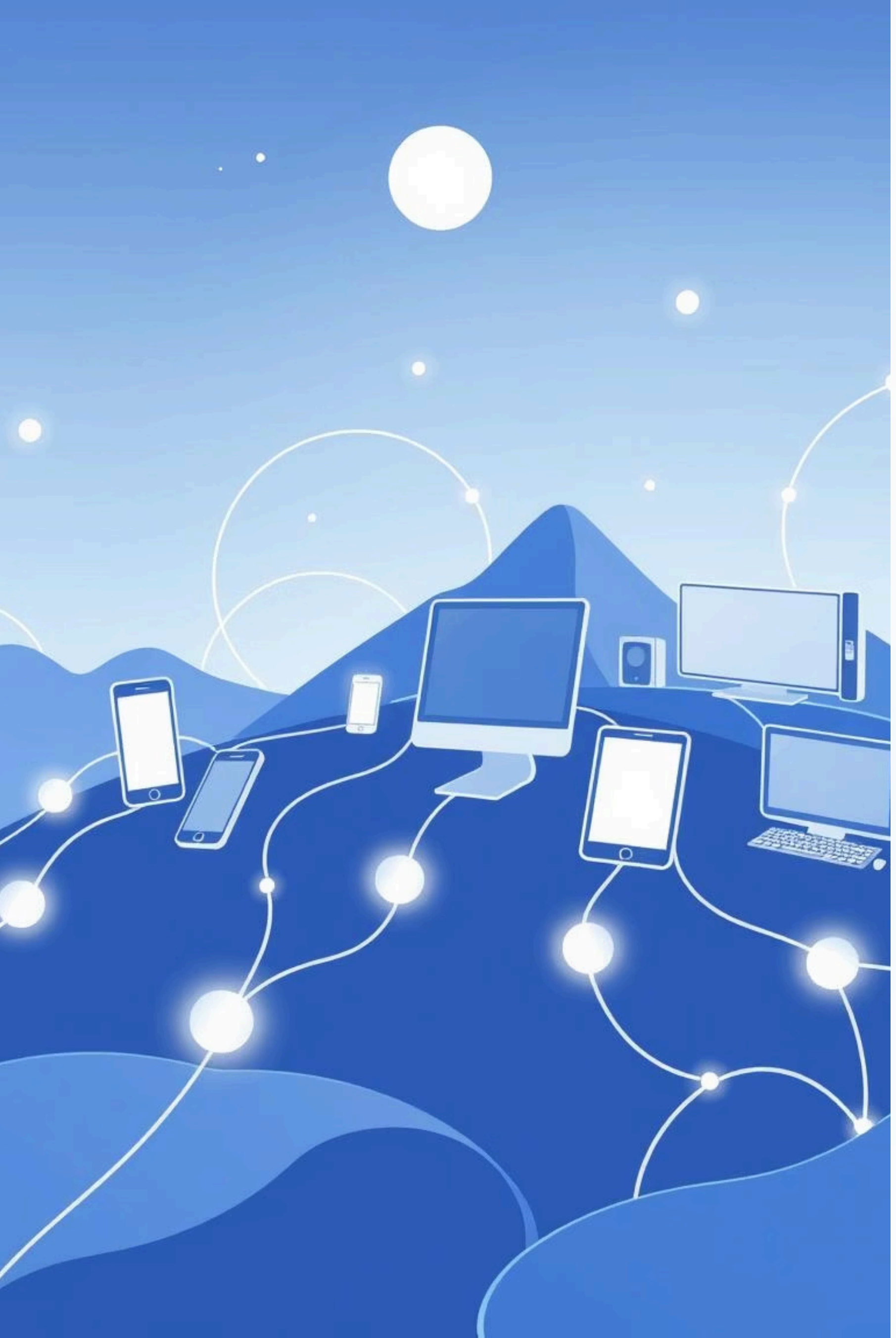
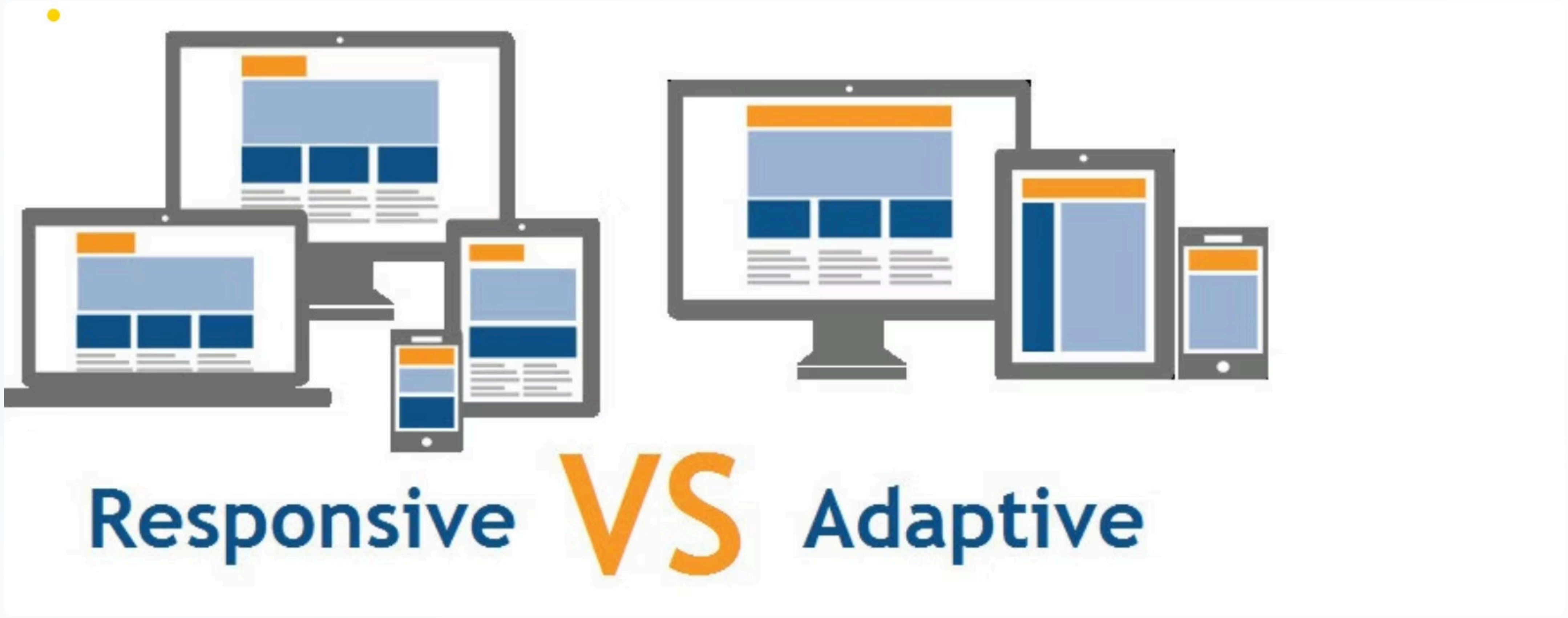


Responsive Design

Responsive vs Adaptive



Why Responsive Design Matters

Device Diversity

Users access apps on a vast array of devices, from small smartwatches to large desktop monitors. Responsive design ensures a consistent, optimal experience across all of them.

Enhanced User Experience

A well-designed responsive app provides intuitive navigation and readable content, leading to higher user satisfaction and engagement. No more awkward scaling or hidden elements.

Future-Proofing

As new devices and form factors emerge, a responsive approach ensures your app remains relevant and functional without major overhauls, saving development time and resources.

Flutter's Approach to Responsiveness

Key principles include:

- **Flexible Layouts:** Widgets like Expanded, Flexible.
- MediaQuery .
- flutter_screenutil package.

Approach 1: Using MediaQuery for Responsiveness

`MediaQuery.of(context)` provides direct access to the device's screen size, orientation, pixel density, and more. It's a built-in Flutter solution, offering granular control.

How it works:

- Retrieve screen dimensions (e.g., `MediaQuery.of(context).size`).
- Use these values to calculate widget sizes, paddings, and font sizes dynamically.

MediaQuery: Code Example

Example: Not responsiveText Size

```
class NotResponsive extends StatelessWidget {
  const NotResponsive ({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Broken Layout")),
      body: Center(
        child: Container(
          width: 400,
          height: 200,
          color: Colors.redAccent,
          child: const Center(
            child: Text(
              "Too wide for small screens!",
              style: TextStyle(fontSize: 20, color: Colors.white),
            ),
          ),
        ),
      ),
    );
  }
}
```

Responsive

Example: Responsive Container and Text

```
import 'package:flutter/material.dart';

class Responsive extends StatelessWidget {
  const Responsive({super.key});

  @override
  Widget build(BuildContext context) {
    double width = MediaQuery.of(context).size.width;

    return Scaffold(
      appBar: AppBar(title: const Text("Broken Layout")),
      body: Center(
        child: Container(
          width: width * 0.9, // Adjust width to be 80% of the screen width
          height: 200,
          color: Colors.redAccent,
          child: const Center(
            child: Text(
              "Too wide for small screens!",
              style: TextStyle(fontSize: 20, color: Colors.white),
            ),
          ),
        ),
      ),
    );
  }
}
```

Task:

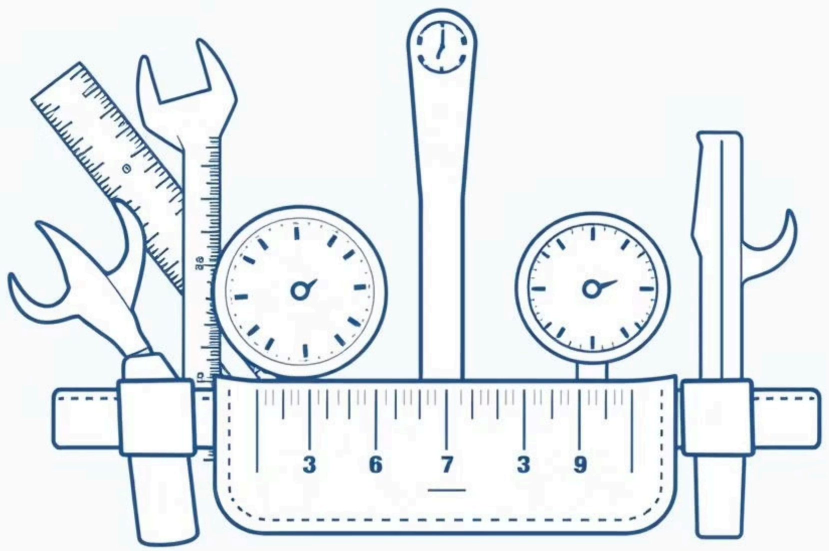
create a `Container` with

1) red color

2) height=400

3) width = half of screen width always

Approach 2: Using flutter_screenutil Package



flutter_screenutil is a powerful package that simplifies responsive design by providing a way to scale UI elements based on a design draft size. You define a base screen size, and the package handles the scaling automatically.

How it works:

- Initialize with your design's screen width/height (e.g., 360x690).
- Use extensions like .w (width), .h (height), .sp (scaled pixel for text) directly on numbers.
- The package calculates the actual size relative to the device's screen.

Assignment: Responsive BMI Calculator