The Strategy Of Advancing Mobile Web Application’s Layout And Drawing

Liankui Qiu, Panlong He, Lei Luo
School of Computer Science and Engineering, University of Electronic Science and Technology of China
comyqiu@163.com, hepanlong@yahoo.com.cn, lluo@uestc.edu.cn

Abstract

As the time of mobile Internet has come gently, the mobile applications has progressively designed by techniques based on web technology. The web technology has a lot of strong points such as high-speed developing, fast adapting, and so on. However, on the other hand, it has harsh requirements about the performance and cost of the application. This paper comes out with a strategy of advancing the mobile web applications engine layout and drawing. This engine provides the function of global refreshing and local refreshing which makes applications able to have the engine quickly rendering pages according to the specific scene. It also improves the property of applications, which, somehow, relieve the property bottleneck of development of mobile phone which use the technique based on web technology. Has been tested, Applications can use the strategy according to the specific scene, and maximum reduce the redundancy of layout and drawing.

Keywords—Mobile web application, layout and drawing, refresh interface, web technology

1. Introduction

The applications of mobile phone based on web technology can be treated as customized web pages running out of explorer’s interface. Every application is facing to specific lightweight task. It can be simply compiled by HTML, Cascading Style Sheet (CSS), and JavaScript. The special technique based on web technology makes applications have the features of light-weight, easy developed, low-coupling with Operation System and full-featured [1].

2. The Overview of Mobile Web applications’ Engine Layout techniques

The technique of layout provides an implementation mechanism for the perfect rendering of HTML elements in explorer. In the specific browsing, all the HTML elements are show as rectangles (box). The layout is placing these boxes of different sizes to the right places with some rules [2].

Box Model: It’s used to describe the rectangles produced by every element and is the principle of definition. As is shown in Figure 1, the box is optional composed with a centre content area, the padding border around it and the margin area. The whole size of the box is the total width of the element, padding area and margin area [3].

Visual Structured Model: As it described in the previous, all HTML elements produces a box, which describe the element and its property, the place it takes in the page layout. Making it structured means distribute every box in the screen, and structured model provide the ways and basis making the HTML element structured.

Containing Block: Usually, the size and place of a box are according to on its containing block. For an element, which is not the root one and has not used absolute positioning, its containing block is the content area’s edge of its nearest block-level ancestor element. The establishing of containing block builds a structured layout context for all its descendant.

Positioning Mode: CSS positioning is the extending of Cascading Style Sheet. It extends the ability of positioning HTML elements and their position and visual in 3D space. He mainly positioning modes is Static Positioning, Relative Positioning and Absolute Positioning. In Static Positioning, boxes’ borders are produced in normal way and in Relative Positioning, the borders have offset according to the containing block. Boxes’ borders have nothing to do
with the normal position’s layout in Absolute Positioning Mode.

The Engine Layout Scheme of Applications: It’s composed by box model’s management module and structured management module. The layout and typesetting are according to the DOM tree which is created by the result of HTML documentations’ analysis. During the typesetting, all the HTML elements are shown as box models. The typesetting is placing these boxes with different sizes to the right place in some rules. When achieving the layout engine, the idea of Tow allocations and Adjustment. Firstly, the box model’s management module creates element boxes for the HTML elements of the documentation object tree according to their information, and does the first layout and adjustment for the boxes’ variety of widths. Then, the structured management module will do the layout and adjustment for these boxes which include page information a second time and make the layout according to the rules established by the visual structured model.

In the box model, every HTML element generates one rectangular box. The location of each box is based on its containing block. Therefore, we can suppose that the typesetting of elements is in a big containing block. Each containing block has some virtual rows. We pull row management into inline elements’ typesetting. We put the element into the current row and record the height of the current row. If the current element’s height is bigger than the current row’s, update the height of the current row. When the rest width of the current row cannot hold an element, make a wrapping. Adjusting the row height according to the element’s, and use the sum of all the rows’ height as the height of containing block. In this way, using three levels’ management of block, row and element to organize HTML elements can reduce the complexity of typesetting.

3. The Current Problem of Layout and Drawing

When the page is loading, it needs to analyze the structure of the documentation and calculate the layout of the page combined with different styles at the same time. At last, complete the page drawing though the explorer’s rendering. Each element in DOM structure has its own box (model), all these are needed to be placed to the location where them should be according to the result of the explorer’s calculation which based on different styles (explorers’, defined by developers’). When the location, size, and other property of each kind of boxes, such as color, the font size, are certain, the explorer will draw these elements according to their own character. So, comes the pages’ content. All the process is called drawing[4].

This action will be touch off not only when the first load of the page but also the actions below.

1. The adjunction, modifying (the content) and deleting of HTML elements.
2. Using new style or modifying any property of elements that affect its outward.
3. The scrolling of the page.

There’s no doubt that a lot of operations above will be made during the mobile web applications’ development. All these operations will cause the page’s re-layout and re-drawing. Consequent drawing will lead the screen flashing.

Take the code below for instance.

```javascript
var item = document.getElementById("item");
document.body.scrollTop = 0;
item.style.width = 10;
item.style.height = 10;
```

It may cause three times of full screen’s drawing. Once is happen when assigned to body’s scrollTop, once is happen when assigned to item’s width, and the other is happen when assigned to item’s height. However, actually the application will hope that the full screen drawn would do only once.

According to the example above, we can easily think of improve applications’ performance by reducing unnecessary drawings. Here’s another code.

It may cause one full screen drawing, but actually, the application just wants to draw the local region where the element of item is. So, another way to improve performance is to pull in the local drawing mechanism.

Generally, some explorer will not enforce each act of drawing immediately, but hold a queue of drawing task. Mobile web applications will batch focus on the tasks according to actual needs. This is surely an ideal result, but neither pc explorer nor embedded explorer can make it perfectly. And in the embedded environment, having this complex intelligent judgment during each drawing will surely heavy the engine’s burden, and drop its running performance. Therefore, it’s not a completely solution at present.

However, some other explorer engine came up with another idea of solving problems. So long as it’s difficult for the engine to do the intelligent judgments, we can provide switch of full screen drawing for applications. The applications developers can decide when to use the full screen drawing according to the specific scenarios. Nokia widget provides two interfaces to application developers. One is prepareForTransition, the other is performTransition. The function of prepareForTransition will stop Transition. The function of prepareForTransition will stop all the layout s and drawings caused by applications until
meeting `performTransition`, the engine will do the full screen drawing once [5].

![Figure 2 Nokia layout and drawing flow diagram](image)

This scheme, to a large extent, solve the problem comes out at the beginning of the text. But there’re still two problems. In the complex scenarios, redrawing still exists. Take the code below for instance.

```javascript
function transitionA(){
    window.prepareForTransition();
    ................
    window.performTransition()
}
function transitionB(){
    window.prepareForTransition();
    ................
    transitionA();
    window.performTransition()
}
```

In this code, `transitionB` uses a pair of control function to control the drawing of page, but its process `transitionA` is called. As the code is, pairs of drawing function are used in `transitionA` to control full screen drawing. When this happen, the engine will do the full screen drawing twice, and it’s not fit the expectance of the application. Besides, this scheme still not comes up with a solution to solve the local drawing.

4. The Advanced Strategy of Layout and Drawing

Although the scheme from Nokia does not solve all the problem, it provides a very good idea which gives the right of layout and drawing to developers who are mostly familiar with the scenarios, so that solve the bottle neck of the engine’s control of layout and drawing. Following this idea, we come up with a more comprehensive scheme of solution.

![Figure 3 Advanced layout and drawing interface flow diagram](image)

- Pointing to the first problem of Nokia’s scheme, we propose a way to run the most outside control switch but ignore all the drawing switches inside, as is shown in Figure 3.

  In this way, we can maximum control the occasion of screen drawing.

- As it described in the overview of Layout techniques above, we know that DOM tree’s element are shown as a rectangle (Box). After every layout, the height and width of each box is fixed. So, in the precondition of not changing the other and the local boxes’ location, if we will do something to the element object, we just need to distribute and draw in its element’s box, but not the full screen. The engine’s drawing is based on layer, first draw the lowest layer then above, and the top come the last. When have drawing in some special element’s rectangle, the engine will do the things below.

  1. Get the height and width of the rectangle according to the element object.

  2. Make all the elements object which in the lower layer than this rectangle certain in turn, and make sure these objects’ region’s intersection and clip region.

  3. Draw the element object of the lowest layer which needs local drawing.

Combined with the experiences above, mobile web application develop engine gives a interface of local layout and drawing, `performTransition(elementObject)`, the `elementObject` in it is an element object which need a local layout and drawing.
Figure 4 Completely layout and drawing flow diagram

The engine provides developers the interface of layout and drawing, and the developers should abide by the rules below. (Suppose the element object of local layout and drawing is elementObj)

1. The interface of layout and drawing cannot be used if the width and height of elementObj or the value of top, left, right, bottom are changed, because the change of these property will lead to the change of other element objects’ rectangle’s location. Only draw elementObj’s rectangle cannot show the page’s view correctly. At this time, the correct value of elementObj is the wrong elementObj’s father’s containing block.

2. For other situation, we just need the local layout and drawing of elementObj’s rectangle.

5. Test

It takes a reading-software for instance, testing the promotion of the performance by using the interface of global layout and drawing and the one of local.

In this paper, we take Dopod D600 which is based on Windows Mobile platform as an example. This phone's performance parameters are shown as below:

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone model</td>
<td>Dopod S900C</td>
</tr>
<tr>
<td>Screen parameter</td>
<td>240×320 pixel</td>
</tr>
<tr>
<td>Operating system</td>
<td>Windows Mobile 5.0 Professional</td>
</tr>
<tr>
<td>CPU frequency</td>
<td>200MHz</td>
</tr>
<tr>
<td>Memory</td>
<td>64MB ROM+64MB RAM</td>
</tr>
</tbody>
</table>

Testing scene I
Click ‘love’ in index page (Figure 5), and turn to the page of channel and booklist (Figure 6). The application loads the page of channel and booklist in iframe window first, and then makes the focus to the corresponding options, and updates the menu option.

Table 2. The result of testing scene I

<table>
<thead>
<tr>
<th>The use of interface</th>
<th>Global drawing (times)</th>
<th>Local drawing (times)</th>
<th>The time for drawing (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not used</td>
<td>6</td>
<td>0</td>
<td>3891</td>
</tr>
<tr>
<td>Only use the global one</td>
<td>2</td>
<td>0</td>
<td>1558</td>
</tr>
<tr>
<td>Use both according to reality</td>
<td>1</td>
<td>3</td>
<td>1186</td>
</tr>
</tbody>
</table>

Testing scene II
In the page of channel and booklist (Figure 6), press the keys of up and down to move the focus of booklist.

Table 3. The result of testing scene II

<table>
<thead>
<tr>
<th>The use of interface</th>
<th>Global drawing (times)</th>
<th>Local drawing (times)</th>
<th>The time for drawing (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not used</td>
<td>5</td>
<td>0</td>
<td>3362</td>
</tr>
<tr>
<td>Only use the global one</td>
<td>1</td>
<td>0</td>
<td>752</td>
</tr>
<tr>
<td>Use both according to reality</td>
<td>0</td>
<td>2</td>
<td>531</td>
</tr>
</tbody>
</table>

Testing result
As is shown in table 2 and table 3, we can find that legitimately use the two interfaces of layout and drawing which are provided for application developers by the engine, can to a large extent reduce the times of layout and drawing, improve the running speed, and advance users’ experience.

6. Conclusion

This text deeply analyzes how to advance mobile web applications’ layout and drawing. Focus on the character of mobile web application, comes out with a set of feasible solution. The mobile web application engine used this scheme, has already used in developments on phone, and get good effects. It’s specifically reflected in directions bellow. Full screen and local layout and drawing interfaces are given and avoid the engine have complex algorithm of layout and intelligent judgment of drawing, and reduce the engine’s burden, improve the running efficiency. Applications can use the two interface according to the...
specific scene, and maximum reduce the redundancy of layout and drawing, greatly reduce the flashing and other performance problems caused by it.

6. References


