

CMSC388A

Web Application Development with JavaScript



CSSIII

Department of Computer Science
University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

Review: Box Model

- A block-level element (e.g., p) contains four edges (top, bottom, right, and left) defining a box
- Every box has three available properties that can be adjusted to control its appearance:

border

margin

padding



Review: Box Model

- The width and height of an element with CSS is only for the **content area**. Add padding, borders, and margins to calculate the full size of an element

border

margin

padding



- Tutorial
 - https://www.w3schools.com/css/css_boxmodel.asp

Width and Height Properties

- Box width
 - left + right padding, left + right border, left + right margin, content width
 - **width** property - sets the **content** width
- Box height is determined in the same way
 - **height** property - sets the **content** height
- **Example:** WidthHeight.html

Table Formatting

- **Example:** TableFormatting.html

Custom Properties (variables)

- We define custom properties in the root element (**:root**)
 - Use **--ANAME: property value**
 - » E.g., `--my-favorite-color: red;`
 - To refer to the value use `var`
 - » E.g., `var(--my-favorite-color);`
- **Example:** CustomProperties.html

Display Property

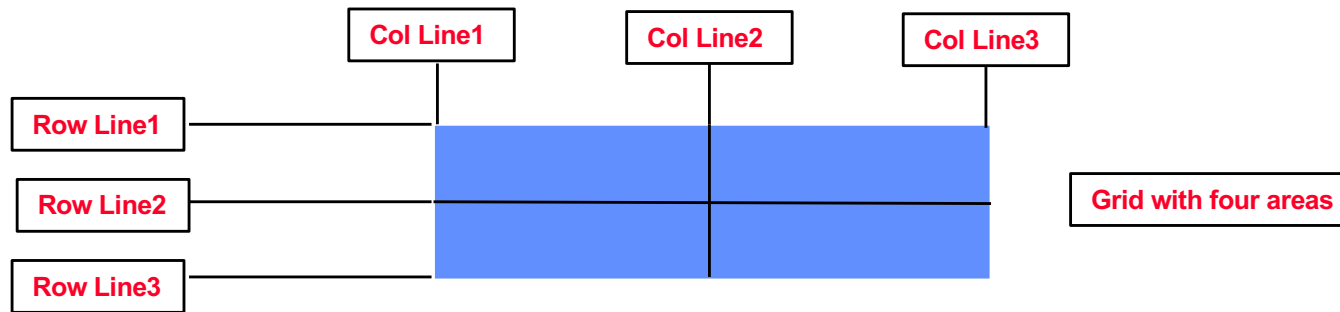
- **display property**
 - Defines the type of rendering box (e.g., block, inline) of an element
 - **Values**
 - » **inline** - causes a block-level element to act like an inline one
 - » **block** - causes an inline element to act like a block-level one
 - » **inline-block** - causes a block-level element to flow like an inline one while retaining other features of a block-level element
 - » **none** - hides an element from the page
 - » **flex** - displays element as a block-level flex container
 - » **grid** - displays element as a block-level grid container
 - **Reference:** https://www.w3schools.com/cssref/pr_class_display.php
- **Example:** InlineBlockNoneDisplayProperty.html

flex Display Property

- **flex** display property - Sets a container (e.g., div) to be a flexbox (flexible box layout) element. Elements in the container are organized in a row or column
- Terminology
 - **main axis** - defined by the flex-direction property
 - **cross axis** - perpendicular to main axis
- **flex-direction** property values → row, row-reverse, column, column-reverse
- **Example:** FlexDisplayProperty.html

grid Display Property

- **grid** display property - Sets a container (e.g., div) to use a grid
 - grid - a collection of horizontal and vertical lines
 - » Horizontal lines are called **rows**
 - » Vertical lines are called **columns**
 - » Space between rows and columns is called **gap**



grid Display Property

- Using the grid property, we can create layouts easily (in the past, using float and positioning)
- Unlike the **flex** property, adding the **grid** property will not make the elements look any different, as you will only get a one-column grid
- Several alternatives to specify the rows and columns properties
 - Positioning with grid-template areas
 - Line-based placement
- **Example:** Grid.html
 - **grid-template-columns** property – defines the number and width of columns
 - **grid-template-rows** property – defines the height of each row
- **Example:** GridTemplateAreas.html

float Property

- **CSS normal document flow/normal position** - placing of elements one after another based on the document structure and whether the element is an inline or block element
- **float**
 - Places an element on the left or right side of the container, enabling text and inline elements to wrap around it
 - **Values**
 - » none, left, right
- **Example:** FloatI.html
- **Example:** FloatII.html
 - Creating a layout using floats

Position Property

- “**position**” property can be set to:
 - **sticky** (complicated, not supported by many browsers)
 - **static** (normal position—no effect)
 - » By default, the position property is set to **static**
 - » **Example:** PositionStatic.html
 - **relative** (adjust relative to normal position)
 - » Space occupied by the element in normal flow is still retained
 - » If you make the window smaller, you will see scroll bars
 - » **Example:** PositionRelative.html
 - **fixed** (fixed position in the viewport (browser window), even when scrolling)
 - » Space occupied by the element in normal flow is still retained
 - » If you make the window smaller, you will NOT see scroll bars
 - » **Example:** PositionFixed.html

Absolute Positioning

- **absolute** positioning
 - Current element is placed in relation to the containing block
 - The containing block is not necessarily the immediate parent block
- Rules for determining the containing block:
 - Nearest ancestor of the element that has a position property value set to something other than **static**
 - If no ancestor has a position property set to something other than static, then the containing block is the body block
- **Example:** PositionAbsoluteContainingBlockBody.html
- **Example:** PositionAbsoluteContainingBlockDiv.html

z-index property

- “**z-index**” property is used to specify which elements are “in front” when they overlap (default value = 0). You can use any values (e.g., 5, 15, 20); the largest one represents the element on top
- **Example:** PositioningZIndex.html

Miscellaneous Properties

- **text-indent** - specifies the indentation to use for the first line of a block
- **line-height** - height of a line of text
- **letter-spacing** - space in-between letters
- **word-spacing** - space in-between words

CSS Templates

- CSS (theme) templates
 - Primarily for static HTML sites
 - Goes beyond CSS. A template distribution includes
 - HTML files
 - CSS files
 - Images, fonts, icons
- https://www.w3schools.com/w3css/w3css_templates.asp
- <https://www.free-css.com/>
- <https://template.co/>
- <https://stylehout.com/>
- <https://html5up.net/>
- <https://freebiesbug.com/>