More Display Goodness

The CSS Display Module Level 3 has added new ways to tell browsers how we want to layout and display the content on our pages.

We still have `block`, `inline`, none and other values for display that have been available since CSS 1.0 back in the 1990's.

The new features include a two-value version of the attribute.

```
.container {
  /* these are equivalent */
  display: block;
  display: block flow;
}
```

display outside: The outside container value

The `<display: outside>`, the first value in the two-element display declaration, defines the outer display type, meaning the type of the outer box container.

There are three possible values:

- Block: The element generates a box that is block-level when placed in flow layout
- Inline: The element generates a box that is inline-level when placed in flow layout
- Run-in: The element generates an run-in box, which is a type of inline-level box with special behavior that attempts to merge it into a subsequent block container. See Run-In Layout for details.

If there is a value for `<display-outside>` but no value for `<display-inside>`, the value for `<display inside>` will default to `flow` to mirror current behavior.

display inside: How the parent lays

out children

The <display-inside> keywords define the element's formatting context that lays out its non-replaced content. Possible values are:

- flow: The element lays out its contents using flow layout (block-and-inline layout). See the flow entry in Inner Display Layout Models for more information
- flow-root: The element generates a block container box, and lays out its contents using flow layout. It always establishes a new block formatting context for its contents
- table: The element generates a principal table wrapper box that establishes a block formatting context with an additional table grid box that establishes a table formatting context
- flex: The element generates a principal flex container box and establishes a flex formatting context
- grid: The element generates a principal grid container box, and establishes a grid formatting context
- ruby: The element generates a ruby container box and establishes a ruby formatting context in addition to integrating its base-level contents into its parent formatting context (if it is inline) or generating a wrapper box of the appropriate outer display type (if it is not). See CSS Ruby Layout Module Level 1 for more information

If a `<display-inside>` value is specified but `<display-outside>` is not, the element's outer display type defaults to `block`. ruby elements default to `inline`.

Run-in layouts

Using `<display-outer>` with a `run-in` value will generate a run-in box. The run-in box will merge into a block that comes after it, inserting itself at the beginning of that block's inline-level content.

This is useful for formatting content where the appropriate DOM structure is to have a headline preceding the following prose, but the desired display is an inline headline laying out with the text.

Making sure things still work

The older names for the display property have equivalents in the new two-element syntax. The equivalencies are listed in the table below:

Old Name	New Name
block	block flow
flow-root	block flow-root
inline	inline flow
inline-block	inline flow
flex	block flex
inline-flex	inline flex
grid	block grid
inline-grid	inline grid

As shown in the example at the beginning of the post we can future proof our declarations using something like this:

```
.container {
  /* these are equivalent */
  display: block;
  display: block flow;
}
```

All browsers support the first declaration. If the browser also supports the second declaration (functionally identical to the first) then it'll use it since "last valid declaration wins". If it doesn't support it, then the rule is ignored.