Understanding Responsive and Adaptive

Silk UI provides a set of adaptive and responsive primitives that allow developers to easily create for multiple form factors, in contrast with what happens with other development technologies, particularly for mobile.

The adaptive patterns in Silk UI Mobile allow developers to adapt layouts and content for phone, tablet, and different orientations.

The responsive patterns in Silk UI Web allow developers to build fully responsive web apps.

Silk UI for Mobile, Web Responsive or Multichannel Apps

There are different templates which offer a solution for every specific need, whether it is for different mobile devices, fully responsive or multichannel apps — learn more about Silk UI templates.

Silk UI Mobile comes with three templates. Both the Phone and Tablet templates target their specific devices, but they also offer a good fallback to other form factors.

If you need to create different versions of the same screen for each device in the same application, use the Universal template. After that, follow the guidelines on the home screen to be able to navigate to the correct screen.

Adaptive in Silk UI Mobile

Depending on screen size and resolution, Silk UI Mobile captures device information (phone or tablet) to adapt content to the screen.

Layouts for phone and tablet have different ways of handling screen size when used on other devices:

- The Phone Theme will create guard rails on the side when used on tablets. This prevents the original content from stretching and breaking the UI.
• The Tablet Theme will have a UI similar to the Phone Theme when used in phones and by making use of adaptive patterns — like columns — it's very easy to adjust content.

*Phone template's adaptive behavior*

*Tablet template, also using adaptive patterns*

**Adaptive Patterns**

Every app is different. The way your app adapts to a specific device is, therefore, unique. To address this, Silk UI Mobile offers a set of [adaptive patterns](https://success.outsystems.com/Documentation/SILK_UI_Framework/05_Understanding_Responsive_and_Adaptive). These patterns will give you full control of the way your application will look on different screen sizes, allowing for different settings in landscape or portrait for phones and tablets.
Adaptive Actions

Silk UI Mobile patterns are able to adapt the UI to the device, using only CSS. But you need a little extra to run different logic according to the device. That's why Silk UI Mobile has a set of client-side actions that allow developers to decide which screen areas to hide, how images are placed or even which screen to navigate to, according to the detected device.
Responsive in Silk UI Web

In Silk UI Web everything is responsive — themes, layouts, and patterns. But it is a different kind of responsive. Instead of using a client-side approach, where all content is always sent to the browser and it then decides, through CSS, what to display, Silk UI Web uses a server-side approach.

This server-side approach is sometimes called **RESS** (Responsive through Server-Side components). Learn more about the RESS approach by reading [LukeW](http://www.smashingmagazine.com)’s takes the subject.

When using a server-side approach, content is not adapted as the browser is resized. However, that is not the expected behavior for responsive. The main use case is when a specific device accesses the application and requires an adapted version of the page to improve fit and performance.

Silk UI's Implementation of Server-Side Responsive

The server-side responsive approach has several advantages over the client-side approach:
• Only the required content is sent to the browser because the server is aware of the device in use.

• There are no CSS media-queries, which makes responsive easier to understand and handle.

• The device context is present on the server, which means different strategies can be used to handle devices.

Silk UI does not rely on CSS to handle the page rendering for devices. This means no media queries will be found in our CSS, which is a good practice since media queries are difficult to read, hard to maintain and easy to break.

```css
@media screen
  and (max-width: 700px)
  and (min-width: 340px)
  and (device-pixel-ratio: 2) {
    ...
  }
```

### How it Works

There’s a snippet of JavaScript that runs on first page load and that finds out, through its user agent or viewport width, what the device is: desktop, tablet or smartphone.

A cookie is set for future visits and this device context is sent to the server and placed in the session. From this point on, the applications have full device context for the runtime so that, whenever you return, the content is already adapted.

The page rendering is directed to the device by the injection of a specific class on the page.

You can see the detected device by turning on **Development Mode**.

The responsive behavior is achieved by using a set of patterns and blocks.

There’s a set of patterns which simplify responsive behavior development:

• Columns

• Responsive Patterns
Columns Responsive Behavior

A set of structure patterns is provided with the framework. These patterns allow dividing content in columns which follow the underlying grid. They are blocks with specific responsive parameters, which allow the developer to configure how they should react on each device.

Possible values for tablet and smartphone are:

- Break All: All columns break, which means each column will occupy the total width of the layout.
- Break First: Only the first column breaks, which means the first column will occupy the total width and the other ones will remain side by side.
- Break Middle.
- Break Last.
- Don’t Break.

With Silk UI, you can still use the platform grid to organize content, but it has no responsive behavior. Use the columns pattern instead, as that way responsive can be configured and more easily understood.

Responsive Patterns

A set of responsive patterns is provided and it allows you to define exactly what is sent to the browser, depending on configuration.

- **DisplayOnDevice**: Content is displayed only on a specific device.
- **LoadOnVisible**: By default, content is not sent to the browser, but it is loaded as user scrolls.
- **ResponsiveImages**: Depending on the device, it sends only the specific image.
- **ToggleOnDevice**: Only loads a trigger, which loads content through AJAX upon user action.
Device Simulation

In non-productive environments, the device simulation handler is triggered automatically on window resize. This is meant to help developers test the application's responsive behavior.

When device simulation is turned on, device guidelines will appear on the screen and a page reload is triggered for each resizing, to fit contents to that device.

Note that the responsive setting is shared between applications through a domain cookie.

Specific device previews can be triggered from the handler to view the application in a device frame.
You are able to activate device simulation in production environments by setting the `ForceSimulationDevice` site property to `true` in the WebPatterns module of the Silk UI Web application.

### CSS Device Context

When the device is detected or if the context is present at the server, a set of classes is added to the page layout. Specifically, the `<div>` with class “Page”.

```html
<script type="text/javascript">...</script>
<script type="text/javascript">...</script>
<div class="Page chrome en desktop hd landscape">
  <div id="wt51_wtMenu" class="Menu Header_menu OSInline">...</div>
  <div class="Header">...</div>
</div>
```

The information it adds is the following:

- **Device**: desktop, tablet, phone.
- **Dimension**: small, big, hd.
- **Orientation**: landscape, portrait
- **Browser**: chrome, firefox, IE8, IE9, IE10, IE11, Safari, etc.
- **Operation System**: windows, ios, osx, android, ubuntu, linux, unknownos

This makes CSS coding easier:

```css
.Button {
  padding: 10px;
}
.tablet .Button {
  padding: 15px;
}
.phone .Button {
  padding: 12px;
}
.desktop.IE8 .Button {
  padding: 9px;
}
```