LEARN CSS PSEUDO SELECTORS BY BUILDING A BALANCE SHEET

Introduction:

You can use CSS pseudo selectors to change specific HTML elements.

In this course, you'll build a balance sheet using pseudo selectors. You'll learn how to change the style of an element when you hover over it with your mouse, and trigger other events on your webpage.

Step 1:

Set up your HTML with the `DOCTYPE`, `html` indicating this document is in English, `head`, and `body` elements.

Give your `head` element the appropriate `meta` elements for the `charset` and `viewport`, a `title` element with an appropriate title, and a `link` element for your stylesheet.

Step 2:

Within your `body` element, nest a `section` element within a `main` element.

Step 3:

Within your `section` element, add an `h1` element with a nested `span` element.

Step 4:

Screen readers announce HTML elements based on the document flow. We will eventually want the balance sheet to have a heading of "Balance Sheet" and a subheading of "AcmeWidgetCorp". However, this order does not make sense if announced by a screen reader.

Give your existing `span` the `class` attribute set to `flex`, and add two `span` elements within it. Give the first the text `AcmeWidgetCorp`. Give the second the text `Balance Sheet`. You will use CSS to reverse the order of the text on the page, but the HTML order will make more sense for a screen reader.

Step 5:

Below your `h1` element, create a `div` element. Give it an `id` attribute set to `years`. You want this particular element to be hidden from screen readers, so give it the `aria-hidden` attribute set to `true`.

Step 6:

Within your `div` element, add three `span` elements. Give each of them a `class` attribute set to `year`, and add the following text (in order): `2019`, `2020`, and `2021`.

Step 7:

Below your existing `div` element, add a new `div` element with a `class` set to `table-wrap`. This will be the container for your tables.

Within that, add three `table` elements. You will be using CSS to style these into a single table, but using separate tables will help screen readers understand the document flow.

Step 8:

HTML tables use the `caption` element to describe what the table is about. The `caption` element should always be the first child of a `table`, but can be positioned with the `caption-side` CSS property.

Add a `caption` element to your first `table`, and give it the text `Assets`.

Step 9:

The `thead` and `tbody` elements are used to indicate which portion of your table is the header, and which portion contains the primary data or content.

Add a `thead` and `tbody` to your first `table`, below the `caption` element.

Step 10:

The `tr` element is used to indicate a table row. Add a `tr` element within your `thead` element. In your new `tr` element, add a `td` element, followed by three `th` elements.

The `td` element indicates a data cell, while the `th` element indicates a header cell.

Step 11:

Within each of your new `th` elements, nest a `span` element with the `class` set to `sr-only year`. Give them the following text (in order): `2019`, `2020`, and `2021`.

Give your third `th` element the `class` attribute set to `current`.

Leave the `td` element empty. This element exists only to ensure your table has a four-column layout and associate the headers with the correct columns.

Step 12:

Within your tbody element, add four tr elements. Give the first three a class attribute set to data, and the fourth a class attribute set to total.

Step 13:

In your first tr, add a th element with the text Cash This is the cash we currently have on hand.. Wrap all of that text except Cash in a span element with the class set to description.

Following that, add three td elements with the following text (in order): $25, $30, $28. Give the third td element a class attribute set to current.

Step 14:

In your second tr element, add a th element with the text Checking Our primary transactional account.. Wrap that text, except for Checking , in a span element with the class attribute set to description.

Following that, add three td elements with the following text (in order): $54, $56, $53. Give the third td element a class attribute set to current.

Step 15:

In your third tr element, add a th element with the text Savings Funds set aside for emergencies.. Wrap that text, except for Savings , in a span element with the class attribute set to description.

Following that, add three td elements with the following text (in order): $500, $650, $728. Give the third td element a class attribute set to current.

Step 16:

In your fourth tr element, add a th element with the text Total Assets. Wrap the text Assets in a span element with the class attribute set to sr-only.

Following that, add three td elements with the following text (in order): $579, $736, $809. Give the third td element a class attribute set to current.

Step 17:

Time to move on to your second table. Start by giving it a caption element set to Liabilities. Then add your thead and tbody.

Step 18:

Within your thead, add a tr. Inside that, add a td and three th elements.

Step 19:

Give each th element a span element with the class set to sr-only and the following text, in order: 2019, 2020, and 2021.

Step 20:

Within the tbody element, add four tr elements. Give the first three the class attribute set to data, and the fourth the class attribute set to total.

Step 21:

Within the first tr, add a th element with the text Loans The outstanding balance on our startup loan.. Wrap that text, except for Loans , within a span element with the class set to description.

Add three td elements below that, and give them the following text, in order: $500, $250, and $0. Give the third td element a class set to current.

Step 22:

Within the second tr, add a th element with the text Expenses Annual anticipated expenses, such as payroll.. Wrap that text, except for Expenses , within a span element with the class set to description.

Add three td elements below that, and give them the following text, in order: $200, $300, and $400. Give the third td element a class set to current.

Step 23:

Within the third tr, add a th element with the text Credit The outstanding balance on our credit card.. Wrap that text, except for Credit , within a span element with the class set to description.

Add three td elements below that, and give them the following text, in order: $50, $50, and $75. Give the third td element a class set to current.

Step 24:

In your fourth tr element, add a th element with the text Total Liabilities. Wrap the text Liabilities in a span element with the class attribute set to sr-only.

Following that, add three td elements with the following text (in order): $750, $600, $475. Give the third td element a class attribute set to current.

Step 25:

For your third table, add a caption with the text Net Worth, and set up a table header and table body.

Step 26:

Within your thead, create a tr element. In that, add a td and three th elements. Within each of the th elements, add a span element with the class set to sr-only and the following text, in order: 2019, 2020, and 2021.

Step 27:

Within the tbody, add a tr with the class set to total. In that, add a th with the text Total Net Worth, and wrap Net Worth in a span with the class set to sr-only.

Then add three td elements, giving the third a class set to current, and giving each the following text: $-171, $136, $334.

Step 28:

Time to style your table. Start by resetting the box model. Create an `html` selector and give it a `box-sizing` property set to `border-box`.

Step 29:

Create a `body` selector and give it a `font-family` property set to `sans-serif` and a `color` set to `#0a0a23`.

Step 30:

Before you get too far into your styling, you should make use of the `sr-only` class. You can use CSS to make elements with this class completely hidden from the visual page, but still be announced by screen readers.

The CSS you are about to write is a common set of properties used to ensure elements are completely hidden visually.

The `span[class~="sr-only"]` selector will select any `span` element whose `class` *includes* `sr-only`. Create that selector, and give it a `border` property set to `0`.

Step 31:

The CSS `clip` property is used to define the visible portions of an element. Set the `span[class~="sr-only"]` selector to have a `clip` property of `rect(1px, 1px, 1px, 1px)`.

The `clip-path` property determines the shape the `clip` property should take. Set the `clip-path` property to the value of `inset(50%)`, forming the clip-path into a rectangle within the element.

Step 32:

Now you need to size these elements down. Give your `span[class~="sr-only"]` selector a `width` and `height` property set to `1px`.

Step 33:

To prevent the text content from overflowing, give your `span[class~="sr-only"]` selector an `overflow` property set to `hidden` and a `white-space` property set to `nowrap`.

Step 34:

Finally, you need to take these hidden elements out of the document flow. Give the `span[class~="sr-only"]` selector a `position` property set to `absolute`, a `padding` property set to `0`, and a `margin` property set to `-1px`. This will ensure that not only are they no longer visible, but they are not even within the page view.

Step 35:

Time to style your table heading. Create an `h1` selector. Give it a `max-width` property set to `37.25rem`, a `margin` property set to `0 auto`, and a `padding` property set to `1.5rem 1.25rem`.

Step 36:

Target your flex container with an `h1 .flex` selector. Give it a `display` property set to `flex` to enable the flexbox layout. Then set the `flex-direction` property to `column-reverse` - this will display the nested elements from bottom to top. Finally, set the `gap` property to `1rem` to create some space between the elements.

Step 37:

The `:first-of-type` pseudo-selector is used to target the first element that matches the selector. Create an `h1 .flex span:first-of-type` selector to target the first `span` element in your `.flex` container. Remember that your `span` elements are reversed, visually, so this will appear to be the second element on the page.

Give your new selector a `font-size` property of `0.7em` to make it look like a sub-heading.

Step 38:

The `:last-of-type` pseudo-selector does the exact opposite - it targets the last element that matches the selector. Create an `h1 .flex span:last-of-type` selector to target the last `span` in your flex container, and give it a `font-size` property set to `1.2em` to make it look like a header.

Step 39:

You wrapped your table in a section element - now you can style that to give your table a border. Create a `section` selector, and give it a `max-width` property set to `40rem` for responsive design. Set the `margin` property to `0 auto` to center it, and set the `border` property to `2px solid #d0d0d5`.

Step 40:

The last part of your table heading is your years. Create a `#years` selector, and enable flexbox. Justify the content to the end of the flex direction, and make the element sticky. Fix it to the top of its container with `top: 0`.

Step 41:

Now apply some color to your #years. Make the text #fff and the background #0a0a23.

Step 42:

The calc() function is a CSS function that allows you to calculate a value based on other values. For example, you can use it to calculate the width of the viewport minus the margin of an element:

Example Code:

```
.example {

  margin: 10px;

  width: calc(100% - 20px);

}
```

Give #years a margin of 0 -2px, and a padding set to 0.5rem calc(1.25rem + 2px) 0.5rem 0.

Step 43:

Adding position sticky moved the element into its own stack. To ensure your #years element does not get hidden by different stacks, add a z-index property set to 999 in the #years rule.

Step 44:

Style the text within your #years element by creating a #years span[class] selector. The span[class] syntax will target any span element that has a class attribute set, regardless of the attribute's value.

Give your new selector a bold font, a width of 4.5rem, and text aligned to the right.

Step 45:

You wrapped your tables in a container with the table-wrap class. Create a selector for that class, and give it a padding set to 0 0.75rem 1.5rem 0.75rem.

Step 46:

Before you start diving into the table itself, your span elements are currently bolded. Create a span:not(.sr-only) selector and give it a font-weight property set to normal.

The :not() pseudo-selector is used to target all elements that do not match the selector - in this case, any of your span elements that do not have the sr-only class. This ensures that your earlier rules for the span[class~="sr-only"] selector are not overwritten.

Step 47:

Rather than having to constantly double-check you are not overwriting your earlier properties, you can use the !important keyword to ensure these properties are always applied, regardless of order or specificity.

Give each property in your span[class~="sr-only"] selector an !important keyword. Do not change any of the values.

Step 48:

Now that you have added the !important keyword, you can remove the :not(.sr-only) from your span selector.

Step 49:

Create a `table` selector to target your tables. Set the `border-collapse` property to `collapse`, which will allow cell borders to collapse into a single border, instead of a border around each cell. Also set the `border` property to `0` to hide the borders themselves.

Step 50:

Ensure your table fills its container with a `width` property set to `100%`, then position it relatively and give it a top margin of `3rem`.

Step 51:

Next you need to style your `caption` elements to look more like headers. Create a `table caption` selector. Set the text to have a color of `#356eaf`, a size of `1.3em`, and a normal weight.

Step 52:

Now give the captions an absolute position, and shift them `-2.25rem` from the top and `0.5rem` from the left.

Step 53:

Create a selector to target your `td` elements within your table body. Give them a width to fill the viewport, with a minimum and maximum of `4rem`. This approach ensures that the width is fixed, whereas setting `width` specifically would allow the elements to shrink to the container.

Step 54:

Now target the `th` elements within your table body, and give them a width of the entire container, less `12rem`.

Step 55:

The `[attribute="value"]` selector targets any element that has an attribute with a specific value. Create a `tr[class="total"]` selector to target specifically your `tr` elements with the `total` class. Give it a bottom border of `4px double #0a0a23` and make the font bold.

Step 56:

Using the same selector syntax, target the `th` elements within your table rows where the `class` is `total`. Align the text to the left, and give them a padding of `0.5rem 0 0.25rem 0.5rem`.

Step 57:

The key difference between `tr[class="total"]` and `tr.total` is that the first will select `tr` elements where the *only* class is `total`. The second will select `tr` elements where the class *includes* `total`.

In your case, `tr.total` will work. You can use this selector to target all `td` elements within your `.total` rows. Align the text to the right, and give them a padding of `0 0.25rem`.

Step 58:

he `:nth-of-type()` pseudo-selector is used to target specific elements based on their order among siblings of the same type. Use this pseudo-selector to target the third `td` element within your `total` table rows. Give it a right padding of `0.5rem`.

Step 59:

Give your `tr.total` elements a hover effect that changes the background to `#99c9ff`.

Step 60:

Select your `td` elements with the `class` value of `current`, and make the font italic.

Step 61:

Select the `tr` elements with the `class` set to `data`. Give them a background image of `linear-gradient(to bottom, #dfdfe2 1.845rem, white 1.845rem)`.

Step 62:

Select the `th` elements within your `tr.data` elements. Align the text to the left, and give them a top padding of `0.3rem` and a left padding of `0.5rem`.

Step 63:

Create a `tr.data th .description` selector to target the elements with the `class` set to `description` that are within your `th` elements in your `.data` table rows. Give them a block display, make the text italic with a normal weight, and position them with a `padding` set to `1rem 0 0.75rem` and a right margin of `-13.5rem`.

Step 64:

Your `span` elements now all have more specific styling, which means you can remove your `span` rule.

Step 65:

Your dollar amounts are currently misaligned. Create a selector to target the `td` elements within your `tr.data` elements. Vertically align the text to the top, horizontally align the text to the right, and set the padding to `0.3rem 0.25rem 0`.

Step 66:

Create another selector for the `td` elements within your `tr.data` element, but specifically select the last one. Give this a right padding of `0.5rem`.

With this, your balance sheet is complete!