Tutorials

HTML

CSS

Beginner

Intermediate

Class and ID Selectors

Grouping and Nesting

Pseudo Classes

Shorthand Properties

Background Images

Specificity

Display

Pseudo Elements

Page Layout

Display

A key trick to the manipulation of HTML elements is understanding that there's nothing at all special about how most of them work. Most pages could be made up from just a few tags that can be styled any which way you choose. The browser's default visual representation of most HTML elements consist of varying font styles, margins, padding and, essentially, **display types**.

The most fundamental types of display are **inline**, **block** and **none** and they can be manipulated with the `display` property and the shockingly surprising values `inline`, `block` and `none`.

Inline

`inline` does just what it says - boxes that are displayed inline follow the flow of a line. Anchor (links) and

Advanced
JavaScript

References
HTML Tags
CSS Properties

Articles

Examples

Home
About HTML Dog
Contact us
Link to HTML Dog
Site map

@htmldog

inline follow the flow of a line. Anchor (links) and emphasis are examples of elements that are displayed inline by default.

The following code, for example, will cause all list items in a list to appear next to each other in one continuous line rather than each one having its own line:

```
li { display: inline }
```

Block

`block` makes a box standalone, fitting the entire width of its containing box, with an effective line break before and after it. Unlike inline boxes, block boxes allow greater manipulation of height, margins, and padding. Heading and paragraph elements are examples of elements that are displayed this way by default in browsers.

The next example will make all links in “nav” large clickable blocks:

```
#navigation a {
    display: block;
    padding: 20px 10px;
```

```
}
```

`display: inline-block` will keep a box inline but lend the greater formatting flexibility of block boxes, allowing margin to the right and left of the box, for example.

None

`none`, well, doesn't display a box at all, which may sound pretty useless but can be used to good effect with dynamic effects, such as switching extended information on and off at the click of a link, or in alternative stylesheets.

The following code, for example, could be used in a print stylesheet to basically "turn off" the display of elements such as navigation that would be useless in that situation:

```
#navigation, #related_links { display: none }
```

`display: none` and `visibility: hidden` vary in that `display: none` takes the element's box completely out of play, whereas `visibility: hidden`

keeps the box and its flow in place without visually representing its contents. For example, if the second paragraph of 3 were set to `display: none`, the first paragraph would run straight into the third whereas if it were set to `visibility: hidden`, there would be a gap where the paragraph should be.

Tables

OK. So that was the basics. Now for something a little more advanced and rarely used…

Perhaps the best way to understand the table-related `display` property values is to think of HTML tables. `table` is the initial display and you can mimic the `tr` and `td` elements with the `table-row` and `table-cell` property values respectively.

The `display` property goes further by offering `table-column`, `table-row-group`, `table-column-group`, `table-header-group`, `table-footer-group` and `table-caption` as values, which are all quite self-descriptive. The immediately obvious benefit of these values is that you can construct a table by columns, rather than the row-biased method used in HTML.

Finally, the value `inline-table` basically sets the table without line breaks before and after it.

Be careful when using these values. Older browsers struggle with them and getting carried away with CSS tables can seriously damage your accessibility. HTML should be used to convey meaning, so if you have tabular data it should be arranged in HTML tables. Using CSS tables exclusively could result in a mash of data that is completely unreadable without the CSS. Bad. And not in a Michael Jackson way.

Other display types

`list-item` displays a box in the way that you would usually expect an `li` HTML element to be displayed. To work properly then, elements displayed this way should be nested in a `ul` or `ol` element.

`run-in` makes a box either in-line or block depending on the display of its parent.

Related pages

- Next Page: Pseudo Elements
- Previous Page: Specificity
- Span and Div (HTML Intermediate Tutorial): The bog-standard basic

elements typically displayed inline and block.

Working Examples

- Block and inline 1
- Block and inline 2
- Block and inline 3: inline-block

Reference

- display
- visibility

HTML Dog