NEW PERSPECTIVES

Tutorial 3
Designing a Web Page with CSS

HTML and CSS

6TH EDITION

Objectives

- Create a reset style sheet
- Explore page layout designs
- Center a block element
- Create a floating element
- Clear a floating layout
- Explore grid-based layouts

Objectives (continued)

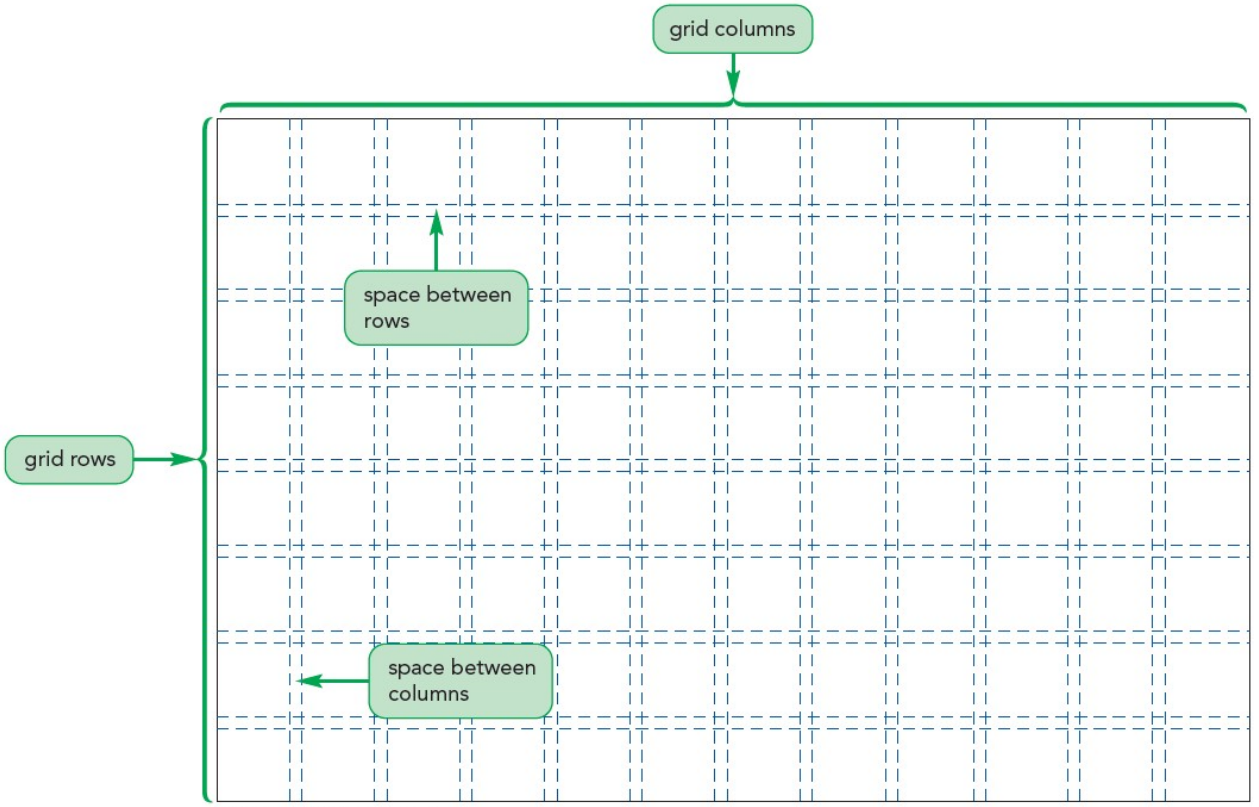
- Create a layout grid
- Format a grid
- Explore the CSS grid styles
- Explore positioning styles
- Work with relative positioning
- Work with absolute positioning
- Work with overflow content

Overview of Grid-Based Layouts

- Rows and columns form a grid
  - The number of rows is based on the page content
  - The number of columns is based on the number that provides the most flexibility in laying out the page content

Overview of Grid-Based Layouts

Figure 3-29 Page grid



Overview of Grid-Based Layouts

- Advantages of using a grid:
  - Grids add order to the presentation of page content
  - A consistent logical design gives readers the confidence to find the information they seek
  - It is easily accessible for users with disabilities and special needs
  - It increases the development speed with a systematic framework for the page layout

Fixed and Fluid Grids

- **Fixed grids** – Every column has a fixed position
  - Widths of the columns and margins are specified in pixels
- **Fluid grids** – Provides more support across different devices with different screen sizes.
  - Column width is expressed in percentages

Setting up a Grid

- A grid layout is based on rows of floating elements
- 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

Setting up a Grid

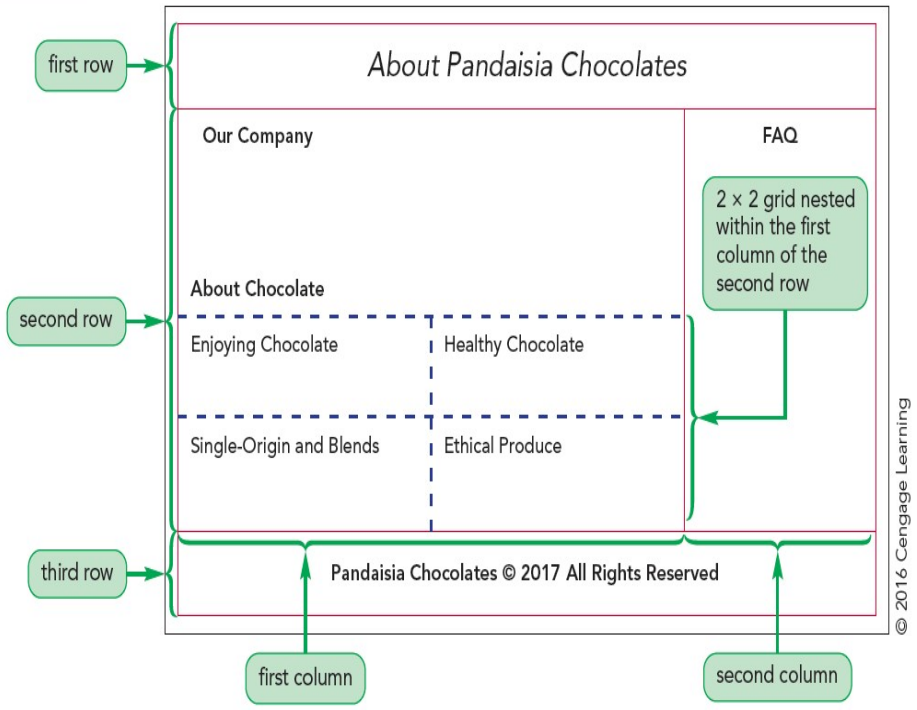
- This is an example of a simple grid consisting of a single row with two columns:

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

The page content is placed within the `div` elements

Setting up a Grid

Figure 3-32 Proposed grid layout for the About Pandaisia Chocolates page



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

```
div.row {
        clear:both;
}
```

Designing the Grid Columns

- Every grid column needs to be floated within its row
- Grid columns are placed within a `div` element having the general class name

  `class="col-`*`numerator`*`-`*`denominator`*`"`

  where *`numerator`*`-`*`denominator`* provides the fractional width of the column

Outlining a Grid

- Outlines – Lines drawn around an element, enclosing the element content, padding, and border spaces
  - `Outline-width: value;` – Specifies the width of a line.
    - Properties of *`value`* are: `thin, medium,` or `thick`
  - `Outline-color: color;` – Specifies the color of a line.
    - Properties of *`color`* are: CSS color name or value

Outlining a Grid

- `Outline-style: style;` – Specifies the design of a line
  - Properties of *`style`* are: `solid, double, dotted, dashed, groove, inset, ridge,` or `outset`
  - All of the outline styles properties can be combined into the outline shorthand property

```
Outline : width style color;
```

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;
}
```

  - *grid* – Selected elements will be displayed as a grid
  - *track-list* – Space-separated list of row heights or column widths

Defining a CSS Grid (continued)

```
section
   {
      display: grid;
      grid-template-rows: 100px auto 100px;
      grid-template-columns: 25% 50% 25%;
   }
```

Defining a CSS Grid (continued)

- **fr unit** – Represents the fraction of available space left on the grid after all other rows or columns have attained their maximum allowable size
- For example, the following style creates four columns with the dimension specified in the style rule:

```
grid-template-columns: 200px 250px 1fr 2fr;
```

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 of the specified elements are placed in the grid cell located at the intersection of the first row and first column

Assigning Content to Grid Cells (continued)

- 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;
```

where *integer* defines the starting and ending row or column that contains the content

Assigning Content to Grid Cells (continued)

```
aside
{
        grid-row-start: 2;
        grid-row-end: 3;
        grid-column-start: 1;
        grid-column-end: 2;
}
```

The CSS positioning Styles

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

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

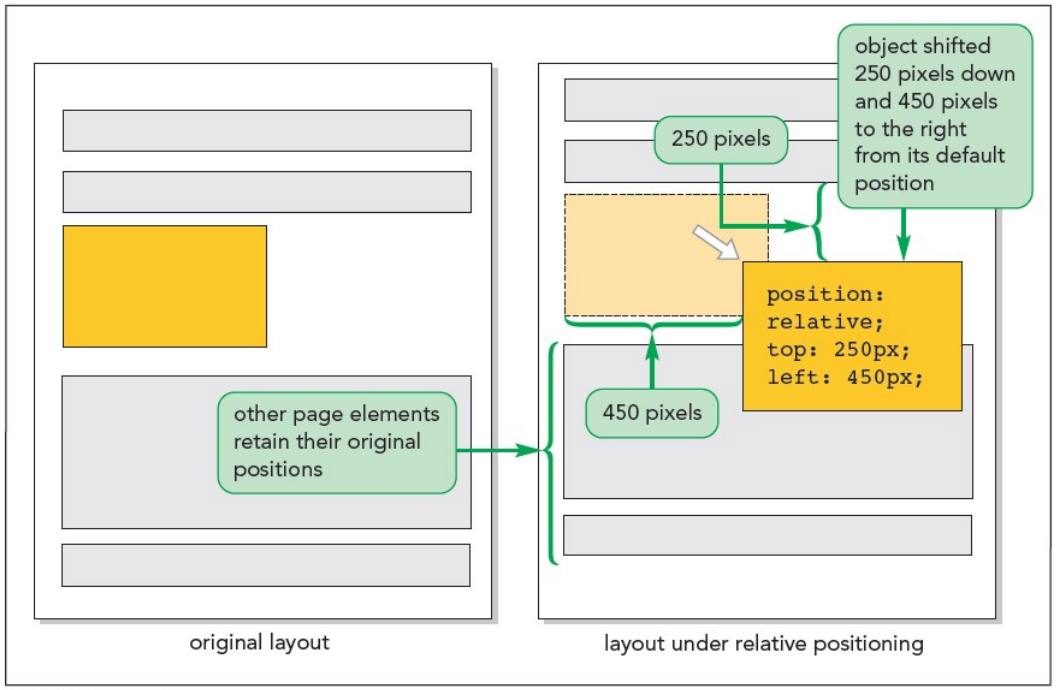
where *type* indicates the kind of positioning applied to the element and `top`, `right`, `bottom`, and `left` properties indicate the coordinates of the element

The CSS Positioning Styles

- **Static positioning** – The element is placed where it would have fallen naturally within the flow of the document
- **Relative positioning** – The element is moved out of its normal position in the document flow
- **Absolute positioning** – The element is placed at specific coordinates within containers

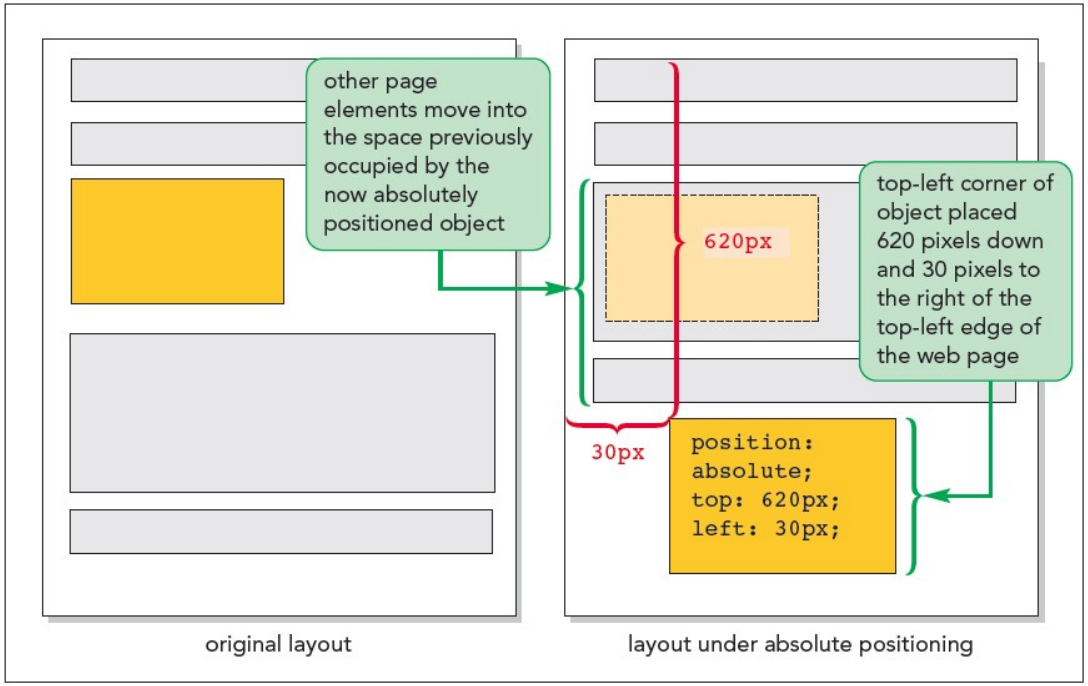
The CSS Positioning Styles

Figure 3-46 Moving an object using relative positioning



The CSS Positioning Styles

Figure 3-47 Moving an object using absolute positioning



Fixed and Inherited Positioning

- **Fixed positioning** – Fixes an object within a browser window to avoids its movement

```
footer { position : fixed; bottom : 10px; }
```

- **Inherited positioning** – Allows an element to inherit the position value of its parent element

Handling Overflow

- To specify how the browsers handle content that overflows the element's boundaries

  `overflow: type;`

  where *type* is `visible` (the default), `hidden`, `scroll`, or `auto`

  - `visible` – Instructs browsers to increase the height of an element to fit overflow contents

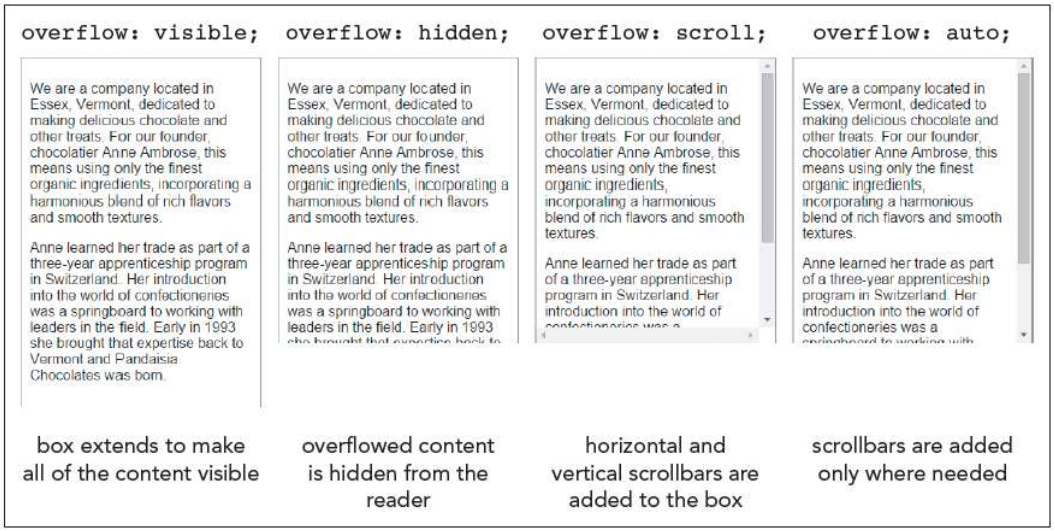
Handling Overflow

- `hidden` – Keeps an element at the specified height and width, but cuts off excess content
- `scroll` – Keeps an element at the specified dimensions, but adds horizontal and vertical scroll bars
- `auto` – Keeps an element at the specified size, adding scroll bars when they are needed

Handling Overflow

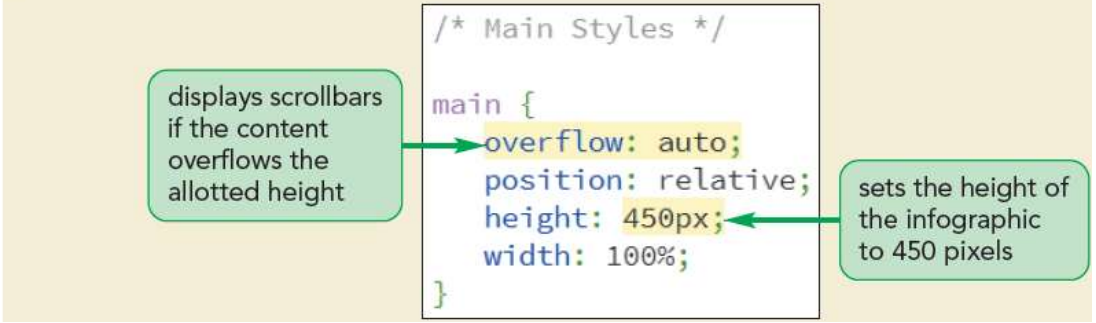
- CSS3 provides the `overflow-x` and `overflow-y` properties to handle

Figure 3-59 Values of the overflow property



Handling Overflow

Figure 3-60 Setting the overflow property

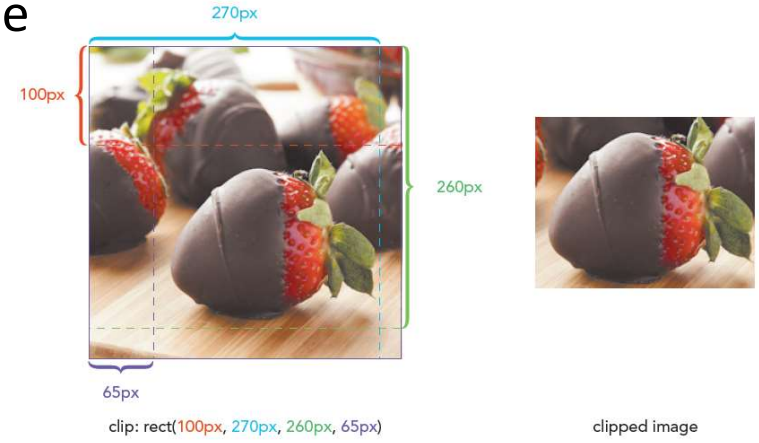


Clipping an Element

- **Clip** – Defines a rectangular region where anything that lies outside of its boundary is hidden:

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

where *top*, *right*, *bottom*, and *left* define the coordinates of the clipping rectangle



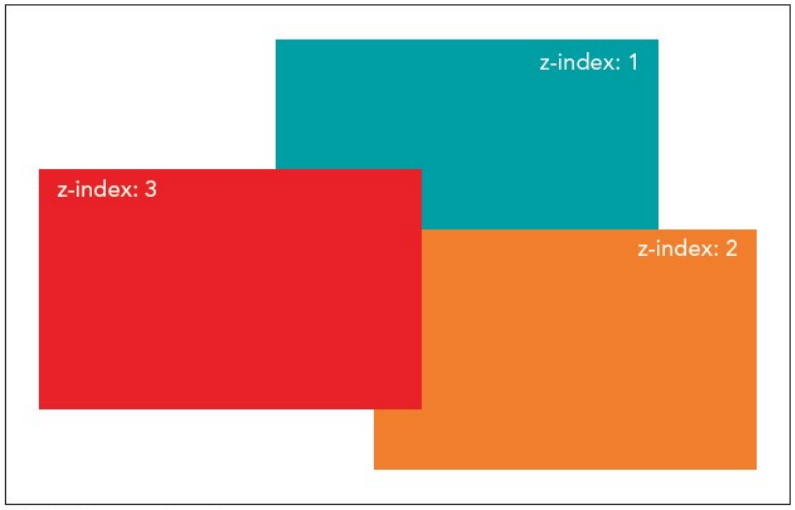
Stacking elements

- To specify different stacking order, use:

  `z-index: value;`

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

  - The `z-index` property works only for elements that are placed with **absolute positioning**



Stacking elements

- An element's z-index value determines its position relative only to other elements that share a common parent