Prebid Skin Implementation Guide

A - Build files

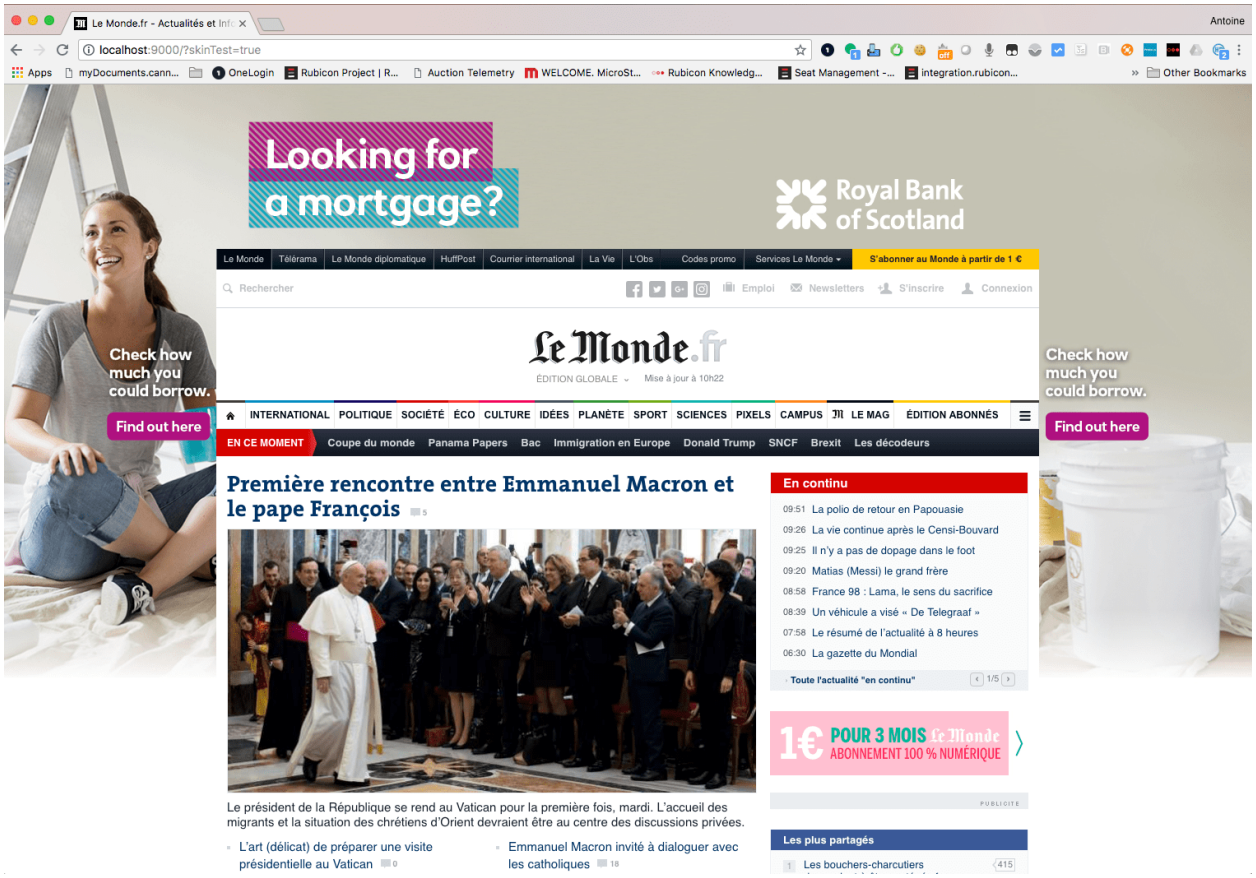
The Prebid Skin file is hosted on Akmai CDN:
https://secure-assets.rubiconproject.com/utils/prebidSkin/prebidSkin.min.js

The unminify version is hosted on:
https://secure-assets.rubiconproject.com/utils/prebidSkin/prebidSkin.js

The code is hosted on our private Github but could be explained if needed.

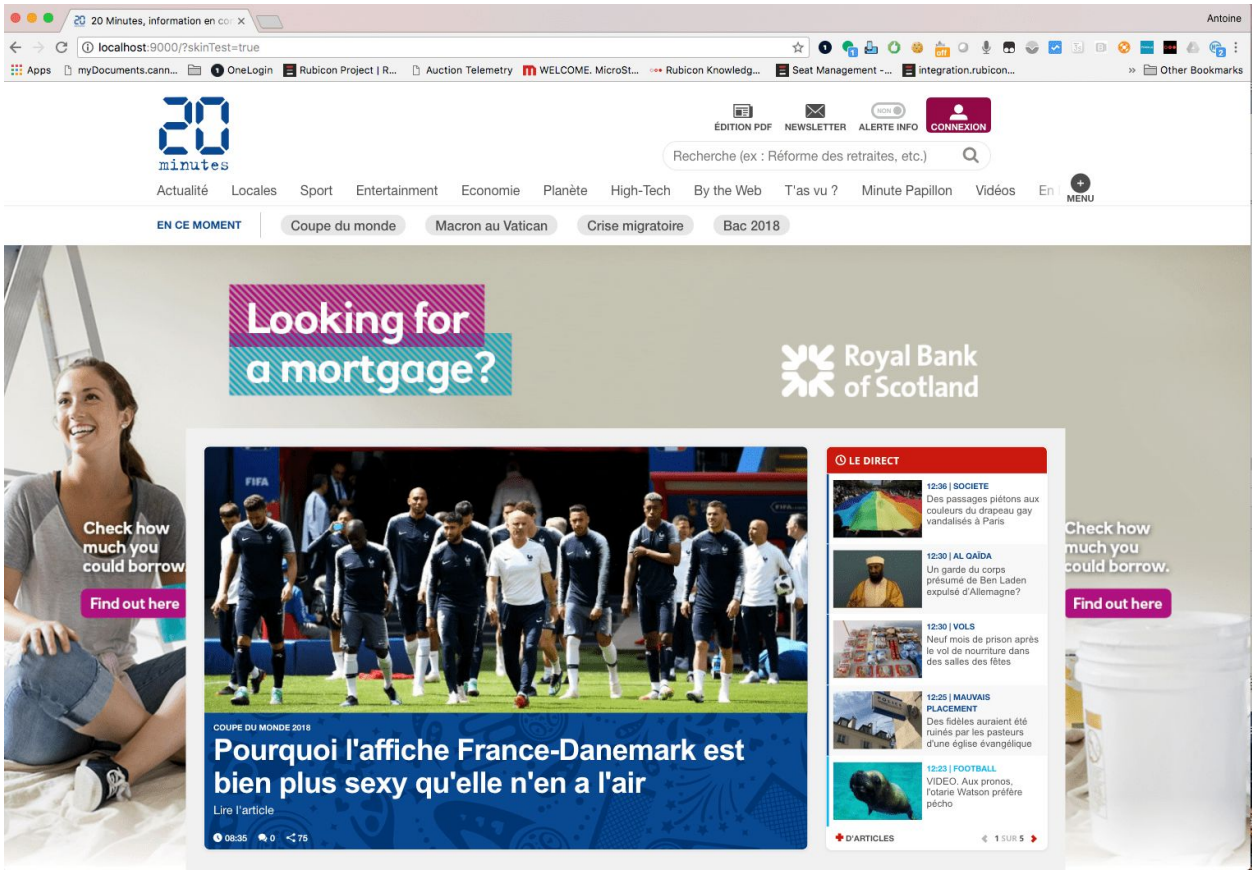
LeMonde (body wrapping)

This integration is an example of using the resize mode with LeMonde website.



20Minutes (content wrapping)

This integration is an example of using the ratio with maxRatio (0.9) mode with 20Minutes website.



B- Examples

```
<script>
var adUnits = [
    {
      code: '/4362169/Ad_test',
      sizes: [[300,250]],
      renderer: {
          url: './prebidSkin.min.js',
          render: function(prebidBid) {
              skinOverlay.renderAd({
                  fullBids: prebidBid,                        // required, please don't edit!
                  insertionMarker: "#habillagepub",      // required, #div, .class or body
where you want to insert skin
                  contentWidthMarker: "#en_ce_moment",    // required, #div, .class or
```

```
body matching the size of the content
                insertionType: "ratio",              // required, try resize or ratio
                maxRatio: 0.9,                       // optional, use with insertionType ratio
and define maximum ratio scale
                header_height: "250",                // optional
                creativeWidth: "1800",               // optional
                targetScroll: true,                  // optional, scrollable or not
                custom_style_iframe: "",             // optional if display not correct
                custom_style_page: "",               // optional if display not correct
                iframe_url: "",                      // optional
                click_url:"https://testAntoine.fr",  // optional
                debug: true                          // optional
            });
        }
    },
     bids: [{
              bidder: 'rubicon',params:{
              accountId: "14062",
              siteId: "70608",
              zoneId: "335918",
          }
        }]
  }];
</script>
```

`@trystring` : is essentially a string (bit of text) used as a query selector.
You must use operators such as `# . <>` then the skins code will try and use convential DOM look up methods.
So, to select an `<element>` with an ID of “foo”, you’d use (`#foo`).
So, to select an `<element>` with an Class of “foo”, you’d use (`.foo`).
or with a class of foo `.foo`:
If several elements are using the same ID or Class, the first one will be selected.

Here is the list of the parameters and their options:

DataParam	Type	Default	Notes

fullBids	@object	bids	required. The bid containing the creative to display. No change or edit allowed!
insertionMarker	@trystring	null	required. The dom element where the div that insert skin and pushes site content down is created.
contentWidthMarker	@trystring	null	required. Used by the renderer as a basis for the width of the website content area. Content area is the exciting part of the website.
insertionType	@string,	null	required. A string that represents how the skin will be integrate, possible value are ratio, resize.
maxRatio	@int	1	optional, maximum ratio scale, will activate also the smart ratio feature.
header_height	@int	250	optional, override the standard config creative width (250px) or special site architecture
creativeWidth	@int	1800	optional, override the standard config creative width (1800px).
custom_style_iframe	@string	null	optional, pass in valid CSS, and it will be appended to the iframe document as style property.

custom_style_page	@string	null	optional, pass in valid CSS, and it will be appended inside a <style> elem in the <head> of the main page document.
targetScroll	@bool	false	optional, if the creative need to be scrolled.
iframe_url	@string	null	optional, use custom url for the iframe. Useful for protecting the main page if the iframe domain is different from delivery page one, as cross domain will block any changes attempts on the main page from the iframe or creative.
click_url	@string	null	optional, allow to track click on the skin creative.
debug	@bool	false	optional, allow to show a test creative. Must be removed in production.

Iframe

If you decide to use the “iframe_url”, please use the following code.
Hosting the iframe on a domain different from the page could be useful on the open auction if you want to protect your website and make sure the creative won’t be able to make any edit on the main page.

```
<html>
   <head>
```

```
      <script type="text/javascript">
         var test = function(e) {
            if (e.data.message !== "render"){
            return;
            }
            document.write(e.data.html);
            var script2 = document.createElement("script");
            var code = "var sendMessage = function(){
parent.window.postMessage({message: 'click', }, '*');};
window.addEventListener('click', sendMessage, false);";
            var inlineScript = document.createTextNode(code);
            script2.appendChild(inlineScript);
            document.getElementsByTagName('head')[0].appendChild(script2);
         }
         window.addEventListener('message', test, false);
      </script>
   </head>
   <body></body>
</html>
```

C- Integration

1- Add the adUnit to the page:

Add the new adUnit with the renderer part and target it on a DFP slot, do notice that this adSlot could be a size different from the skin adUnit size and will be resize to 1x1 if the prebid creative is selected.

2- Find the skin settings best for LeMonde:

This walkthrough will assume you have a basic understanding of concepts and technologies such as web browsers, HTML, CSS, and other terms such as the DOM. If something is written in monospace, it's code related.

A- Some technical details about the prebid Skin:

The current prebid skins approach works by inserting an iframe with ads (the skin) as a background absolute element to the publisher's site with a low z-index. The renderer of skins is made inside the iframe and based on the following specs:
https://secure-assets.rubiconproject.com/utils/prebidSkin/ressources/SkinsSpecs.png

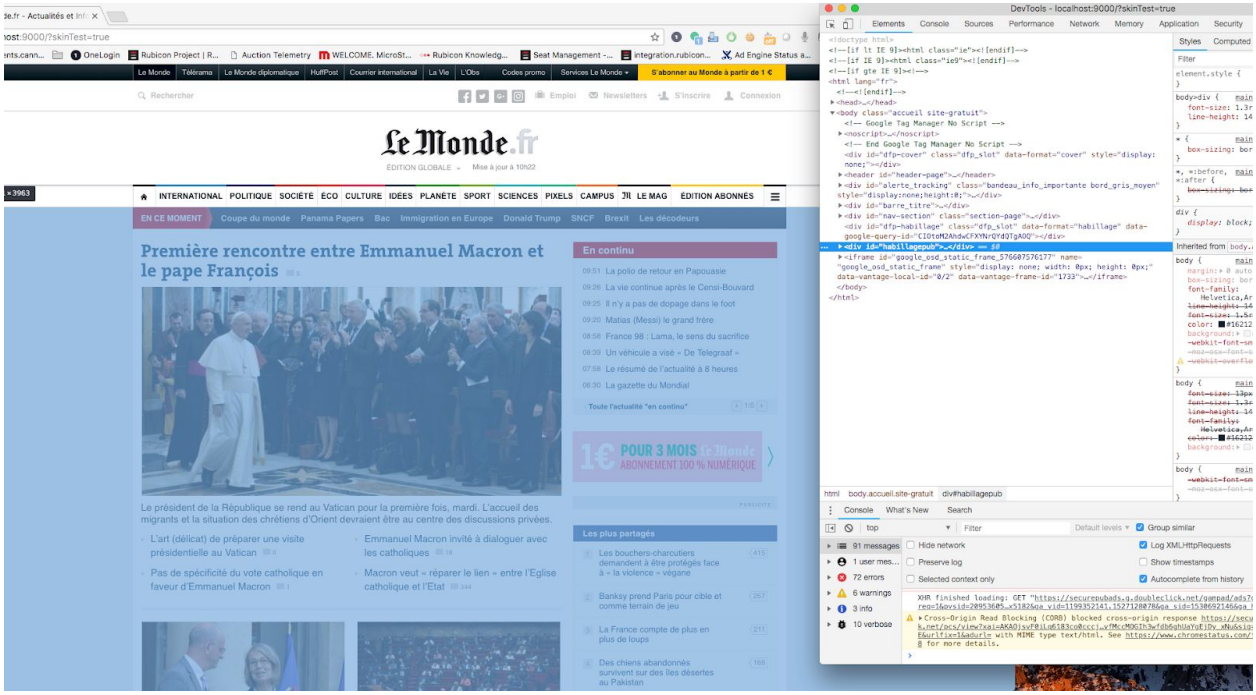
Do notice that the skin will work with other creative size if the specs ratio is respected.

If the publisher want to work on open auction but protect it website structure, using a cross domain will block any attempts to edit the main page by the creative.

B- Find the insertionMarker:

The insertionMarker will be the element wrapped by the skin. You can use <*body*> to embed the whole site or you can decide to show an header. We will need to push to content down by an appropriate amount done by inserting an invisible element with a height applied to it.

To find the marker just open the developer console and hovering over elements until funding the correct one. Here an example for LeMonde where #habillagepub seem perfect.

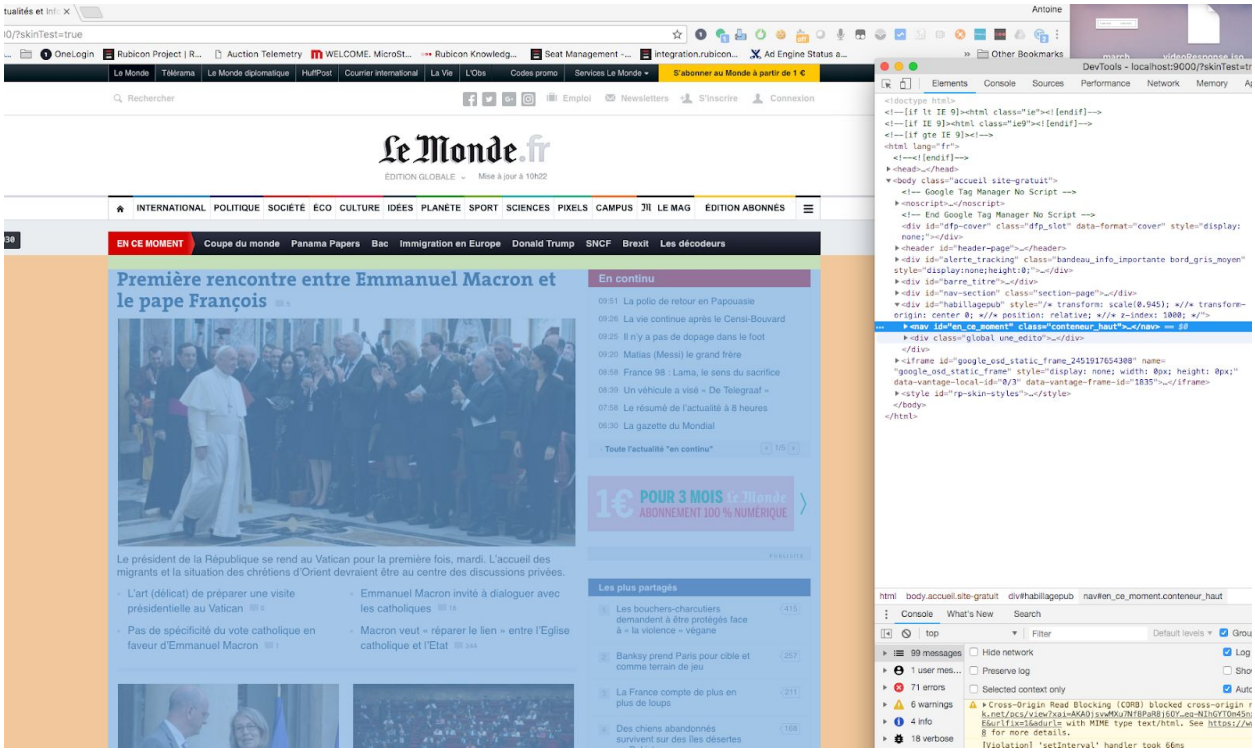


C- Find contentWidthMarker

To adjust the size or ratio of the site, we need to tell the skins renderer what the content element is. The content element is where the interesting part of the website layout is, and most all websites follow a similar pattern of a central area (content).

The reason we have an *elementmarker* is to use the width of that page element as a ratio by which we rescale or resize the skin and the page.

On our page LeMonde the correct content are size seem to be defined by #en_ce_moment or #global



D- Find the best insertionType:

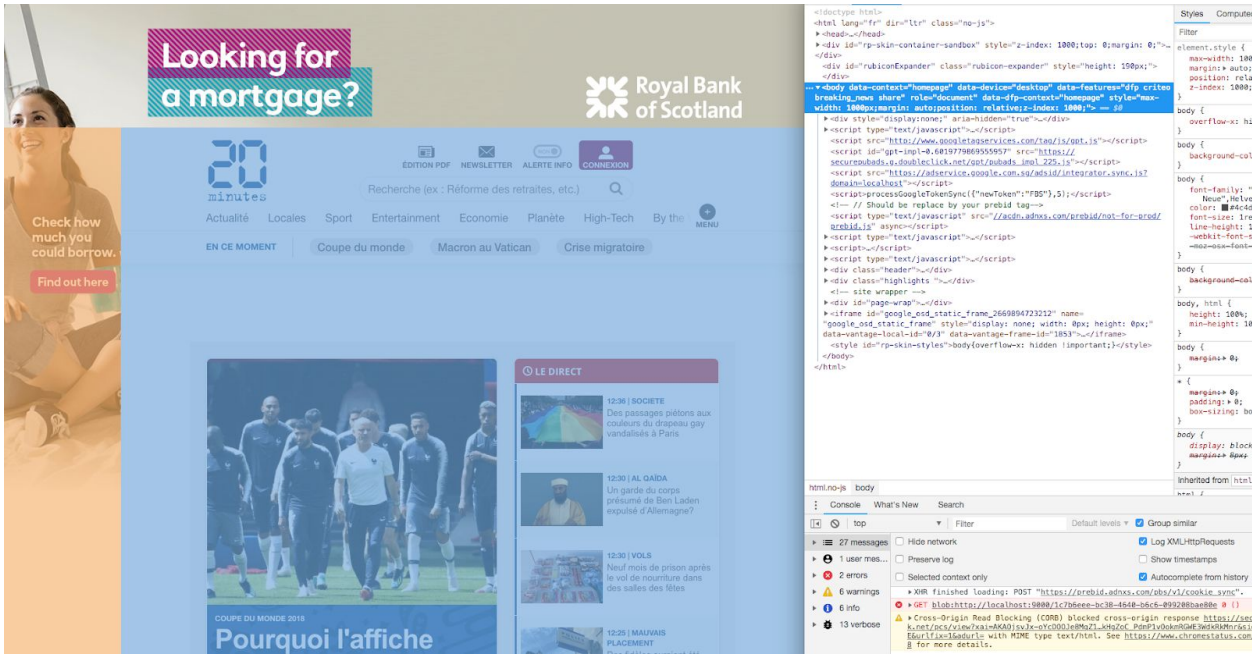
Resize: The skin spec is based around the *elementmarker* being around 1000px wide. If it is the case the "resize" mode will be adapted and force a max-width on the insertionMarker and resize the skin to fit.

Ratio: However sometimes the site won't fit the "resize" mode as the element will be squeeze or destructurate. In this case ratio allow to rescale the site without changing the elements and make the creative fully appearing.

Ratio + maxRatio: Ratio is a good solution but for some screen or site the scale will be too important and the site will appear small. And of course some publishers won't allow this. You can then define a max ratio that is the ratio limit (from 0 to 1). The rendering will be a mix between creative resize and site rescale in the limit of maxRatio (you can even use 1 if publisher doesn't allow rescalling).

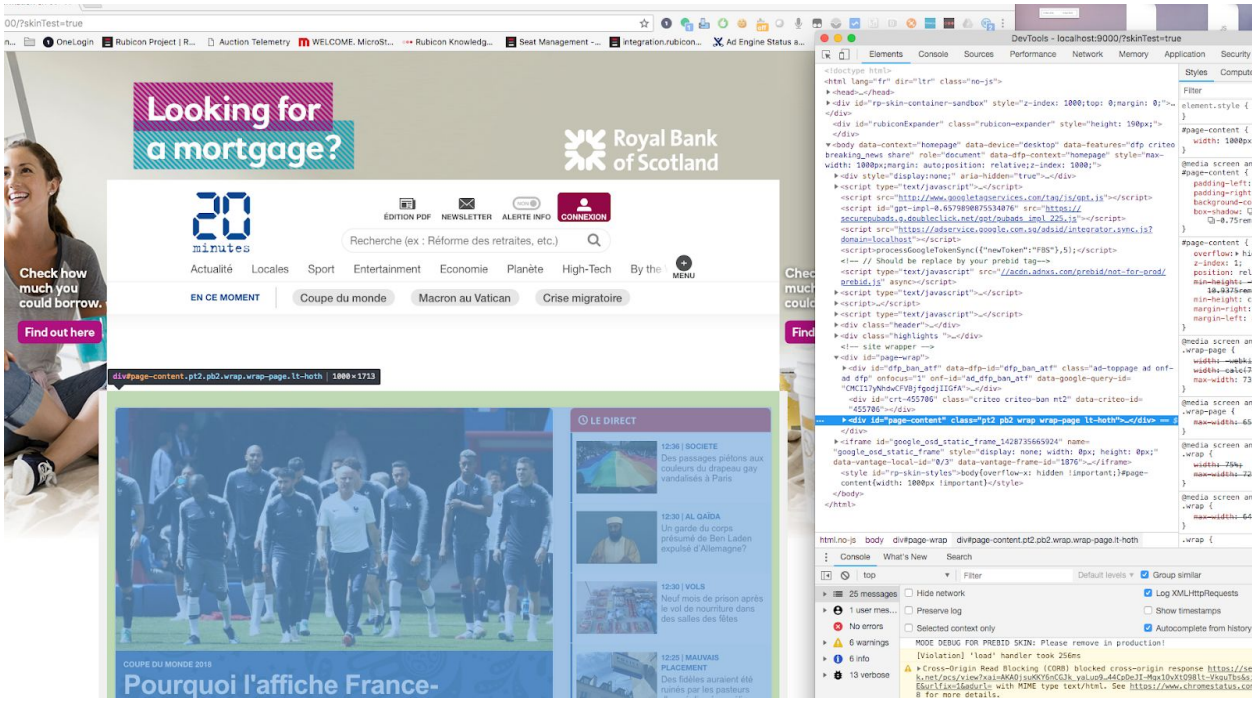
B- Adjusting to fit

Some site has particulars elements that would sometimes require a specific CSS edition.
For instance 20Minutes is great example, if using resize on <body> you could observe the following rendering:



The body and headers elements are wider than the contentWidthMarker (#page-content) defined with a 75% width of headers element. So resize the body will squeeze the content Element.

In this case we need to resize the content area using the following parameters:
custom_style_page: "#page-content{width: 1000px !important}",



Do notice that 3 insertion type are available and made so at least one could fit publishers requirements and avoid “too much” CSS styling.

3- Adapt the skin render parameters:

Use the fields description tab above to use and edit the correct fields.
There are 4 require parameters:

`insertionMarker`: You can use “body” directly if you want to wrap the whole page or the dom element where the div that inserts skin and pushes site content down is created. For instance if you want to keep the menu on the top.

`contentWidthMarker`: Used by the renderer as a basis for the width of the website content area. Content area is the exciting part of the website and not necessarily the wrap, you could use and test different value.

`insertionType`: A string that represents how the skin will be integrate, possible value are:
-ratio: will rescale the page to make sure the whole creative is shown and the page structure will stay the same. If the page appear too small you can use the maxRatio
-ratio (+ maxRatio field): will change the creative size or page scale to create the best render without rescalling over the maxRatio limit.
-resize: will resize the page to show the creative bands area without rescalling. Depending on the screen size, creative could appear to large and some sites may need additional CSS edits using the “custom_style_page” fields.

`fullBids`: don’t edit, it is the bids object.

I would recommend trying the different method and select the best one for your site. Please use a conditionnal adUnit if you want to exclude certain screen size or devices.

4- DFP setup:

Create the DFP prebid line items for the Skin adUnits and sizes following the prebid documentation.

D- Testing

Prebid doesn’t provide any way to force a dummy bids but there is a workaround to trigger the prebid Skin.
First you need to use a standard size for the prebid Skin adUnits such as `sizes: [[300,250]]`with Rubicon Prebid Test Ids:
`accountId: "14062", siteId: "70608", zoneId: "335918"`

1 - Prepare DFP:

Create a sponsorship line item in DFP targeting the skin adUnit and the keyword `hb_adid_rubicon=testSkin`.
Followed prebid documentation to add a new standard Prebid creative in DFP for the skin adUnit size and rubicon bidder (`w.pbjs.renderAd(document, '%%PATTERN:hb_adid_rubicon%%');`).

2- Create a test page:

Use a dummy test page of your site or directly your QA environment. Add the new adUnit with Rubicon test Ids and target it on a DFP slot, do notice that this adSlot could be a size different from the skin adUnit size and will be resize to 1x1 if the prebid creative is selected.

3- Adapt the skin render parameters:

See integration part. Make sure your test page is the same than the production website.
You can use “debug” field to show a test creative or use directly the url parameters (see below).

4- Force the bid response adID:

Then enter the following code in the javascript console to modify the Rubicon adUnit bid and force the adId. Notice that the request (here 300x250 for the zoneId “335918”) must return a valid bid response so we could override it. The bid response “adId” property will then be changed to “testSkin” and the rest of the response dropped.

```
<script>
javascript console> pbjs.setConfig({
  debugging: {
    enabled: true,
    bids: [{
       bidder: 'rubicon',
       adUnitCode: '/%%networkId%%/%%adUnit&&',
       adId: "testSkin"
    }]
  }
});
</script>
```

Don’t forget to deactivate debugging when you are done (or clear your local storage / use incognito mode when testing):

```
<script>
// Disabling debugging
javascript console> pbjs.setConfig({
  debugging: {
    enabled: false
  }
```

```
});
</script>
```

5- Debug mode:

Finally you need to activate the debug mode using the “debug” parameters or you can add some url params and force something to show. The below assumes that the publisher site hasn’t already appended any search params to url. If they have, you will be adding the array of param key values so start off your first param with the concat operator &

http://www.website.co.au?skinTest=true

to use your own (hosted) skin. You can browse the internet for them if you like.

http://www.website.co.au?skinTest=true&customCreative=http://path.to.your/creative.jpg

Ask your solutions engineer for access, information and installation support.