Playing with flexbox layout

Flexbox and css grid are a big improvement to layout. DomUI was started in 2008, and still uses a lot of "old" crud to layout pages and components. This page contains info on experiments to get DomUI's ancient stylesheets into the 21th century (a bit).

Documentation resources

Some great resources on flexbox and css grid..

- A very detailed explanation of all flexbox properties, with great graphics to explain them. The entire website is actually a great resource.
- The same site also explains css grid

DomUI experiments

The AllComponentsPage shows off all DomUI components on a single page. Since we have old and new components we have one column with new and the other column with old components. The initial version used tables, but that was a lot of work and scaled badly. The new version uses css flexbox. The code for the page is this:

```
@Override public void createContent() throws Exception {
   Div container = new Div("flex-table");
   add(container);
   container.add(new HTag(1, "Text components inside a form").css("ui-header flex-table-header"));
   container.add(new Text2F4Fragment().css("flex-table-column"));
   container.add(new Text1F4Fragment().css("flex-table-column"));

   container.add(new HTag(1, "Combo components inside form").css("ui-header flex-table-header"));
   container.add(new Combo2FFragment().css("flex-table-column"));
   container.add(new ComboF4Fragment().css("flex-table-column"));

   container.add(new HTag(1, "LookupInput components").css("ui-header flex-table-header"));
   container.add(new LookupInput2Fragment().css("flex-table-column"));
   container.add(new LookupInput1Fragment().css("flex-table-column"));

   container.add(new HTag(1, "DateInput component").css("ui-header flex-table-header"));
   container.add(new DateInput2Fragment().css("flex-table-column"));
   container.add(new DateInput1Fragment().css("flex-table-column"));

   container.add(new HTag(1, "Buttons").css("ui-header flex-table-header"));
   container.add(new ButtonFragment().css("flex-table-column"));

   container.add(new HTag(1, "Text components outside a form").css("ui-header flex-table-header"));
   container.add(new Text2RawFragment().css("flex-table-column"));
   container.add(new TextRawFragment().css("flex-table-column"));
}
```

and the css looks like this:

```
.flex-table {
   display: flex;
   justify-content: flex-start;
   flex-wrap: wrap;
}

.flex-table-header {
   width: 100%;
   flex-grow: 2;
   min-width: 200px;
}
.flex-table-column {
   flex: 1;
   min-width: 200px;
   padding: 5px 10px;
   flex-basis: auto;           // Determine size based on content
}
```

This delivers a nice responsive page where the two columns will be adjacent when there is room, but they will be shown one under the other when space gets less.

The above css can be played with using this fiddle to get some idea.

One thing to watch for: when the flex-table-column thing contains tables that are set to 100% things go wrong: these tables scale to 100% of the body making the column take all horizontal space.