Tutorial 3: Designing a Page Layout

Hia Al Saleh

October 14th, 2024

Contents

1 Objectives

- Create a reset style sheet
- Explore page layout designs
- Center a block element
- Create a floating element
- Clear a floating layout
- Prevent container collapse
- Explore grid-based layouts
- Create a layout grid
- Format a grid
- Explore the CSS grid styles
- Explore positioning styles
- Work with relative positioning
- Work with absolute positioning
- Work with overflow content

2 Page Layout with Floating Elements

2.1 Introducing the display Style

HTML elements are classified into:

- Block elements, such as paragraphs or headings
- Inline elements, such as emphasized text or inline images

The display style can be defined for any page element using:

```
display: type;
```

where type defines the display type.

2.2 Creating a Reset Style Sheet

A reset style sheet supersedes a browser's default styles and provides a consistent starting point for page design. The first style rule in a sheet is the display property used to display HTML5 structural elements.

2.3 Exploring Page Layout Designs

Web page layouts fall into three categories:

- Fixed layout – Size of the page and page elements are fixed, usually using pixels as the unit of measure.
- Fluid layout – The width of the page elements are set as a percent of the available screen width.
- Elastic layout – Images and text are always sized in proportion to each other in em units.

Responsive design – The layout and design of a page changes in response to the device that is rendering it.

2.4 Working with Width and Height

The width and height of an element are set using the following properties:

```
width: value;
height: value;
```

where value is the width or height using one of the CSS units of measurement or as a percentage of the width or height of the parent element.

2.5 Centering a Block Element

Block elements can be centered horizontally within their parent element by setting both the left and right margins to auto:

```
body {
    margin-left: auto;
    margin-right: auto;
}
```

2.6 Vertical Centering

Centering an element vertically can be accomplished by displaying the parent element as a table cell and setting the vertical-align property to middle. For example, to vertically center the following h1 heading within the div element:

```
<div>
    <h1>Pandaisia Chocolates</h1>
</div>
```

Apply the style rule:

```
div {
    height: 40px;
    display: table-cell;
    vertical-align: middle;
}
```

2.7 Floating Page Content

Floating an element takes it out of position and places it along the left or right side of its parent element. To float an element, apply:

```
float: position;
```

where position is none (the default), left to float the object on the left margin, or right to float the object on the right margin.

2.8 Clearing a Float

To ensure that an element is always displayed below floated elements, use:

```
clear: position;
```

where position can be left, right, both, or none.

2.9 Refining a Floated Layout

The content box model refers to the width property, which only considers the width of an element's content. Additional space includes padding or borders. The border box model, on the other hand, bases the width property on the sum of the content, padding, and border spaces. The additional space taken up by the padding and border is subtracted from the space given to the content.

2.10 Working with Container Collapse

Container collapse occurs when an empty container has no content and contains floated elements. To prevent this, use the after pseudo-element to add a placeholder element after the footer:

```
container::after {
    clear: both;
    content: "";
    display: table;
}
```

3 Page Layout Grids

A grid layout is based on rows of floating elements, where each floating element constitutes a column. The set of elements floating side-by-side establishes a row. Many grid layouts use the div (or division) element to mark distinct rows and columns of the grid.

3.1 Setting up a Grid

This is an example of a simple grid consisting of a single row with two columns:

```
<div class="row">
    <div class="column1"></div>
    <div class="column2"></div>
</div>
```

The page content is placed within the div elements.

3.2 Designing the Grid Rows

Grid rows contain floating columns. Since a grid row starts a new line within a page, it should only be displayed when both margins are clear of previously floated columns.

3.3 Defining a CSS Grid

To create a grid display without the use of div elements, use the following grid-based properties:

```
selector {
    display: grid;
    grid-template-rows: track-list;
    grid-template-columns: track-list;
}
```

The fr unit represents the fraction of available space left on the grid after all other rows or columns have attained their maximum allowable size.

3.4 Assigning Content to Grid Cells

Elements in a CSS grid are placed within a grid cell at the intersection of a specified row and column. By default, all specified elements are placed in the grid cell located at the intersection of the first row and first column. To place an element in a different cell, use:

```
grid-row-start: integer;
grid-row-end: integer;
grid-column-start: integer;
```

```
grid-column-end: integer;
```

4 Layout with Positioning Styles

To place an element at a specific position within its container, use:

```
position: type;
top: value;
right: value;
bottom: value;
left: value;
```

where type indicates the kind of positioning applied to the element and top, right, bottom, and left properties indicate the coordinates of the element.

4.1 Handling Overflow

Overflow controls how a browser handles excess content:

```
overflow: type;
```

where type can be visible (the default), hidden, scroll, or auto.

4.2 Clipping an Element

The clip property defines a rectangular region through which an element's content can be viewed. Anything that lies outside the boundary of the rectangle is hidden. The syntax of the clip property is:

```
clip: rect(top, right, bottom, left);
```

4.3 Stacking Elements

By default, elements that are loaded later by a browser are displayed on top of elements that are loaded earlier. To specify a different stacking order, use the z-index property:

```
z-index: value;
```

where value is a positive or negative integer, or the keyword auto.