Shrink-to-fit Sizing for CSS 3 Regions

Rossen Atanassov, Microsoft Corporation, August 11, 2011

CSS3 Regions Processing Model

Before we can decide what the proper model for handling shrink-to-fit sizing of CSS 3 regions is, we have to define its "processing model" and our assumptions about it. In particular, we must be clear on which layout in the hierarchy of contents is responsible for sizing, positioning and structure of layout.

Terminology

Master layout

Layout of top level HML document containing CSS3 regions.

Nested layout

HTML content redirected from its original layout into a region of the master layout.

Layout structure

The number of layout boxes created for a given content (HTML) element. Usually 1:1 but in the case of pagination for example, one content element can be represented by many layout boxes.

Fragment

One of many layout boxes representing a single content element. The simplest example of fragmentation is pagination, where many layout boxes are created for the same content element due to layout space constraints.

For the purposes of this document we assume that content element to layout box ratio is 1:N

The following three sections describe the dependencies of master and nested layouts in the processing model of regions. Each section is a subset of the following one, thus the last one describes the complete model.

Model Constraint 1 – Master layout independence

With this constraint on the processing model all decisions of sizing, positioning and layout structure are sole responsibilities of the master layout. IFrame in the current HTML/CSS is an example of this model. Here, the master layout does *not* have any control over what parts of the nested layout go into any particular fragment. Layout boxes for regions are created, sized and positioned by the master layout and then the nested layout flows into these empty layout boxes. The nested layout is dependent on the sizing of regions in the master layout.

In summary, the nested layout depends on the master layout for providing the size of its initial container block. The master layout does not depend on the content or layout of the nested content.

The current draft of CSS3 Regions assumes this model, thus, the used width of a shrink-to-fit region is 0 (different than the 300px default for IFRAME).

Features

- Performance - complete separation of master and nested layouts allows asynchronous processing of layout
- Security – Over the years this model has proven to be secured (ex. iframe)

Limitations

- Shrink-to-fit for regions is not an option.
- Empty or overflow regions are not controllable by the master layout simply because the regions layout structure is predefined by the content of the master layout.

Model Constraint 2 – Master layout dependence on content measure

Relaxing constraint 1, in this version of the processing model the decision of sizing, positioning and layout structure is still responsibility of the master layout with the exception that of shrink-to-fit regions. The master layout is allowed to query the content of nested layouts for their content measure.

Like in processing model 1, here the master layout still does *not* have any control over what parts of the nested layout go into any particular region. This means that any assumptions about what content goes into what region or how many regions will be full or empty of content are inappropriate.

In summary, the nested layout depends on the master layout for providing the size of its initial container block. The master layout depends on the content measure of the redirected content (but not the nested layout).

Features

- Performance – this model still allows asynchronous layout of master and nested layouts with the exception of content measure queries made by the master layout.
- Security – hard to speculate, but since everything is inside of layout only I don't foresee anything major.
- Shrink-to-fit around nested layouts is possible (see content measure options below)

Limitations

- Empty or overflow regions are not controllable by the master layout simply because the regions layout structure is predefined by the content of the master layout.

Model Constraint 3 – Master layout dependence on nested layout

Relaxing constraint 2, in this version of the processing model decisions of sizing and layout structure are controlled by the nested layout. The master layout is still in control of positioning. This means that we can assume knowledge of what nested contents would fall into what regions of the master document.

In summary, both layouts are dependent on their layout structure and sizing.

This is a pretty farfetched model that would require quite a bit of property extensions and changes to both content and layout models. One of the big unknowns is identity of layout boxes – these are no longer controllable by content nor will layout, thus some sort of mediator be required.

Features

- Shrink-to-fit around nested layouts is possible (driven by the nested layouts themselves)
- Empty space or overflow of nested layouts inside of regions is not an issue – layout is very adaptive

Limitations

- Performance – this model requires synchronous layout between master and content layouts
- Complexity – the current HTML model is no longer valid – consider having a table without knowing how many table cells there are.
- Security – speculatively much higher risks than the other two processing models

Shrink-to-fit around nested layouts

For the purposes of this document I am assuming that we are trying to move the CSS3 regions processing model from constraint 1 to constraint 2.

Terminology

Different specs and user agent implementers tend to use different names/terms in order to explain the same concepts related to shrink-to-fit sizing. CSS 2.1 defines shrink-to-fit sizing as a function of three variables: available width, preferred minimum width and preferred width. The formula is as follows:

min(max(preferred minimum width, available width), preferred width)

Available width

Refers to the width of the containing block's content box. This variable is computable during layout.

Preferred minimum width

The width of all flow content (note: absolutely positioned elements do not participate here) inside of a given element if laid out *without any available width constraints* and all possible break opportunities are taken. A simpler definition is – this is the longest non-breakable piece of flow content *Synonyms* – content min width, intrinsic width *Note:* this sizing does not depend on layout and can be done independently. Percentage values are usually ignored during this layout mode (currently undefined scenario by the CSS 2.1 spec).

Preferred width

The width of all flow content inside of a given element if laid out *without any available width constraints* and *only explicit breaks* are taken. *Synonyms* – content max width, intrinsic preferred width *Note:* this sizing does not depend on layout and can be done independently. Percentage values are usually ignored during this layout mode.

Content measure

Since computation of both preferred and preferred minimum width are independent of layout we can say that these are measurements of content or *content measure*. (the term is internal to Microsoft). The content measure has minimum (a.k.a preferred minimum width) and maximum (preferred width).

Height vs. Width

Unlike shrink-to-fit width, computing shrink-to-fit height is done by always taking the maximum extent of the content. However, in order to compute the content height we first have to compute the used width of the content box of the element and then layout the content with respect to it. This makes *height a dependent function of used width*. This important difference (and dependency) between height and width makes computation of used height (or the final content height) possible only during layout.

Fragmented content

The above definitions rely on the assumption that the entire content of an element is used during content measuring. When fragmentation occurs (due to pagination, multicolumn or regions) this

assumption is no longer true and measuring of content becomes dependent on layout in the general case.

Option 1 – single content measure

Same as in CSS 2.1, this option requires no changes to the current content measuring logic and would mean the entire content is measured.

Pros

Easy to implement, good performance characteristic

Cons

If the preferred minimum width of the content is at the end all proceeding regions can end up with large empty gaps.

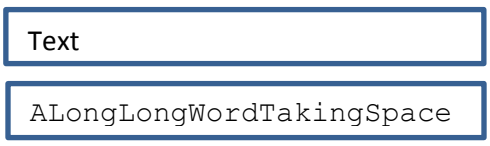
Example

```
<style>
    .text { flow-into: text; }
    .region {
        flow-from: text;
        float: left; clear: left;
        border: 2px solid blue;
    }
    br { break-after: region; }
</style>

<body>
    <div id="text">
        Text <br />
        ALongLongWordTakingSpace
    </div>

    <div class="region"></div>
    <div class="region"></div>
</body>
```

Expected result



Option 2 – content measure respecting region breaks

This minor change to the definition of *what* content must be measured would make a big difference for most practical cases that require STF regions. Instead of measuring all content and producing only one content measure, we can measure content respecting region breaks, thus producing content measure for each fragment (or piece of content spread among few fragments). In cases when the content doesn't fit a single region before the region break is reached, the two or more regions will be of the same width.

Pros

A large set of use cases will be covered.

Cons

More complex than option one in terms of building layout structures and computing content measures

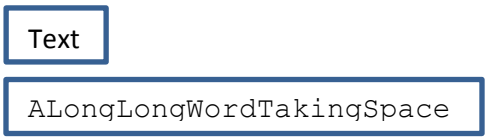
Example

```
<style>
    .text { flow-into: text; }
    .region {
        flow-from: text;
        float: left; clear: left;
        border: 2px solid blue;
    }
    br { break-after: region; }
</style>

<body>
    <div id="text">
        Text <br />
        ALongLongWordTakingSpace
    </div>

    <div class="region"></div>
    <div class="region"></div>
</body>
```

Expected result



Other practical use case is to have many elements inside a single content that are meant to spread among a number of auto sized regions (<img />
 <img />
 etc.)

Option 3 – content measure based on layout

This option will require multiple layout passes in order to determine where content will break. A naïve algorithm could be:

1. layout using the available width – results in a content break based on the available height for the region
2. measure the content up to the content break produced in step 1
3. layout again using the STF width computed using the content measure from step 2 – a new content break is produced
4. back to step 1 starting from the output of step 3

Note that the content breaks produced in step 1 and step 3 are different – this is a proof that more or less content will be measured in step 2 leading to similar problems as Option 1 above.

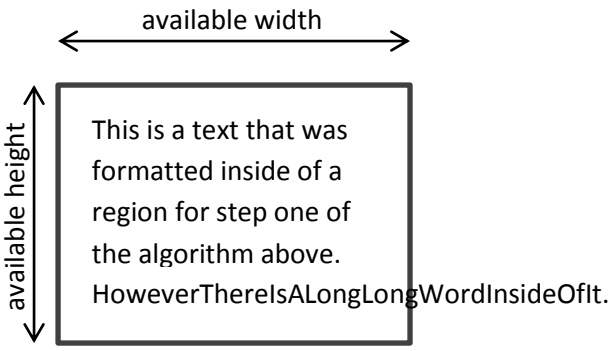
Pros

The content measure will be somewhat based on the content that goes into the regions.

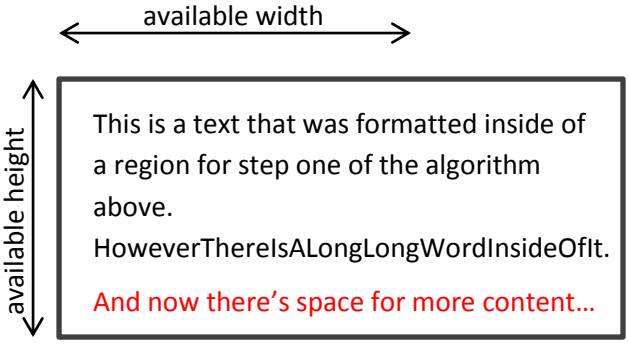
Cons

Performance and complexity of layout – multiple additional passes are required. Additionally, the content measure must be recomputed for each resizing of the region. Lastly, this is still an approximation that can be easily wrong as illustrated below.

Steps 1 and 2



Result of step 3



Use Case Discussions

Use case 1 – balancing of content between regions

Assuming that we chose STF sizing option 2, consider the following use case.

```
<style>
    .text { flow-into: text; }
    br { break-after: region; }
    .page { display: table; width: 400px; height: 600px; }
    .column { display: table-cell; flow-from: text; }
</style>

<div class="text">
    The quick brown fox jumps over the lazy dog.
    The quick brown fox jumps over the lazy dog.
    The quick brown fox jumps over the lazy dog.
    The quick brown fox jumps over the lazy dog.
    The quick brown fox jumps over the lazy dog.
    <br />
    The end.
</div>

<div class="page">
    <div id=r1 class="column"></div>
    <div id=r2 class="column"></div>
</div>
```

Problem statement

It would be easily solvable if table didn't have constraints – we'd know r1 has content up to the break. In table though (and in flexbox too) the actual space for region1 is not known until content measure is calculated for all cells….

Response

The definitions of preferred minimum and preferred width require that content is laid out without any dependency on containing layout (i.e. available width, height etc). Thus, when we layout the content of the first region fragment all content will be measured (see no width or height limitation above)– all content up to the first region break (again, we assume STF sizing option two). The remaining content for the second fragment will be the one that falls between the first and second region breaks etc.

The above leads to a generalization of what content measures would correspond to what regions as follows.

The number of content measures is defined by the number of region breaks plus one. Thus, assuming n region breaks, the first n+1 regions will get the n+1 content measures of the content and the rest will get the last one.

Further, this problem statement requires that the master layout has the knowledge of what content goes into what region (or at lease has a better approximation of that). This would be possible if we assume processing model with constraint 3. As already stated, this is beyond the scope of this proposal (see the statement of STF sizing around nested layouts) as well as (to my understanding) the scope of the current CSS3 regions focus).

Another option for solving this problem is to consider STF sizing option 3. There, I tried to explain precisely that – an approximation of what content goes into what region.