December 19, 2018

An authoring environment for adaptively formatted text

Lucas Kovar

Shuo Diao

An authoring environment for adaptively formatted text

ABSTRACT

Online ads and other text content are designed such that the text content completely fits within a defined area. However, the size of the text container may change when viewed under different viewports. For example, a responsive ad is constructed to be displayable in viewports of varying sizes. Text sized for a certain container may no longer properly fit the container as it expands or shrinks. The techniques of this disclosure provide an authoring environment that smartly adjusts text content to maintain fit within the bounds of a container element as the container expands or shrinks, or if the text content dynamically changes. Per the techniques, text fitting is enabled at authoring time (with continuous preview) and at run time.

KEYWORDS

HTML5; responsive ads; dynamic ads; viewport; HTML container; font size; innerHTML

BACKGROUND

Online ads and other text content are designed such the text content is visible and fits within a defined area, known as a container. However, in certain situations, content of the text or the size of the text container changes, e.g., when an ad is viewed in different scenarios. For example, a responsive ad is constructed to be displayable in viewports of different sizes. The layout of a responsive ad adapts to the viewport size, which includes expanding or shrinking text containers. Another example is dynamic ads, which use information in data feeds to determine what text, images, or other information gets displayed. As a result, text content shown in a dynamic ad may change when the ad is viewed in different environments. In situations where the size of the container changes or the text content changes, it is difficult to keep the text content properly fitted to the container.

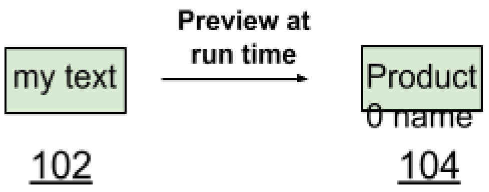


Fig. 1: Dynamic text

Fig. 1 illustrates an example of dynamic text that does not fit within its container at run (or preview) time. At authoring time, an element with dynamic text (“my text”) is created (102). When viewing the ad in a browser, the payload data, e.g., data determined by a data feed, is swapped in (“Product0 name”), which does not fit within the text container (104).

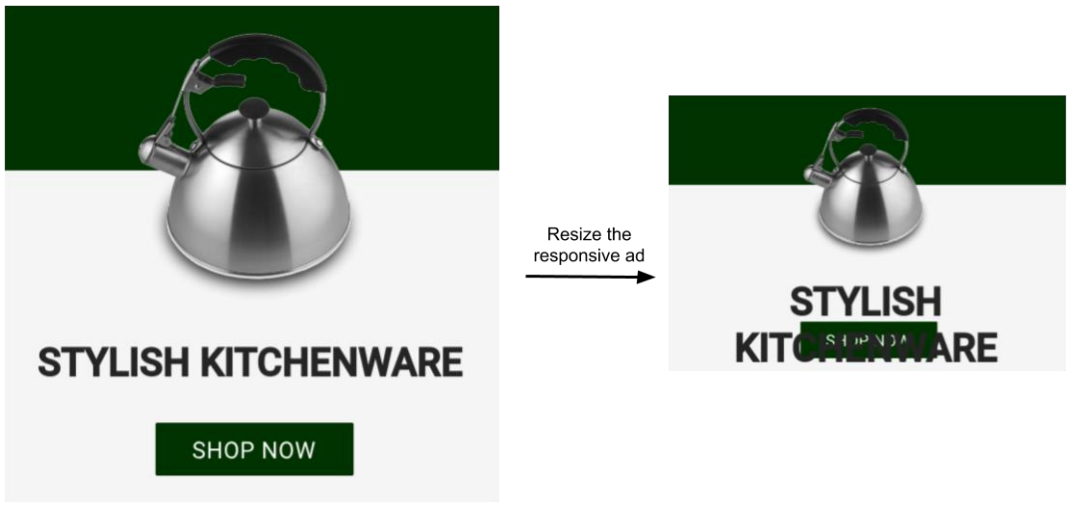


Fig. 2: Responsive ad

Fig. 2 illustrates a similar issue occurring for responsive ads, wherein flexible layouts are used to restyle ad elements when the ad size changes. If the text container has percent-based width or height then the size of the text container changes when the ad resizes; this may result in text being longer than the container. In this case, although the text content is static, a change in container size causes the text to spill out of the container. Figures 1 and 2 illustrate that text that doesn’t fit within its container damages visualization.

When previewing an ad with dynamic text in a browser, the text font size may be automatically reduced if the text does not fit within its bounds. However, since text fitting is not exposed in most user interfaces, current procedures to change default text fit settings call for manually editing undocumented attributes in code view. In addition, text fitting is only applied at run time when loading the ad in a browser, and authors are generally not able to preview text fit at authoring time.

DESCRIPTION

This disclosure presents an authoring environment for adaptively formatting and previewing text such that the text fits within its container. Per the techniques, an author can format text by direct, in-place editing of the text content without resorting to separate input fields or dialogs. Text formatting options are exposed that enable an author to determine how text is fitted. Upon change of these options, the techniques automatically (re-)fit the text. The techniques provide a live and continuously updated preview of the adaptively formatting text as containers resize or text content changes.

User interface

The techniques of this disclosure provide at least two ways to fit text to its container, e.g., reduce font size or truncate text. Both are applicable on a per-element basis. When size reduction is enabled, the font size of an element is reduced by the smallest amount that results in it fitting within the bounds of the element. A custom minimum font size can be specified, which indicates the smallest font size the text can be reduced to regardless of whether it fits within the bounds of the element. If the element has descendants, font sizes of descendant elements scale proportionately as the font size of the element is adjusted.

When text truncation is enabled, text fitting is achieved by removing as little of the text as possible and replacing the removed text by an ellipsis. If ellipsis itself does not fit within the bounds of the element then the text content is removed in its entirety from the node. If the element has descendants, a later child is truncated before its previous siblings. If both font reduction and text truncation are enabled, font size reduction is performed first, and text truncation done if the text doesn't fit at its minimum size.

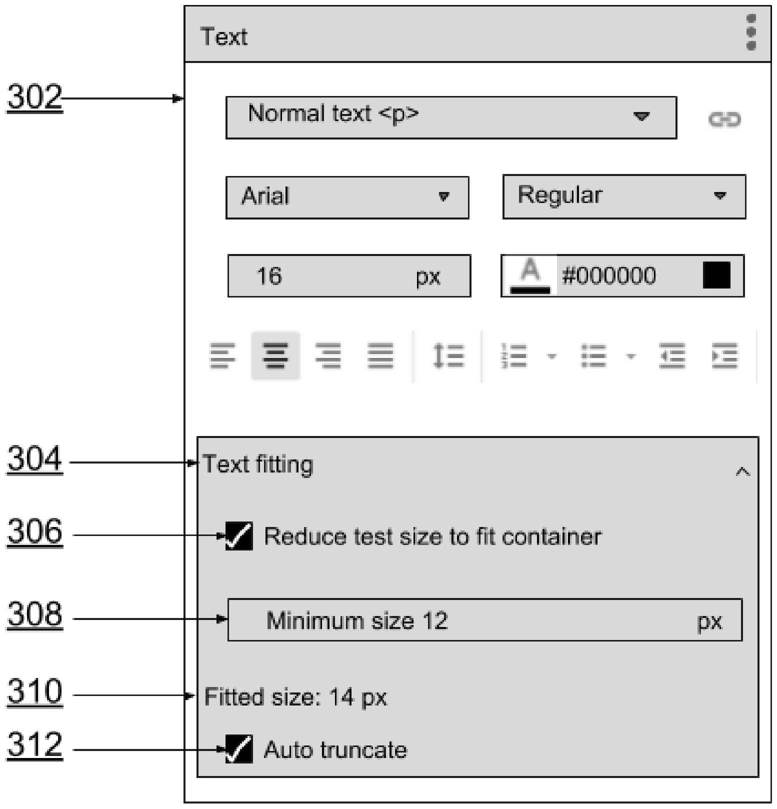


Fig. 3: Example user interface for text fitting

Fig. 3 illustrates an example user interface for text fitting. In this example, text fitting settings are viewed and edited from a text fitting section (304), which is exposed within a larger text-attribute settings panel (302). There are at least four fields in the text fitting section:

- Reduce text size to fit container (306), e.g., a checkbox indicating if font size reduction is enabled.

- Minimum size (308), an input field that specifies the smallest size for text fitting. This field is only enabled when the size reduction checkbox is checked.
- Fitted size (310), a read-only field showing the fitted font size. This field is only enabled when the size reduction checkbox is checked.
- Truncate to fit container (312), e.g., a checkbox indicating if text truncation is enabled.

Text fitting settings are edited from the text fitting section, and are applied to the elements that are selected. The text fit applied at runtime is the same as the text fit applied during preview of the document, e.g., within a browser. When the element selection on stage is changed, the text fitting user interface is updated to reflect the text fitting settings of the current selection.

Design: Text fitting attributes

```
<div data-fit-font-size fitting-truncate
    min-font-size-px="7">This text has text fitting enabled
</div>
```

Fig. 4: Text-fitting settings annotated with attributes

Fig. 4 illustrates an example of the text-fitting settings of an element annotated with attributes. Elements with size reduction enabled are annotated, e.g., with attribute ‘data-fit-font-size’, and elements with truncation enabled are annotated with attribute ‘fitting-truncate’. Custom minimum font size is specified with attribute ‘min-font-size’. In the example of Fig. 4, size reduction and text truncation is enabled, and the custom minimum font size is set to 7px. If a custom minimum font size is not specified, a default value, e.g., 10px, is used.

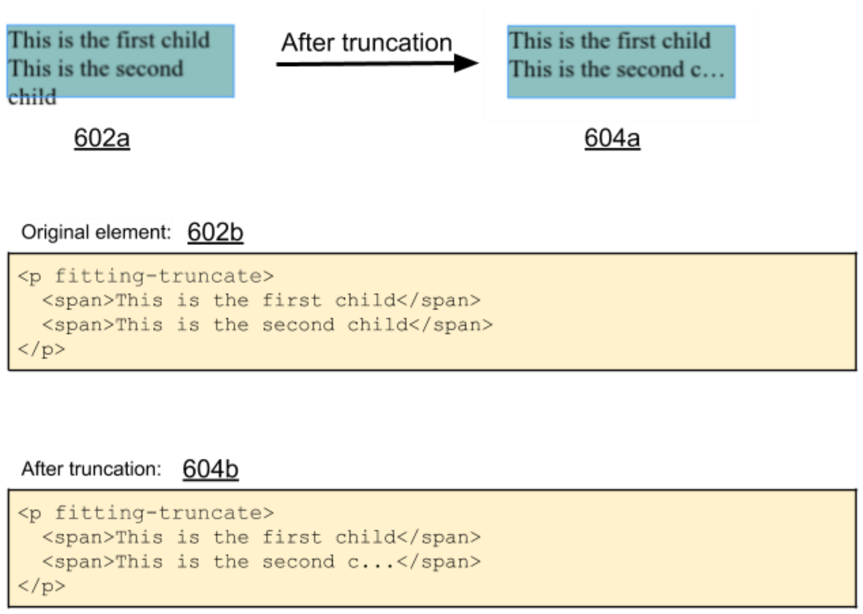
Design: Fitting an element

As explained before, elements with text fitting attributes are fitted by reducing font size and/or truncating the text. Text fitting on elements at both authoring time and run time can be achieved via a common module that holds the main logic for fitting text into its container. This module is referred to below as TextFitter.

```
<div data-fit-font-size style="font-size: 12px"
    data-override-style="font-size: 12px">
  This text is too long and should be fitted
</div>
```

Fig. 5: Element with a fitted font size

Size Reduction: Fig. 5 illustrates an example of an element with fitted font size of 12px. When size reduction is enabled, TextFitter reduces the font size by the smallest amount to make the text fit within the bounds of the element, using binary search. TextFitter sets the font size as an inline style. When a minimum font size is set, it is respected when finding the fitted size. At authoring time, the inline font size style is converted to an override style, e.g., a standard author-time-only style. If an inline font size style has been set before applying text fitting, the inline style is preserved and restored when the override font size is removed.



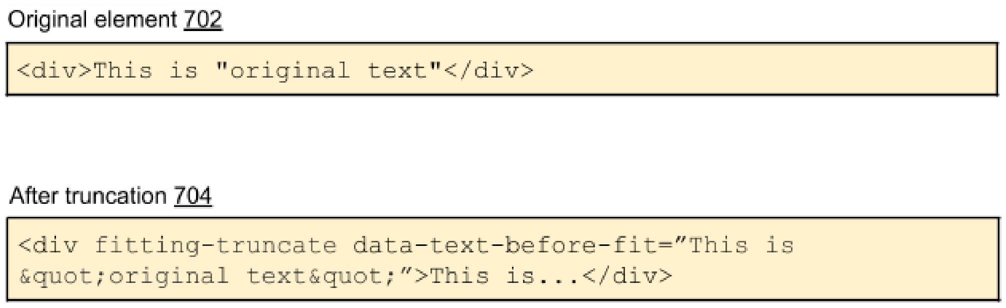
```
<p fitting-truncate>
  <span>This is the first child</span>
  <span>This is the second child</span>
</p>
```

```
<p fitting-truncate>
  <span>This is the first child</span>
  <span>This is the second c...</span>
</p>
```

Fig. 6: Run-time display of the truncation function, and the corresponding HTML elements

Text Truncation: Fig. 6 illustrates an example of run-time display of the truncation function (602a, 604a), and the corresponding HTML elements (602b, 604b). When text truncation is enabled, TextFitter removes as little of the text as possible to make the text fits within the container. If truncation is applied on an element with child nodes, truncation is done by removing text from its child nodes, starting from the last child. Truncated text may be restored, e.g., when the container is re-enlarged to its original size. This disclosure provides at least two techniques to restoring authored text, as follows.

A first technique to restore authored text after truncation:


Original element 702

```
<div>This is "original text"</div>
```

After truncation 704

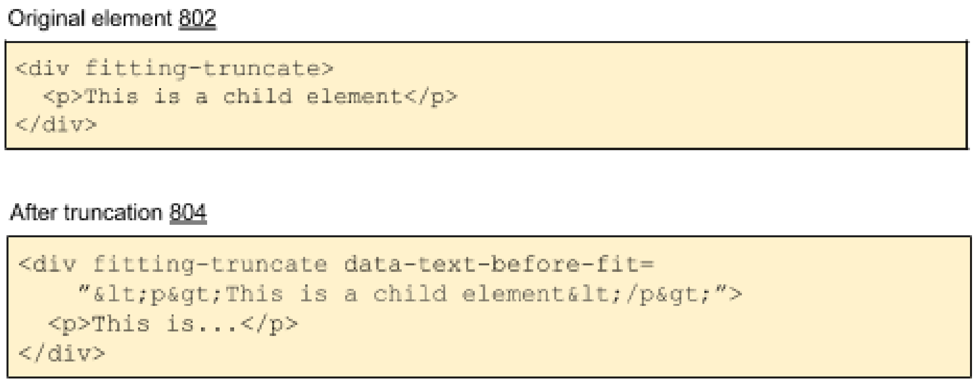
```
<div fitting-truncate data-text-before-fit=”This is
&quot;original text&quot;”>This is...</div>
```

Fig. 7: Restoring truncated text by adding an attribute to an HTML element

Fig. 7 illustrates restoring truncated text per a first technique. When text in an original element (702) is truncated, an attribute, e.g., ‘data-text-before-fit’, is added to the element to achieve truncation (704). The attribute stores the original text content with appropriate sanitizing and escaping. To restore the untruncated text, the innerHTML of the element is reset to the value held in the attribute. This method is easy to implement, and is appropriate when the element does not have any child elements, e.g., an element with just text nodes.

A second technique to restore authored text after truncation:

If an element has a nested DOM structure, the first technique described above may not be able to undo/redo accurately, due to the fact that when the innerHTML of an element is reset, a new set of child elements are created.



Original element 802

```
<div fitting-truncate>
  <p>This is a child element</p>
</div>
```

After truncation 804

```
<div fitting-truncate data-text-before-fit=
    "&lt;p&gt;This is a child element&lt;/p&gt;">
  <p>This is...</p>
</div>
```

Fig. 8: Elements with nested DOM structure do not submit to text restoral after truncation per the first technique

This is illustrated in Fig. 8, in which an original element (802) has a nested DOM structure. The text fitting routine is invoked on the element and the text content of the child element is truncated (804). Under a subsequent redo, the HTML of the element HTML is reset and another new child element is created. This causes problems since the DOM container is changed when the operation is executed and redone. This is especially a problem in authoring environments that allow the opening of an element for edit by changing DOM container, e.g., via DOM container navigation control or outliner panel. When an element is opened for edit, its original HTML is restored, which creates a new child element (P) and sets it as the new DOM container. Similar problems occur, for example, when breaking apart a group with text fitting enabled, or enabling text fitting on a parent element that has a child with text fitting enabled.

Per the techniques of this disclosure, when an element with text fitting enabled has nested DOM structure, instead of directly modifying the child nodes of an element (e.g., by truncating text or reducing font size), copies of the child nodes are made, and text fitting is performed on the cloned copies. This achieves text fitting on the element while keeping the original child nodes unchanged. Meanwhile, the original child nodes are stored in a temporary container that is set to “display: none”, and this container is appended to the element such that the original child nodes are easily restored while undoing the text fit.

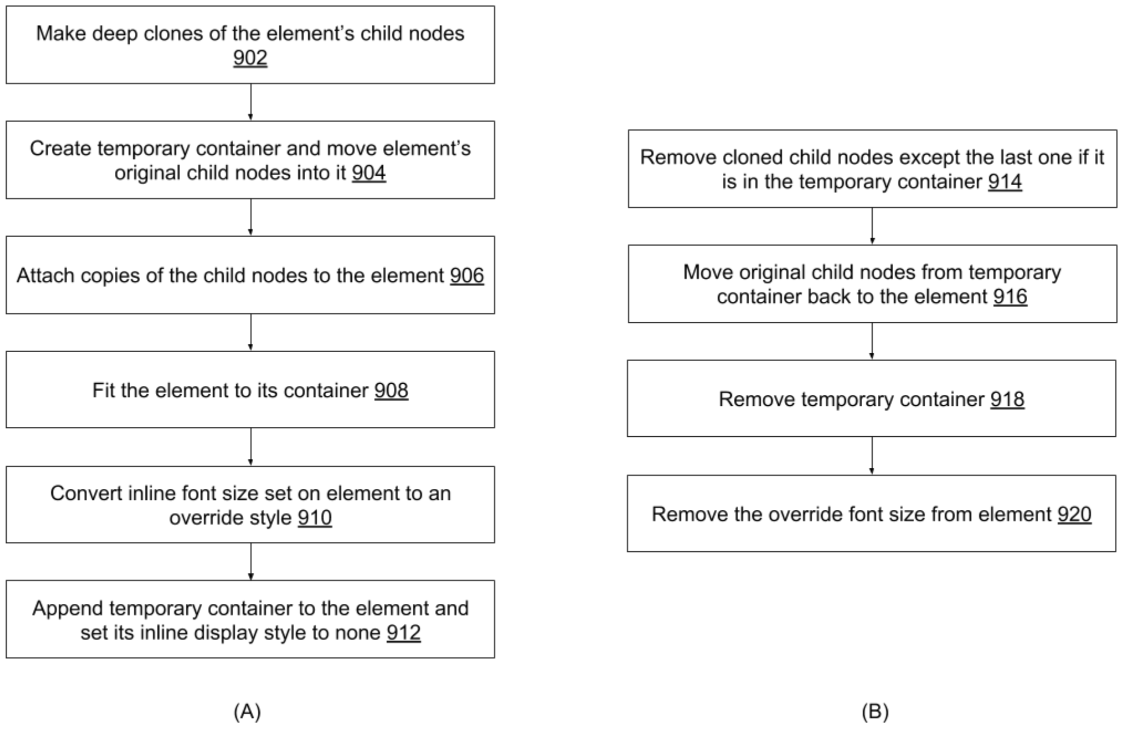
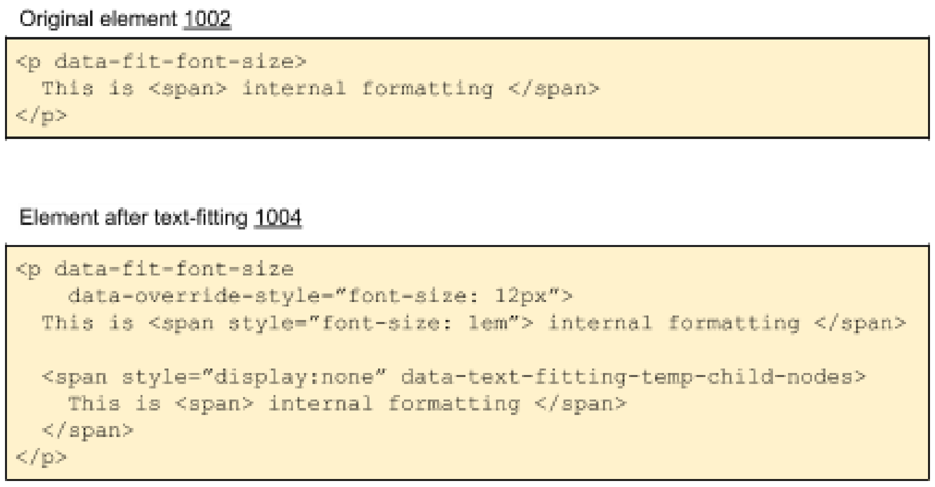

Make deep clones of the element’s child nodes 902
Create temporary container and move element’s original child nodes into it 904
Attach copies of the child nodes to the element 906
Fit the element to its container 908
Convert inline font size set on element to an override style 910
Append temporary container to the element and set its inline display style to none 912
Remove cloned child nodes except the last one if it is in the temporary container 914
Move original child nodes from temporary container back to the element 916
Remove temporary container 918
Remove the override font size from element 920
(A)
(B)

Fig. 9: (A) Text-fitting via truncation of text, and (B) its reversal, per the second technique

Fig. 9 illustrates text fitting via truncation of text and its reversal, per the second technique. To fit text to its container (illustrated in Fig. 9A), deep clones of the child nodes of an element child nodes are made (902). A temporary container is created (904), and the original child nodes of an element moved into it. Copies of the child nodes are attached to the element (906). The element is fit to the container (908). If size reduction is enabled, inline font size is added to the element and the copied child nodes. If truncation is enabled, the text content of the element text is truncated, and its innerHTML may be changed. The inline font size set on the element is converted to an override style (910). It is unnecessary to convert the inline font sizes set on the cloned child nodes, as they are discarded upon the undoing of the fit. The temporary container that holds the original child nodes to the element is appended to the element, and its inline 'display' style set to 'none' so that it does not appear on stage (912).

To undo a fit of the element to its container (illustrated in Fig. 9B), the cloned child nodes are removed (914), except for the last one if it is the temporary container. The original child nodes are moved from the temporary container back to the element (916). The temporary container is removed (918), and the override font size is removed from the element (920).



Original element 1002

```
<p data-fit-font-size>
  This is <span> internal formatting </span>
</p>
```

Element after text-fitting 1004

```
<p data-fit-font-size
    data-override-style="font-size: 12px">
  This is <span style="font-size: 1em"> internal formatting </span>

  <span style="display:none" data-text-fitting-temp-child-nodes>
    This is <span> internal formatting </span>
  </span>
</p>
```

Fig. 10: An original element, and the element after text fitting using truncation by the second technique

Fig. 10 illustrates an example of the second technique for text fitting using truncation. An original element (1002) comprises a text node and a span element. The element after text-fitting (1004) includes a temporary container that has its display parameter set to none and holds the original, untruncated text.

To guard against the possibility that text-fit settings have changed due to a previous application of text fitting, prior to applying text fitting, the fit of elements is undone in order to restore the authored font size and text content. If the text fits within the bounds of the container without any fitting then no changes have been made to the font size or text content; the fitted size field then shows the original size.

A changed text-fit setting is reflected at the user interface as follows. When the text fitting settings of an element are changed, a `TEXT_FITTING_CHANGED` event is fired. The text panel that includes the text-fitting section captures such an event and refreshes the panel UI, e.g., in case of an undo/redo of a text fitting operation.

Design: Editing from a text fitting user interface

When an edit is made from the text fitting UI, e.g., checking the size-reduction checkbox, changing the minimum font size, etc., a text-fitting event is fired. A text panel view controller captures the event and initializes a text fitting operation to update the text fitting settings on the selected elements.

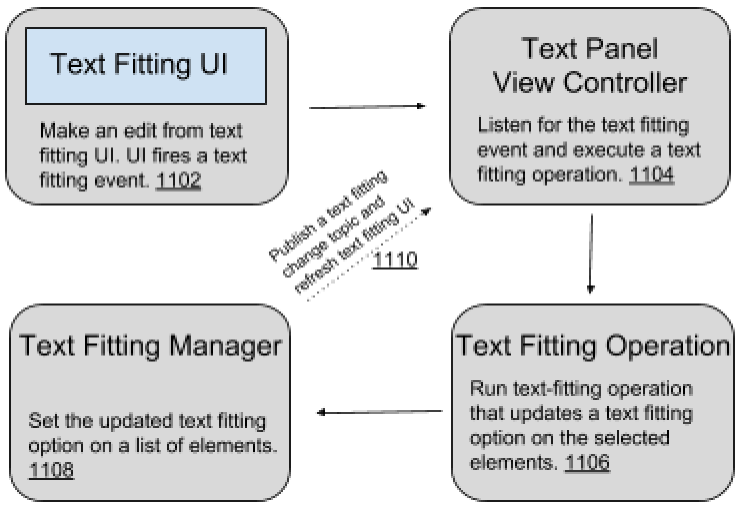
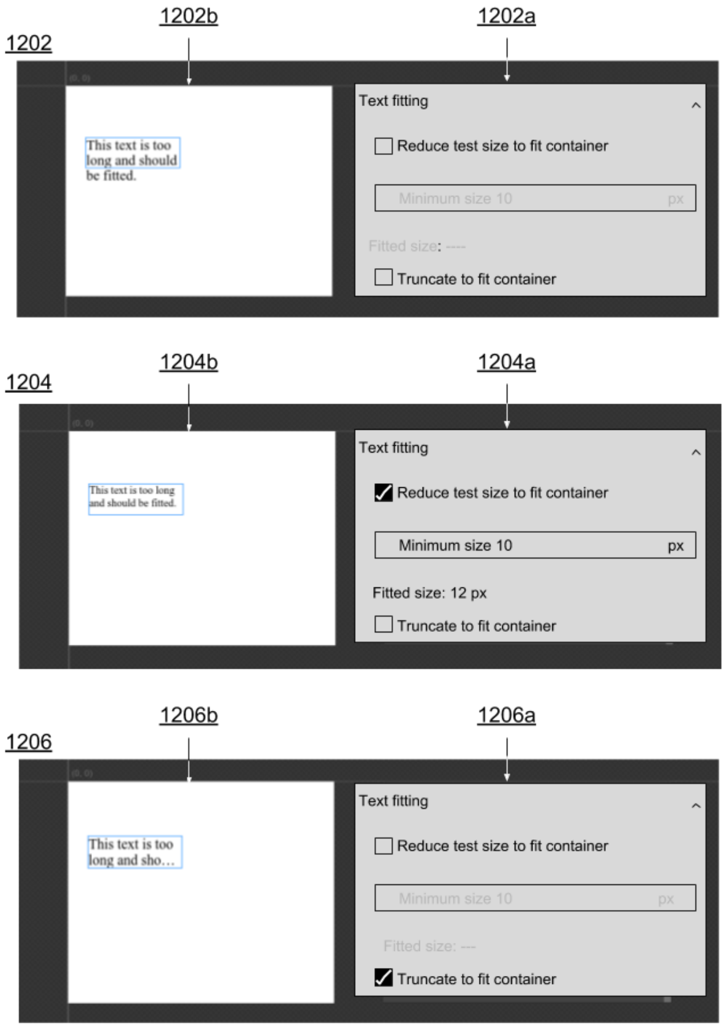

Text Fitting UI
Make an edit from text fitting UI. UI fires a text fitting event. 1102
Text Panel View Controller
Listen for the text fitting event and execute a text fitting operation. 1104
Publish a text fitting change topic and refresh text fitting UI
1110
Text Fitting Manager
Set the updated text fitting option on a list of elements. 1108
Text Fitting Operation
Run text-fitting operation that updates a text fitting option on the selected elements. 1106

Fig. 11: Event flow for editing text-fit settings

The event flow that occurs when editing a text-fit setting is illustrated in Fig. 11. An author makes an edit within a text-fitting UI (1102), and the UI fires a text-fitting event. A text panel view controller (1104) captures the event and executes a text-fitting operation. The text-fitting operation (1106) comprises updating a text-fitting option on the elements selected by the author. The text fitting manager (1108) updates the corresponding attributes on the elements based on the edited text fitting option.

Since text fitting settings have been changed, the edited element is refit with the new settings. While undoing a text fit, the old text fit settings stored prior to execution are restored, and edited elements are refitted with the old settings. Only one text fitting option (size reduction, truncation, or minimum font size) is edited from UI at a time. Thus an operation updates one option while leaving the other options unchanged. A text fitting change topic is published, and text-fitting UI refreshed (1110).



1202:
This text is too long and should be fitted.
Text fitting
☐ Reduce test size to fit container
Minimum size 10 px
Fitted size: ----
☐ Truncate to fit container

1204:
This text is too long and should be fitted.
Text fitting
☑ Reduce test size to fit container
Minimum size 10 px
Fitted size: 12 px
☐ Truncate to fit container

1206:
This text is too long and sho…
Text fitting
☐ Reduce test size to fit container
Minimum size 10 px
Fitted size: ---
☑ Truncate to fit container

Fig. 12: Selections made to the user interface and corresponding changes to the preview window

Fig. 12 illustrates examples of selections made on the user interface and corresponding changes to the preview window. Initially, text fitting is not enabled (1202), as illustrated by the absence of check marks in the text-fitting section 1202a of the text panel, and the preview / authoring panel (1202b) correspondingly shows text that spills out of its container due to its relatively large font size, e.g., 16px. Next, size reduction is enabled (1204) as evidenced by the checkmark (1204a), with minimum font size 10px, and the text fits within its container (1204b) at a reduced font of 12px. Further, text truncation is enabled (1206) as evidenced by the checkmark (1206a), and the preview / authoring panel shows correspondingly truncated text (1206b). In this manner, when text fitting is enabled, elements are fitted at authoring time.

Besides editing from text fitting UI, elements in the document that have text fitting enabled are fitted when a document is opened, because their fit is undone when the document is serialized. To achieve this, an app-level controller is added, which captures the document-added event and fits the elements. Since there are other cases where elements are fitted, a debouncer is used so elements are fitted just once within a certain interval. Using a debouncer avoids repeating the same calls.

Since text fitting is controlled via attributes on elements, it is not changed when editing media rules. In such a case, the text-fitting section is disabled. In addition, text fitting is also disabled when a text block is opened for editing.

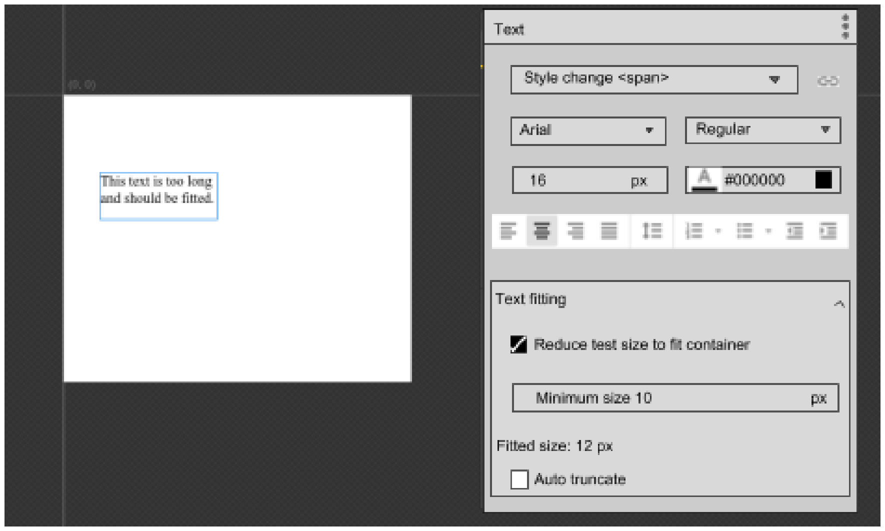
When a new text binding to innerText or innerHTML is added to an element, a default text fitting setting is added to the element, which enables size reduction and disables text truncation. To properly handle undo/redo, the default text fitting is enabled as part of binding edit operation. When the edited element is in a group definition or a group instance, text fitting is enabled on group instances. Text fitting is only enabled when new text binding is added. If text binding already exists on the element, no changes are made on the text fitting of the element.

<u>Design: Update text fitting from selection</u>

The text fitting settings shown in the UI are based on the current element selection. The text panel controller receives a changed selection, retrieves text fitting settings for the current selection from the text fitting manager, and updates the text fitting section in the text panel UI accordingly.

To ensure consistency, when multiple elements are selected, size reduction and text truncation checkboxes are examined only if they are to be examined for at least one selected element. If the selected elements do not have a shared minimum font size, the minimum size field shows a blank. A default minimum size, e.g., 10px, is used if a custom minimum size is not specified. If no elements are selected, the entire text fitting section is hidden.

If an element has been fitted, the font size fields in the text panel and text controller bar show the authored font size, and the fitted size field in the text fitting section shows the fitted value after text fitting.


This text is too long and should be fitted.

Text

Style change <span>

Arial · Regular

16 px · #000000

Text fitting

[✓] Reduce test size to fit container

Minimum size 10 px

Fitted size: 12 px

[ ] Auto truncate

Fig. 13: Authored font size and fitted font size

Fig. 13 illustrates an example of authored versus fitted font sizes. In this figure, the authored font size is 16px and the fitted font size is 12px. Users can update the authored font size, in which case the text fitting is rerun and the fitted size updated.

Design: Refit an element

For a text block with text fitting enabled, if an edit directly or indirectly change how its text fits within its bounds, text fitting may be re-run as necessary. Some scenarios where text fitting is re-applied on the affected elements are as follows.

- **Change in text style:** With text fitting enabled, text styles of an element can still be updated from text panel or text options bar. Many edits, such for font size, font family or indent (except for text color), may change how text fits within the bounds of an element; therefore text fitting is re-run when such edits are performed. However, when editing a text block, e.g., when inserting or editing a link, text fitting does not run, and hence in this case does not require a re-run.
- **Change in the width or height of an element:** When the width or height of an element changes, text fitting is rerun. This includes the following cases:
    - Editing the width or height of an element directly from a properties panel, a CSS panel or selection tool when transform control is enabled. The width or height can be changed by typing or dragging. When changing by dragging, the edited value changes continuously such that the element is fitted continuously.
    - An element has percent-based width or height, and the width or height of its parent element is changed. This also includes the case when intermediate elements have percent-based width or height, and the width/height of a more ancestral element changes.

- **Change in viewport size:** When the viewport size is changed, the direct children of a page are refitted if they have percent-based width or height. Their descendants are also refitted if intermediate elements have percent-based width or height.
- **Change in active media rule:** Styles can be overridden in media rule; therefore when the active media rule is changed, elements with text fitting enabled are refitted.
- **Change in text fitting settings:** When size reduction, minimum font size or truncation state are changed, the elements are refitted based on the new settings.

Design: Undoing text fit to show authored text content and font size

Elements with text fitting enabled are generally fitted continuously, except for the following scenarios when the original text content or/and font size is restored:

- When a text tool is selected and a text block opened for editing, the untruncated text is swapped in to show the full content for editing. Therefore, when a text block is made editable, its fit is undone to restore the authored text content. In this case, the authored font size is not restored, in order to make it convenient when working, e.g., in responsive ads with large authored font size. When editing is done and the text block is made uneditable, it is refitted by invoking the text fitting routine.
- Text fitting may also be adjusted upon DOM container change, which can be made, e.g., from DOM navigation control, by double clicking on an element, or by using an outliner panel. When entering an element with text fitting enabled from its ancestor, the element's fit is undone in order to restore the untruncated text and the original font size. If there is an intermediate element with text fitting enabled between the element and the ancestor, the intermediate element's fit is undone. When exiting an element and entering its ancestor, if the element has text fitting enabled, or there is an intermediate element with text fitting enabled, the element or the intermediate element is refit.

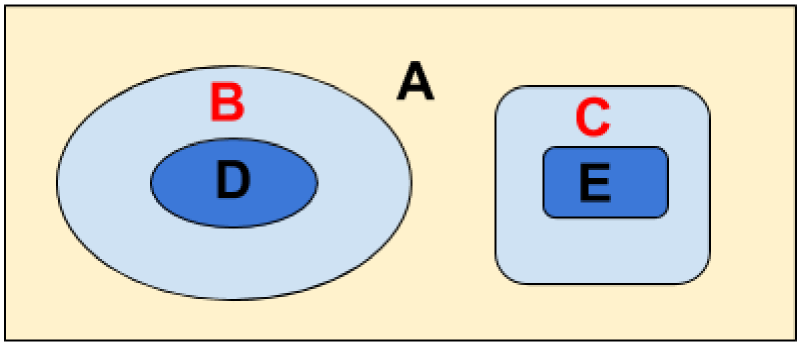


Fig. 14: Adjusting text fitting across DOM boundaries

Fig. 14 illustrates adjusting text fitting across DOM boundaries. In the scenario of Fig. 14, both element B and C have text fitting enabled. Thus,

- when entering B from A, or entering D from A, B's fit is undone;
- when exiting B to enter A, or exiting D to enter A, B is refitted;
- when exiting D to enter E (which can be done, e.g., from the outliner panel), B is refitted and C's fit is undone.

- When a document is serialized, e.g., saved to disk or while entering code view, the original text content of elements in the user document is restored. The override styles are already removed when the document is serialized. When an edit is made in code view, the elements are refitted when switching to design view.
- When the last instance of a group is removed and the group archived, the original text content and font size of the elements in the group are restored. When a new instance is added, the text fitting controller captures the element added event and refits the element.
- When an element is copied to an application clipboard, the original text content and font size is restored. When the element is pasted (added), it is refitted and preview updated.

<u>Design: Enabling text fitting on an element that has text-fitting enabled children</u>

When an element and its descendants both have text fitting enabled, it is unclear which element has text-fitting precedence. Consider the following element:

```
<p data-fit-font-size>
  <span fitting-truncate> This is a child element </span>
</p>
```

If text fitting is first applied on the parent element P, the font size of the text is reduced. However, if text fitting is first applied on the child element, the text is truncated instead. Thus a different fitting order results in a different fitting result. Since the fitting precedence is often unclear, the techniques of this disclosure provide that if text fitting is enabled on an element, it does not have text-fitting-enabled descendants. This policy leads to the following behavior:

- When the current DOM container is an element with text fitting enabled, or has an ancestor with text fitting enabled, the text fitting section is disabled.

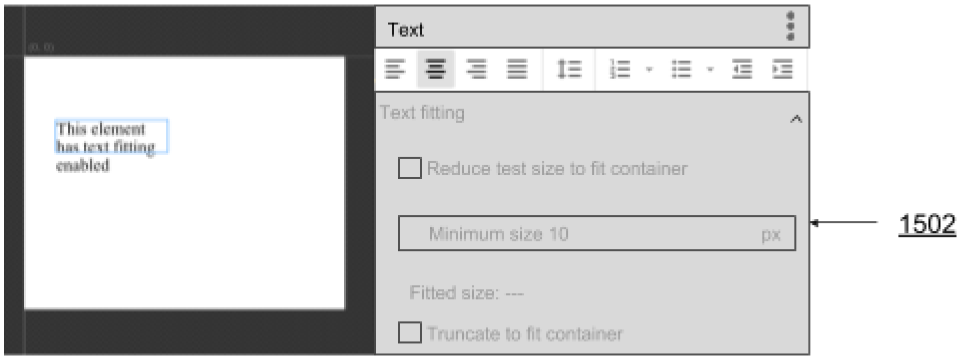
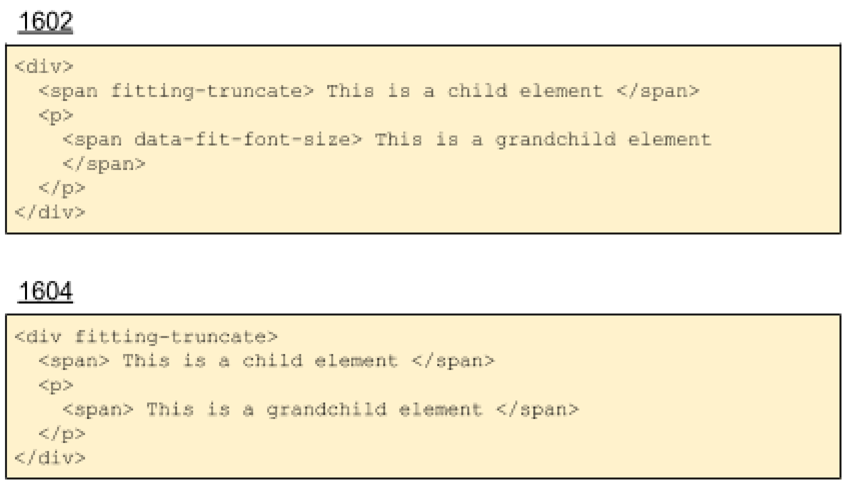

This element has text fitting enabled
Text
Text fitting
Reduce test size to fit container
Minimum size 10
px
Fitted size: ---
Truncate to fit container
1502

Fig. 15: Text fitting section is disabled when the current DOM container is an element with text fitting enabled

This is shown in Fig. 15, where the text fitting section (1502) is disabled as the DOM container is a wrapper element with text fitting enabled.

- While enabling text fitting on an element whose children also have text fitting enabled, text fitting is removed from the children.



```
<div>
  <span fitting-truncate> This is a child element </span>
  <p>
    <span data-fit-font-size> This is a grandchild element
    </span>
  </p>
</div>
```

```
<div fitting-truncate>
  <span> This is a child element </span>
  <p>
    <span> This is a grandchild element </span>
  </p>
</div>
```

Fig. 16: Text fitting is removed from children of a parent element that has text fitting enabled

This is illustrated in Fig. 16, where text fitting is enabled on a parent DIV element (1602). Text fitting is thereby removed from both its span elements (1604).

Design: Custom element to achieve text fitting at run time

In some implementations, a custom element is used to achieve text fitting at run time. The custom element is added to an element of the document that has text fitting enabled, and is removed when there are no elements with text fitting enabled. Thus the custom element may be added/removed when text fitting of elements is changed, or when elements themselves are added or removed. Similarly, the custom element is added when a binding is added in the document, and is removed when the last binding is removed.

Per techniques of this disclosure, text is automatically adjusted or formatted, e.g., by reducing the font size, truncating the text using an inserted ellipsis, etc. By ensuring that text content, e.g., a dynamic ad, a responsive ad, etc., remains fitted within its container, the techniques of this disclosure ensure that the content remains useful and informative. Per the techniques, text fitting or formatting tools are exposed in the authoring environment, with continuous preview facility. Text fitting is made available to HTML elements at both authoring and run times. Fitting or formatting is applicable to plain text and elements with nested DOM structure, e.g., text with forced line breaks or with subsections of different formatting. The techniques of this disclosure apply to any authoring tool or environment that enables authoring of dynamic text within re-sizeable containers.

CONCLUSION

The techniques of this disclosure provide an authoring environment that smartly adjusts text content to maintain fit within the bounds of its container element as the container expands or shrinks, or if the text content dynamically changes. Per the techniques, text fitting is enabled at authoring time (with continuous preview) and at run time.