles.filter(function (s) { return s !==
''; }).join(' '));
        });

        each(this.store.containers, function (container) {
            var target =
                container.node === document.documentElement ? window :
container.node;
            target.addEventListener('scroll', this$1.delegate);
            target.addEventListener('resize', this$1.delegate);
        });

        /**
         * Manually invoke delegate once to capture
         * element and container dimensions, container
         * scroll position, and trigger any valid reveals
         */
        this.delegate();

        /**
         * Wipe any existing `setTimeout` now
         * that initialization has completed.
         */
        this.initTimeout = null;
    }

    function animate(element, force) {
        if ( force === void 0 ) force = {};
        var pristine = force.pristine || this.pristine;
        var delayed =
            element.config.useDelay === 'always' ||
            (element.config.useDelay === 'onload' && pristine) ||
            (element.config.useDelay === 'once' && !element.seen);

        var shouldReveal = element.visible && !element.revealed;
        var shouldReset = !element.visible && element.revealed &&
element.config.reset;

        if (force.reveal || shouldReveal) {
            return triggerReveal.call(this, element, delayed)
        }

        if (force.reset || shouldReset) {
            return triggerReset.call(this, element)
        }
    }

    function triggerReveal(element, delayed) {
        var styles = [
            element.styles.inline.generated,
            element.styles.opacity.computed,
            element.styles.transform.generated.final
        ];
        if (delayed) {
            styles.push(element.styles.transition.generated.delayed);
        } else {
            styles.push(element.styles.transition.generated.instant);
        }
        element.revealed = element.seen = true;
        applyStyle(element.node, styles.filter(function (s) { return s !== '';
}).join(' '));
        registerCallbacks.call(this, element, delayed);
    }

    function triggerReset(element) {
        var styles = [
            element.styles.inline.generated,
            element.styles.opacity.generated,
            element.styles.transform.generated.initial,
            element.styles.transition.generated.instant
        ];
        element.revealed = false;
        applyStyle(element.node, styles.filter(function (s) { return s !== '';
}).join(' '));
        registerCallbacks.call(this, element);
    }
    function registerCallbacks(element, isDelayed) {
        var this$1 = this;

        var duration = isDelayed
            ? element.config.duration + element.config.delay
            : element.config.duration;

        var beforeCallback = element.revealed
            ? element.config.beforeReveal
            : element.config.beforeReset;

        var afterCallback = element.revealed
            ? element.config.afterReveal
            : element.config.afterReset;

        var elapsed = 0;
        if (element.callbackTimer) {
            elapsed = Date.now() - element.callbackTimer.start;
            window.clearTimeout(element.callbackTimer.clock);
        }

        beforeCallback(element.node);

        element.callbackTimer = {
            start: Date.now(),
            clock: window.setTimeout(function () {
                afterCallback(element.node);
                element.callbackTimer = null;
                if (element.revealed && !element.config.reset &&
element.config.cleanup) {
                    clean.call(this$1, element.node);
                }
            }, duration - elapsed)
        };
    }

    function sequence(element, pristine) {
        if ( pristine === void 0 ) pristine = this.pristine;

        /**
         * We first check if the element should reset.
         */
        if (!element.visible && element.revealed && element.config.reset) {
            return animate.call(this, element, { reset: true })
        }

        var seq = this.store.sequences[element.sequence.id];
        var i = element.sequence.index;

        if (seq) {
            var visible = new SequenceModel(seq, 'visible', this.store);
            var revealed = new SequenceModel(seq, 'revealed', this.store);

            seq.models = { visible: visible, revealed: revealed };

            /**
             * If the sequence has no revealed members,
             * then we reveal the first visible element
             * within that sequence.
             *
             * The sequence then cues a recursive call
             * in both directions.
             */
            if (!revealed.body.length) {
                var nextId = seq.members[visible.body[0]];
                var nextElement = this.store.elements[nextId];

                if (nextElement) {
                    cue.call(this, seq, visible.body[0], -1, pristine);
                    cue.call(this, seq, visible.body[0], +1, pristine);
                    return animate.call(this, nextElement, { reveal: true,
pristine: pristine })
                }
            }

            /**
             * If our element isn’t resetting, we check the
             * element sequence index against the head, and
             * then the foot of the sequence.
             */
            if (
                !seq.blocked.head &&
                i === [].concat( revealed.head ).pop() &&
                i >= [].concat( visible.body ).shift()
            ) {
                cue.call(this, seq, i, -1, pristine);
                return animate.call(this, element, { reveal: true, pristine:
pristine })
            }

            if (
                !seq.blocked.foot &&
                i === [].concat( revealed.foot ).shift() &&
                i <= [].concat( visible.body ).pop()
            ) {
                cue.call(this, seq, i, +1, pristine);
                return animate.call(this, element, { reveal: true, pristine:
pristine })
            }
        }
    }

    function Sequence(interval) {
        var i = Math.abs(interval);
        if (!isNaN(i)) {
            this.id = nextUniqueId();
            this.interval = Math.max(i, 16);
            this.members = [];
            this.models = {};
            this.blocked = {
                head: false,
                foot: false
            };
        } else {
            throw new RangeError('Invalid sequence interval.')
        }
    }

    function SequenceModel(seq, prop, store) {
        var this$1 = this;

        this.head = [];
        this.body = [];
        this.foot = [];

        each(seq.members, function (id, index) {
            var element = store.elements[id];
            if (element && element[prop]) {
                this$1.body.push(index);
            }
        });

        if (this.body.length) {
            each(seq.members, function (id, index) {
                var element = store.elements[id];
                if (element && !element[prop]) {
                    if (index < this$1.body[0]) {
                        this$1.head.push(index);
                    } else {
                        this$1.foot.push(index);
                    }
                }
            });
        }
    }

    function cue(seq, i, direction, pristine) {
        var this$1 = this;

        var blocked = ['head', null, 'foot'][1 + direction];
        var nextId = seq.members[i + direction];
        var nextElement = this.store.elements[nextId];

        seq.blocked[blocked] = true;

        setTimeout(function () {
            seq.blocked[blocked] = false;
            if (nextElement) {
                sequence.call(this$1, nextElement, pristine);
            }
        }, seq.interval);
    }

    function reveal(target, options, syncing) {
        var this$1 = this;
        if ( options === void 0 ) options = {};
        if ( syncing === void 0 ) syncing = false;

        var containerBuffer = [];
        var sequence$$1;
        var interval = options.interval || defaults.interval;

        try {
            if (interval) {
                sequence$$1 = new Sequence(interval);
            }

            var nodes = tealight(target);
            if (!nodes.length) {
                throw new Error('Invalid reveal target.')
            }

            var elements = nodes.reduce(function (elementBuffer, elementNode)
{
                var element = {};
                var existingId = elementNode.getAttribute('data-sr-id');

                if (existingId) {
                    deepAssign(element, this$1.store.elements[existingId]);

                    /**
```

```
                     * In order to prevent previously generated styles
                     * from throwing off the new styles, the style tag
                     * has to be reverted to its pre-reveal state.
                     */
                    applyStyle(element.node, element.styles.inline.computed);
                } else {
                    element.id = nextUniqueId();
                    element.node = elementNode;
                    element.seen = false;
                    element.revealed = false;
                    element.visible = false;
                }

                var config = deepAssign({}, element.config || this$1.defaults,
options);

                if ((!config.mobile && isMobile()) || (!config.desktop &&
!isMobile())) {
                    if (existingId) {
                        clean.call(this$1, element);
                    }
                    return elementBuffer // skip elements that are disabled
                }

                var containerNode = tealight(config.container)[0];
                if (!containerNode) {
                    throw new Error('Invalid container.')
                }
                if (!containerNode.contains(elementNode)) {
                    return elementBuffer // skip elements found outside the
container
                }

                var containerId;
                {
                    containerId = getContainerId(
                        containerNode,
                        containerBuffer,
                        this$1.store.containers
                    );
                    if (containerId === null) {
                        containerId = nextUniqueId();
                        containerBuffer.push({ id: containerId, node:
containerNode });
                    }
                }

                element.config = config;
```

```
        element.containerId = containerId;
        element.styles = style(element);

        if (sequence$$1) {
            element.sequence = {
                id: sequence$$1.id,
                index: sequence$$1.members.length
            };
            sequence$$1.members.push(element.id);
        }

        elementBuffer.push(element);
        return elementBuffer
    }, []);

    /**
     * Modifying the DOM via setAttribute needs to be handled
     * separately from reading computed styles in the map above
     * for the browser to batch DOM changes (limiting reflows)
     */
    each(elements, function (element) {
        this$1.store.elements[element.id] = element;
        element.node.setAttribute('data-sr-id', element.id);
    });
} catch (e) {
    return logger.call(this, 'Reveal failed.', e.message)
}

/**
 * Now that element set-up is complete...
 * Let’s commit any container and sequence data we have to the store.
 */
each(containerBuffer, function (container) {
    this$1.store.containers[container.id] = {
        id: container.id,
        node: container.node
    };
});
if (sequence$$1) {
    this.store.sequences[sequence$$1.id] = sequence$$1;
}

/**
 * If reveal wasn't invoked by sync, we want to
 * make sure to add this call to the history.
 */
if (syncing !== true) {
    this.store.history.push({ target: target, options: options });
```

```
        /**
         * Push initialization to the event queue, giving
         * multiple reveal calls time to be interpreted.
         */
        if (this.initTimeout) {
            window.clearTimeout(this.initTimeout);
        }
        this.initTimeout = window.setTimeout(initialize.bind(this), 0);
    }
}

function getContainerId(node) {
    var collections = [], len = arguments.length - 1;
    while ( len-- > 0 ) collections[ len ] = arguments[ len + 1 ];

    var id = null;
    each(collections, function (collection) {
        each(collection, function (container) {
            if (id === null && container.node === node) {
                id = container.id;
            }
        });
    });
    return id
}

/**
 * Re-runs the reveal method for each record stored in history,
 * for capturing new content asynchronously loaded into the DOM.
 */
function sync() {
    var this$1 = this;

    each(this.store.history, function (record) {
        reveal.call(this$1, record.target, record.options, true);
    });

    initialize.call(this);
}

var polyfill = function (x) { return (x > 0) - (x < 0) || +x; };
var mathSign = Math.sign || polyfill;

/*! @license miniraf v1.0.1

    Copyright 2018 Fisssion LLC.
```

```
        Permission is hereby granted, free of charge, to any person obtaining
a copy
        of this software and associated documentation files (the "Software"),
to deal
        in the Software without restriction, including without limitation the
rights
        to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
        copies of the Software, and to permit persons to whom the Software is
        furnished to do so, subject to the following conditions:

        The above copyright notice and this permission notice shall be
included in all
        copies or substantial portions of the Software.

        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
        IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
        FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
        AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
        LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
        OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE
        SOFTWARE.

    */
    var polyfill$1 = (function () {
        var clock = Date.now();

        return function (callback) {
            var currentTime = Date.now();
            if (currentTime - clock > 16) {
                clock = currentTime;
                callback(currentTime);
            } else {
                setTimeout(function () { return polyfill$1(callback); }, 0);
            }
        }
    })();

    var miniraf = window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        polyfill$1;
    function getGeometry(target, isContainer) {
        /**
         * We want to ignore padding and scrollbars for container elements.
         * More information here: https://goo.gl/vOZpbz
         */
        var height = isContainer ? target.node.clientHeight :
target.node.offsetHeight;
        var width = isContainer ? target.node.clientWidth :
target.node.offsetWidth;

        var offsetTop = 0;
        var offsetLeft = 0;
        var node = target.node;

        do {
            if (!isNaN(node.offsetTop)) {
                offsetTop += node.offsetTop;
            }
            if (!isNaN(node.offsetLeft)) {
                offsetLeft += node.offsetLeft;
            }
            node = node.offsetParent;
        } while (node)

        return {
            bounds: {
                top: offsetTop,
                right: offsetLeft + width,
                bottom: offsetTop + height,
                left: offsetLeft
            },
            height: height,
            width: width
        }
    }

    function getScrolled(container) {
        var top, left;
        if (container.node === document.documentElement) {
            top = window.pageYOffset;
            left = window.pageXOffset;
        } else {
            top = container.node.scrollTop;
            left = container.node.scrollLeft;
        }
        return { top: top, left: left }
    }
    function isElementVisible(element) {
        if ( element === void 0 ) element = {};

        var container = this.store.containers[element.containerId];
        if (!container) { return }

        var viewFactor = Math.max(0, Math.min(1, element.config.viewFactor));
        var viewOffset = element.config.viewOffset;

        var elementBounds = {
            top: element.geometry.bounds.top + element.geometry.height *
viewFactor,
            right: element.geometry.bounds.right - element.geometry.width *
viewFactor,
            bottom: element.geometry.bounds.bottom - element.geometry.height *
viewFactor,
            left: element.geometry.bounds.left + element.geometry.width *
viewFactor
        };

        var containerBounds = {
            top: container.geometry.bounds.top + container.scroll.top +
viewOffset.top,
            right: container.geometry.bounds.right + container.scroll.left -
viewOffset.right,
            bottom:
                container.geometry.bounds.bottom + container.scroll.top -
viewOffset.bottom,
            left: container.geometry.bounds.left + container.scroll.left +
viewOffset.left
        };

        return (
            (elementBounds.top < containerBounds.bottom &&
                elementBounds.right > containerBounds.left &&
                elementBounds.bottom > containerBounds.top &&
                elementBounds.left < containerBounds.right) ||
            element.styles.position === 'fixed'
        )
    }

    function delegate(
        event,
        elements
    ) {
        var this$1 = this;
 }
            });
            this$1.pristine = false;
        });
    }

    function isTransformSupported() {
        var style = document.documentElement.style;
        return 'transform' in style || 'WebkitTransform' in style
    }

    function isTransitionSupported() {
        var style = document.documentElement.style;
        return 'transition' in style || 'WebkitTransition' in style
    }

    var version = "4.0.9";

    var boundDelegate;
    var boundDestroy;
    var boundReveal;
    var boundClean;
    var boundSync;
    var config;
    var debug;
    var instance;

    function ScrollReveal(options) {
        if ( options === void 0 ) options = {};

        var invokedWithoutNew =
            typeof this === 'undefined' ||
            Object.getPrototypeOf(this) !== ScrollReveal.prototype;

        if (invokedWithoutNew) {
            return new ScrollReveal(options)
        }

        if (!ScrollReveal.isSupported()) {
            logger.call(this, 'Instantiation failed.', 'This browser is not 
supported.');
            return mount.failure()
        }

        var buffer;
        try {
            buffer = config
                ? deepAssign({}, config, options)
                : deepAssign({}, defaults, options);
        } catch (e) {
            logger.call(this, 'Invalid configuration.', e.message);
            return mount.failure()
        }

        try {
            var container = tealight(buffer.container)[0];
            if (!container) {
                throw new Error('Invalid container.')
            }
        } catch (e) {
            logger.call(this, e.message);
            return mount.failure()
        }

        config = buffer;

        if ((!config.mobile && isMobile()) || (!config.desktop &&
!isMobile())) {
            logger.call(
                this,
                'This device is disabled.',
                ("desktop: " + (config.desktop)),
                ("mobile: " + (config.mobile))
            );
            return mount.failure()
        }

        mount.success();

        this.store = {
            containers: {},
            elements: {},
            history: [],
            sequences: {}
        };

        this.pristine = true;

        boundDelegate = boundDelegate || delegate.bind(this);
        boundDestroy = boundDestroy || destroy.bind(this);
        boundReveal = boundReveal || reveal.bind(this);
        boundClean = boundClean || clean.bind(this);
        boundSync = boundSync || sync.bind(this);

        Object.defineProperty(this, 'delegate', { get: function () { return
boundDelegate; } });
```

```
        Object.defineProperty(this, 'destroy', { get: function () { return
boundDestroy; } });
        Object.defineProperty(this, 'reveal', { get: function () { return
boundReveal; } });
        Object.defineProperty(this, 'clean', { get: function () { return
boundClean; } });
        Object.defineProperty(this, 'sync', { get: function () { return
boundSync; } });

        Object.defineProperty(this, 'defaults', { get: function () { return
config; } });
        Object.defineProperty(this, 'version', { get: function () { return
version; } });
        Object.defineProperty(this, 'noop', { get: function () { return false;
} });

        return instance ? instance : (instance = this)
    }

    ScrollReveal.isSupported = function () { return isTransformSupported() &&
isTransitionSupported(); };

    Object.defineProperty(ScrollReveal, 'debug', {
        get: function () { return debug || false; },
        set: function (value) { return (debug = typeof value === 'boolean' ?
value : debug); }
    });

    ScrollReveal();

    return ScrollReveal;

}));
```