Monday 29th Jan 2024

[CSS Pseudo Selectors - A Balanced Sheet]

① Setting up the HTML.

② Adding <section></section>

③ <h1></h1>

④
ABC
DEF

⑤ [Below h1 element] creating a div element + id attribute of years. You want this particular element to be hidden from screen readers, so give it the ~~aria~~ aria-hidden attribute and set to true.

⑥ (index.html) adding 3 span elements within the div. [cont'd]
2019 x3.

⑦ (index.html) adding a new div element + three table elements.
 → buat jadi container.
<table></table> x3

⑧ (index.html) Adding a caption's table.
Note= the caption element should be the 1st child of a table, but can be positioned with the caption-side CSS property!

<table>
<caption>Assets</caption>
</table>

⑨ (index.html) Adding the thead and tbody elements (used to indicate which portion of your table is the header, and which portion contains the primary data or content.

<table>
<caption>Assets</caption>
<thead></thead>
<tbody></tbody>
</table>

(10) (index.html) The tr element is used to indicate a table row. The td element indicates a data cell, and th element indicates a header cell.

```
<thead>
  <tr><td>
    <th></th>
    <th></th>
    <th></th>
  </td></tr>
</thead>
```

(11) (index.html) [cont'd]

```
<thead>
  <tr>
    <td></td>
    <th><span class="sr-only year">2019</span></th>
                                  2020
    <th class="current"><span class="sr-only year"
    current>2021</span></th>
```

(12) (index.html)

```
<tbody>
  <tr class="data"></tr>   x3
  <tr class="total"></tr>
</tbody>
```

(13) (index.html) → first <tr></tr>

```
<tr class="data">
  <th>Cash <span class="description">
  This is the cash we currently have
  on hand.</span></th>
  <td>$25</td>
  <td>$30</td>
  <td class="current">$28</td>
</tr>
```

(14) (index.html) EXAMPLES ONLY!!!

```
<tr class="data"> → 2nd <tr></tr>
  <th>Checking <span class="description">
  Xxz.</span></th>
  <td>$54</td>
  <td class="current">$53</td>
</tr>
```

(15) (index.html) → 3rd <tr></tr>

```
<tr class="data">
```

⇒ all the formats are exactly same like step 13 & 14.

(16) (index.html) → 4th <tr></tr>

```
<tr class="total">
  <th>Total <span class="sr-only">Assets</span></th>
  <td>$579</td>
  ⋮
  <td class="current">$809</td>
</tr>
```

(17) (index.html) written on the 2nd table.

```
<table>
  <caption>Liabilities</caption>
```

→

```
<thead></thead>
<tbody></tbody>
</table>
```

(18) (index.html) [cont'd]

Adding <tr></tr> within the <thead></thead>

<td></td>

<th></th>

```
<thead>
  <tr>
    <td>
      <th></th> x 3
    </td>
  </tr>
</thead>
```

(19) (index.html) [cont'd] giving each th element a span element with the class set to sr-only and the following text in order.

ex : <th> 2019 < th>

(20) (index.html) [cont'd] adding 4 (tr) elements. Setting the (class) attribute set to (data) for the first three (tr) elements & setting the 4th the (class) attribute set to (total).

(21) (index.html) [cont'd]

(see step 13 for the reference)

(22) (index.html) - see step 13 for the reference.

(23) (index.html) - steps & the formats are same like step 22.

(index.html) - still as same as step 23.

(24) (index.html) [for the 4th tr element].

```
<tr class="total">
  <th> Total <span class="sr-only">Liabilities</span></th>
  <td> $750</td>
  <td> $600 </td>
  <td class="current"> $475 </td>
</tr>
```

(25) (index.html) → for the 3rd table.

```
<table>
  <caption>Net Worth</caption>
  <thead></thead>
  <tbody></tbody>
</table>
```

(26) (index.html) → cont'd

```
<td></td>
  <th><span class="sr-only">2019</
  span></th> x 3
```

↓

2020

↓

2021

(27) (index.html) → within the tbody → the format is still same.

(28) (styles.css)

```
html {
  box-sizing: border-box;
}
```

(29) (styles.css)

```
body {
  font-family: sans-serif;
  color : #0a0a23;
}
```

(30) (styles.css) 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.

```
span[class~="sr-only"] {
  border : 0;
}
```

→ this selector will select any span element whose class includes sr-only.

(31) (styles.css) [cont'd] The CSS `clip` property is used to define the visible portions of an element.

```
span[class~="sr-only"] {
  border: 0;
  clip : rect(1px, 1px, 1px, 1px);
  clip-path: inset(50%);
}
```

→ forming the clip-path into a rectangle. Within the element

→ determines the shape the clip property should take.

(32) (styles.css) [cont'd] → sizing these elements down.

```
width : 1px;
height : 1px;
```

(33) (styles.css) [cont'd] → preventing the text content from overflowing

```
overflow : hidden;
white-space: nowrap;
```

(34) (styles.css) [cont'd]

you need to take these hidden elements out of the document flow.

```
Position: absolute;
Padding: 0;
margin: -1px;
```

this will ensure that not only are they no longer visible, but they are not even within the page view.

(35) (styles.css) styling the table heading.

```
h1 {
  max-width: 37.25rem;
  margin: 0 auto;
  padding: 1.5rem 1.25rem;
}
```

(36) (styles.css) targetting your flex container.

```
h1 .flex {
  display: flex;
  flex-direction: column-reverse;
  gap: 1rem;
}
```

→ for enabling the flexbox layout

→ this will display the nested elements from bottom to top.

→ for creating some space

(37) (styles.css)

```
h1 .flex span:first-of-type {
  font-size: 0.7em;
}
```

→ it is called pseudo-selector that is used to target the 1st element that matches the selector.

#h1 .flex span:first-of-type → to target the first span element in your .flex container

Note: your span elements are reversed visually, so this will appear to be the 2nd element on the page (?).

(38) (styles.css)

```
h1 .flex span:last-of-type {
  font-size: 1.2em;
}
```

→ it targets the last element that matches the selector.

#h1 .flex span:last-of-type → to target the last span in your flex container.

(39) (styles.css) →

```
section {
  max-width: 40rem;
  margin: 0 auto;
  border: 2px solid #d0d0d5; }
```

(10) (styles.css) [READ THE INSTRUCTIONS !!]

create a #years selector; → #years { }

enable flexbox; → display: flex;

justify the content to the end of the flex direction; → position: sticky;

make the element sticky; → justify-content: flex-end;

top: 0 → top: 0;

(11) (styles.css) [cont'd from step 40].

color: #fff;

background-color: #0a0a23;

(12) (styles.css) The calc() function is a CSS function that allows you to calculate a value based on the other values.

Ex: you can use it to calculate the width of the viewport minus the margin of an element:

```
.example {
  margin: 10px;
  width: calc(100% - 20px);
}
```

```
#years {
  margin: 0 -2px;
  padding: 0.5rem calc(1.25rem + 2px) 0.5rem 0;
}
```

(13) (styles.css) [cont'd] adding z-index: 999;

(14) (styles.css) styling the text within your #years element by creating a #years span [class]

```
#years span[class] {
  font-weight: bold;
  width: 4.5rem;
  text-align: right;
}
```

↳ this syntax will target any span element that has a class attribute set, regardless of the attribute's value.

(15) (styles.css)

```
.table-wrap {
  padding: 0 0.75rem 1.5rem 0.75rem;
}
```

(46) Before you start diving in to 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.

Note: 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.

```
span:not(.sr-only) {
    font-weight: normal;
}
```

(47) (styles.css) 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.

"!important" ditulis setelah value utama

```
span[class~="sr-only"] {
    border: 0 !important;
    .
    .
    .
    so on
}
```

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

```
span {
    font-weight: normal;
}
```

(49) (styles.css)

allowing cell borders to collapse into a single border.

```
table {
    border-collapse: collapse;
    border: 0;
}
```

(50) (styles.css) [cont'd]

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.

```
width: 100%;
position: relative;
margin-top: 3rem;
```

(51) (styles.css) styling your captions to look more like headers.

```
table caption {
  color: #356eaf;
  font-size: 1.3em;
  font-weight: normal;
}
```

(52) (styles.css) [cont'd]

" shift them -2.25rem from the top & 0.5rem from the left."

```
position: absolute;
top: -2.25rem;
left: 0.5rem;
```

(53) (styles.css) creating a selector to target your td elements within your table body. Give them a width to fill the viewport, with a minimum * max of 4rem.

```
tbody td {
  width: 100vw;
  min-width: 4rem;
  max-width: 4rem;
}
```

(54) (styles.css) targetting the th elements within your table body, and give them a width of the entire container less 12rem.

```
tbody th {
  width: calc(100% - 12rem);
}
```

(55) (styles.css)

The [attribute="value"] selector targets any element that has an attribute with a specific value.

```
tr[class="total"] {
  border-bottom: 4px double #0a0a23;
  font-weight: bold;
}
```

(56) (styles.css)

target the th elements within your table rows where the class is total.

```
tr[class="total"] th {
  text-align: left;
  padding: 0.5rem 0 0.25rem 0.5rem;
}
```

(57) (styles.css) 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`.

```
tr.total td {
  text-align: right;
  padding: 0 0.25rem;
}
```

(58) (styles.css) The `: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.

```
tr.total td:nth-of-type(3)
```
↳ no space.

(59) (styles.css)

```
tr.total:hover {
  background: #99c9ff;
}
```

(60) (styles.css) select your `td` elements with the `class` value of `current` and make the font italic.

```
td[class="current"] {
  font-style: italic;
}
```

(61) (styles.css)

```
tr[class="data"]{
  background: linear-gradient(to bottom, #dfdfe2 1.845rem, white
  1.845rem);
}
```

(62) (styles.css) 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 left padding

```
tr.data th {
  text-align: left;
  padding-top: 0.3rem;
  padding-left: 0.5rem; }
```

(63) (styles.css)

```
tr.data th .description {
    display: block;
    font-style: italic;
    font-weight: normal;
    padding: 1rem 0 0.75rem;
    margin-right: -13.5rem;
}
```

→ target = the elements with the class set to description that are within your th elements in your .data table rows.

(64) (styles.css) your span elements now all have more specific styling, which means you can remove your span rule :)

(65) (styles.css) your dollar amounts are currently misaligned. Create a selector to target the td elements within your tr.data elements.

```
tr.data td {
    vertical-align: top;
    text-align: right;
    padding: 0.3rem 0.25rem 0;
}
```

→ "horizontally align the text to the right"

(66) (styles.css) create another selector for the td elements within your tr.data element, but specifically select the last one.

```
tr.data td:last-of-type {
    padding-right: 0.5rem;
}
```