98.css

A design system for building faithful recreations of old UIs.

Intro

98.css is a CSS library for building interfaces that look like Windows 98. See more on GitHub.

This library relies on the usage of **semantic HTML**. To make a button, you'll need to use a `<button>`. Input elements require labels. Icon buttons rely on `aria-label`. This page will guide you through that process, but accessibility is a primary goal of this project.

You can override many of the styles of your elements while maintaining the appearance provided by this library. Need more padding on your buttons? Go for it. Need to add some color to your input labels? Be our guest.

This library does not contain any JavaScript. It merely styles your HTML with some CSS. This means 98.css is compatible with your frontend framework of choice.

Here is an example of 98.css used with React, and an example with vanilla JavaScript. The fastest way to use 98.css is to import it from unpkg.

```
<link
  rel="stylesheet"
  href="https://unpkg.com/98.css" />
```

You can install 98.css from the GitHub releases page, or from npm.

```
npm install 98.css
```

Components

Button

A command button, also referred to as a push button, is a control that causes the application to perform some action when the user clicks it.

— Microsoft Windows User Experience p. 160

A standard button measures 75px wide and 23px tall, with a raised outer and inner border. They are given 12px of horizontal padding by default.

▸ Show code

When buttons are clicked, the raised borders become sunken. The following button is simulated to be in the pressed (active) state.

▸ Show code

Disabled buttons maintain the same raised border, but have a *washed out* appearance in their label.

▸ Show code

Button focus is communicated with a dotted border, set 4px within the contents of the button. The following example is simulated to be focused.

▸ Show code

Checkbox

A check box represents an independent or non-exclusive choice.

— Microsoft Windows User Experience p. 167

Checkboxes are represented with a sunken panel, populated with a "check" icon when selected, next to a label indicating the choice.

Note: You **must** include a corresponding label **after** your checkbox, using the `<label>` element with a `for` attribute pointed at the `id` of your input. This ensures the checkbox is easy to use with assistive technologies, on top of ensuring a good user experience for all (navigating with the tab key, being able to click the entire label to select the box).

▸ Show code

Checkboxes can be selected and disabled with the standard `checked` and `disabled` attributes.

When grouping inputs, wrap each input in a container with the `.field-row` class. This ensures a consistent spacing between inputs.

▸ Show code

OptionButton

An option button, also referred to as a radio button, represents a single choice within a limited set of mutually exclusive choices. That is, the user can choose only one set of options.

— Microsoft Windows User Experience p. 164

Option buttons can be used via the `radio` type or an input element.

Option buttons can be grouped by specifying a shared name attribute on each input. Just as before: when grouping inputs, wrap each input in a container with the `.field-row` class to ensure a consistent spacing between inputs.

▸ Show code

Option buttons can also be `checked` and `disabled` with their corresponding HTML attributes.

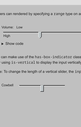
▸ Show code

GroupBox

A group box is a special control you can use to organize a set of controls. A group box is a rectangular frame with an optional label that surrounds a set of controls.

— Microsoft Windows User Experience p. 186

A group box can be used by wrapping your elements with the `fieldset` tag. It contains a sunken outer border and a raised inner border, resembling an engraved box around your controls.



▸ Show code

You can provide your group with a label by placing a `legend` element within the `fieldset`.

▸ Show code

TextBox

A text box (also referred to as an edit control) is a rectangular control where the user enters or edits text. It can be defined to support a single line or multiple lines of text.

— Microsoft Windows User Experience p. 191

Text boxes can rendered by specifying a `text` type on an `input` element. As with checkboxes and radio buttons, you should provide a corresponding label with a property set `for` attribute, and wrap both in a container with the `field-row` class.

▸ Show code

Additionally, you can make use of the `field-row-stacked` class to position your label above the input instead of beside it.

▸ Show code

To support multiple lines in the user's input, use the `textarea` element instead.

▸ Show code

Text boxes can also be disabled and have value with their corresponding HTML attributes.

▸ Show code

Other types of HTML5 text inputs are also supported.

▸ Show code

Slider

A slider, sometimes called a trackbar control, consists of a bar that defines the extent or range of the adjustment and an indicator that shows the current value for the control...

— Microsoft Windows User Experience p. 146

Sliders can rendered by specifying a `range` type on an `input` element.

▸ Show code

You can make use of the `has-box-indicator` class replace the default indicator with a box indicator. furthermore the slider can be wrapped with a `div` using `is-vertical` to display the input vertically.

Note: To change the length of a vertical slider, the `input` width and `div` height.

▸ Show code

Dropdown

A drop-down list box allows the selection of only a single item from a list. In its closed state, the control displays the current value for the control. The user opens the list to change the value.

— Microsoft Windows User Experience p. 170

Dropdowns can be rendered by using the `select` and `option` elements.

▸ Show code

By default, the first option will be selected. You can change this by giving one of your `option` elements the `selected` attribute.

▸ Show code

Window

The following components illustrate how to build complete windows using 98.css.

Title Bar

At the top edge of the window, inside its border, is the title bar (also referred to as the caption or caption bar), which extends across the width of the window. The title bar identifies the contents of the window.

— Microsoft Windows User Experience p. 118

Include command buttons associated with the common commands of the primary window in the title bar. These buttons act as shortcuts to specific window commands.

— Microsoft Windows User Experience p. 122

You can build a complete title bar by making use of three classes, `title-bar-text`, `title-bar-text`, and `title-bar-controls`.

▸ Show code

We make use of `aria-label` to render the Close button, to let assistive technologies know the intent of this button. You may also use "Minimize", "Maximize", "Restore" and "Help" like so:

▸ Show code

You can make a title bar "inactive" by adding `inactive` class...useful when making more than one window.

▸ Show code

Window contents

Every window has a boundary that defines its shape.

— Microsoft Windows Experience p. 118

To give our title bar a home, we make use of the `window` class. This provides a raised outer and inner border, as well as some padding. We can freely resize the window by specifying a width in the container style.

▸ Show code

To draw the contents of the window, we use the `window-body` class under the title bar.

▸ Show code

Status Bar

A status bar is a special area within a window, typically the bottom, that displays information about the current state of what is being viewed in the window or any other contextual information, such as keyboard state.

— Microsoft Windows User Experience p. 146

You can render a status bar with the `status-bar` class, and `status-bar-field` for every child text element.

▸ Show code

TreeView

A tree-view control is a special list box control that displays a set of objects as an indented outline based on their logical hierarchical relationship.

— Microsoft Windows User Experience p. 179

To render a tree view, use an `ul` element with the `tree-view` class. The children of this list (`li` elements), can contain whatever you'd like.

▸ Show code

To make this a tree, we can nest further `ul` elements (no class needed on these). This will provide them with a nice dotted border and indentation to illustrate the structure of the tree.

To create expandable sections, wrap child lists inside of `details` elements.

▸ Show code

Issues, Contributing, etc.

98.css is MIT licensed.

Refer to the GitHub issues page to see bugs in my CSS or report new ones. I'd really like to see your pull requests (especially those new to open-sourced) and will happily provide code reviews. 98.css is a fun, silly project and I'd like to make it a fun place to build your open-source muscle.

Thank you for checking my little project out, I hope it brought you some joy today. Consider starring/following along on GitHub and maybe subscribing to more fun things on my twitter.