Layout Standardisation Guide

This guide serves as a point of reference for considered and consistent visual implementation of our layouts.

Design Approach

We are approaching the layout in the following manner:

- **Responsive** - Make our software work on an array of screen sizes and aspect ratios, beautifully.
- **Fluid** - Using relative units to accommodate layout requirements with minimal layout breakpoints.
- **Mobile portrait first** - We address the space constraints of a portrait mobile experience as a first concern, with progressive enhancement for landscape and then desktop. This contributes to a leaner DOM which is less demanding on the limited mobile device processing capabilities. 1. Mobile Portrait > 2. Mobile Landscape > 3. Desktop

Keep these points in mind when using the above approach:

- Use relative units - **%, vh, vw** - **NB!** Please don't use vmin and vmax as IE and Edge do not support the vmax property
  - Once a parent's size is set in vh and vw, their children can use % as much as possible.
  - Px values can be used when the parent is scaled using transform properties.

- Try to avoid using absolute or fixed units like **px**, **em** and **rem**. These will have their place in certain scenarios, but discuss these between your peers to see if it can't first be solved with relative units.

Positioning

Below are explanations and examples of what roles each positioning property plays in our design approach:

- **Static** - The default positioning property of a DOM element, does not listen to top, bottom, left or right properties. Don't use this property for single parent layout, and since positions aren't inherited, this property should also not have to be set.
  Example:
  None at this point as it is not in our strategy.

- **Absolute** - Places an object in an exact position relative to the nearest parent that has a set position, if no parent has a position set, it looks at the <body>. Absolute positioned items will scroll with the screen.
  Example: Paytables or any content that should be scrollable.

```
#Hamburger, #Home {
 position: absolute;
 width: 9%;
 top: 5%;
 margin: auto;
 z-index: 1;
}
```

- **Fixed** - Allows for exact placement, relative to the viewport. Fixed does not care what is around it, it goes where you tell it. Because it looks at the viewport, it will not scroll with a page.
  Example: Things attached to screen sides and that should stay in place, like the Infobar, full screen elements and modal dialogs.

```
#InfoBar {
 position: fixed;
 height: 3%;
 left: 0;
 right: 0;
 bottom: 0;
 background-color: rgba(0, 0, 0, 0.6);
}
```

  This also works where things are centered to the screen, regardless of browser bars etc

```
#Reels {
 position: fixed;
 margin: auto;
 top: 0;
 bottom: 0;
 left: 0;
 right: 0;
 width: 80%;
}
```

- **Relative** - Places an object relative to where it would have been had no positioning been set, ie. Places it relative to where it would have been placed in the DOM structure.
  Example: Settings list, columns where contents are flowed, or dynamically added/removed.

```
#BetTitle {
color: white;
 position: relative;
 text-align: center;
 top: 12%;
}
```

Size & Positioning

Size & positioning units

Size and positioning units are basically split in two groups namely absolute/fixed and relative. They have no relation to the Position properties of the same name.

Absolute/Fixed - Units do not change its base size based on changes in the DOM or viewport.
Examples of these that we use are:

em – 1em = 100% of parent's font size
px – 1px = 1 device pixel (On high res displays, this could actually equal to more than just one physical pixel)

Relative – Units change with outside influence, like a window resize or browser bars appearing.
Examples of these that we use are:

rem – 1rem = 100% of the root (html) font size (we don't reccoment thisto be used in a fluid layout as it relies on breakpoints to be reactive)
vh – 1vh = 1% of device viewport height
vw – 1vw = 1% of device viewport width
vmin – 1vmin = 1% of the smallest value between vh and vw **NB!** Please don't use this as IE and Edge do not support the vmax property
vmax – 1vmax = 1% of the biggest value between vh and vw **NB!** Please don't use this as IE and Edge do not support the vmax property
% - 1% = 1% of parent's size

Best practice

• Where possible, don't set a width/height, set the edge position, the width/height will follow:

Do.

```
#InfoBar {
 position: fixed;
 height: 3%;
 left: 0;
 right: 0;
 bottom: 0;
 background-color: rgba(0, 0, 0, 0.6);
}
```

Don't!

```
#InfoBar {
 position: fixed;
```

```
 height: 3%;
 left: 0;
 width: 100%;
 bottom: 0;
 background-color: rgba(0, 0, 0, 0.6);
}
```

• As a % on width would be a % of the parent width, and % on height would be a % of the parent height, a {width: 10%; height: 10%} maybe not be square.
If we need something to stay perfectly square (or perfectly round), we need to use the same value for both. Since our game is now reliant on the viewport, it’s a good unit to use as long as the rest of the game is implemented in this way.

Do.

```
#SpinButton {
 position: fixed;
 width: 18vh;
 height: 18vh;
 background-color: rgba(0, 0, 0, 0.51);
 border-radius: 50%;
 margin: auto;
}
```

Don't!

```
#SpinButton {
 position: fixed;
 width: 18%;
 height: 18%;
 background-color: rgba(0, 0, 0, 0.51);
 border-radius: 50%;
 margin: auto;
}
```

• There are a few cases where we need to use use !important to overcome styles that are coming from V, but this should not be in option for most work done. Discuss that use of it with team mates first and look at better options.
I we run into a case where it seem's only !important will work, look at the implementation and find the root cause.

Do.

```
#navigationHome {
 top: 10%;
}

.navigation {
 background-color: hotpink;
}
```

Don't!

There are very few cases where !important can be used but that should be a very low rank resort.

```
#navigationHome {
 top: 10%;
}

.navigation {
 top: 7% !important;
 background-color: hotpink;
}
```

Fonts

Size units

When working with text sizes, use the following guidelines to impement a with consistent, expected behaviour.

- **Font Type** - We have , through research concluded that these font families, in this order should be used for a more cosistent font experience across browsers:
  Arial, Helvetica, sans-serif;

- **Font Size** - Use the same unit for the font size, as is used by it's parent container to ensure consistent scaling behaviour.
  If the parent's height is in vh, the text size should be in vh, and inverse for vw.

- **Line Height** - Use the same unit for the line height as is used for the font size, to insure correct scaling.

- **Max Properties on Parent** - The height and width of a parent can be restricted using max-width and max-height properties. Since there is no equevalent behaviour for font sizes, we need to use media queries to fix the font size