**CSS WorkSheet:**

**The CSS Box Model**

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

<!DOCTYPE html>

<html>

<head>

<style>

.box {

margin: 20px;

border: 5px solid black;

padding: 15px;

width: 200px;

height: 100px;

}

</style>

</head>

<body>

<div class="box">Content Area</div>

</body>

</html>

**The display Property:**

The display property is used to change the default display behavior of HTML elements.

**Block-level Elements**

A block-level element ALWAYS starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

* <div>
* <h1> - <h6>
* <p>
* <form>
* <header>
* <footer>
* <section>

Example:

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: lightblue;

margin-bottom: 10px;

}

</style>

</head>

<body>

<div>This is a block-level element.</div>

<p>This is another block-level element.</p>

</body>

</html>

**Inline Elements**

An inline element DOES NOT start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:

* <span>
* <a>
* <img>

<!DOCTYPE html>

<html>

<head>

<style>

span {

color: red;

}

</style>

</head>

<body>

<p>This is a paragraph with an inline <span>span</span> element.</p>

</body>

</html>

**Display: none;**

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The <script> element uses display: none; as default.

<!DOCTYPE html>

<html>

<head>

<style>

.hidden {

display: none;

}

</style>

</head>

<body>

<div>This is visible.</div>

<div class="hidden">This is hidden.</div>

</body>

</html>

**The display: inline-block Value**

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

<!DOCTYPE html>

<html>

<head>

<style>

.inline-block {

display: inline-block;

width: 100px;

height: 50px;

background-color: lightcoral;

}

</style>

</head>

<body>

<div class="inline-block">Element 1</div>

<div class="inline-block">Element 2</div>

</body>

</html>

What is CSS Flexbox?

Flexbox is short for the Flexible Box Layout module.

Flexbox is a layout method for arranging items in rows or columns.

Flexbox makes it easier to design a flexible responsive layout structure, without using float or positioning.

To start using CSS flexbox, you need to first define a flex container

display:flex;

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

border: 1px solid black;

}

.flex-item {

background-color: lightgreen;

padding: 10px;

margin: 5px;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="flex-item">Item 1</div>

<div class="flex-item">Item 2</div>

<div class="flex-item">Item 3</div>

</div>

</body>

</html>

**The flex-direction property can have one of the following values:**

* row
* column
* row-reverse
* column-reverse

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-direction: column;

border: 1px solid black;

}

.flex-item {

background-color: lightyellow;

padding: 10px;

margin: 5px;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="flex-item">Item 1</div>

<div class="flex-item">Item 2</div>

<div class="flex-item">Item 3</div>

</div>

</body>

</html>

**The CSS flex-wrap Property**

The flex-wrap property specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.

The flex-wrap property can have one of the following values:

* nowrap
* wrap
* wrap-reverse

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

flex-wrap: wrap;

border: 1px solid black;

}

.flex-item {

background-color: lightpink;

padding: 10px;

margin: 5px;

width: 150px;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="flex-item">Item 1</div>

<div class="flex-item">Item 2</div>

<div class="flex-item">Item 3</div>

<div class="flex-item">Item 4</div>

</div>

</body>

</html>

**The CSS justify-content Property**

The justify-content property is used to align the flex items when they do not use all available space on the main-axis (horizontally).

The justify-content property can have one of the following values:

* center
* flex-start
* flex-end
* space-around
* space-between
* space-evenly

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: center;

border: 1px solid black;

}

.flex-item {

background-color: lightblue;

padding: 10px;

margin: 5px;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="flex-item">Item 1</div>

<div class="flex-item">Item 2</div>

<div class="flex-item">Item 3</div>

</div>

</body>

</html>

**The CSS align-items Property**

The align-items property is used to align the flex items when they do not use all available space on the cross-axis (vertically).

The align-items property can have one of the following values:

* center
* flex-start
* flex-end
* stretch
* baseline
* normal

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

align-items: center;

height: 200px;

border: 1px solid black;

}

.flex-item {

background-color: lightgray;

padding: 10px;

margin: 5px;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="flex-item">Item 1</div>

<div class="flex-item">Item 2</div>

</div>

</body>

</html>