Topic

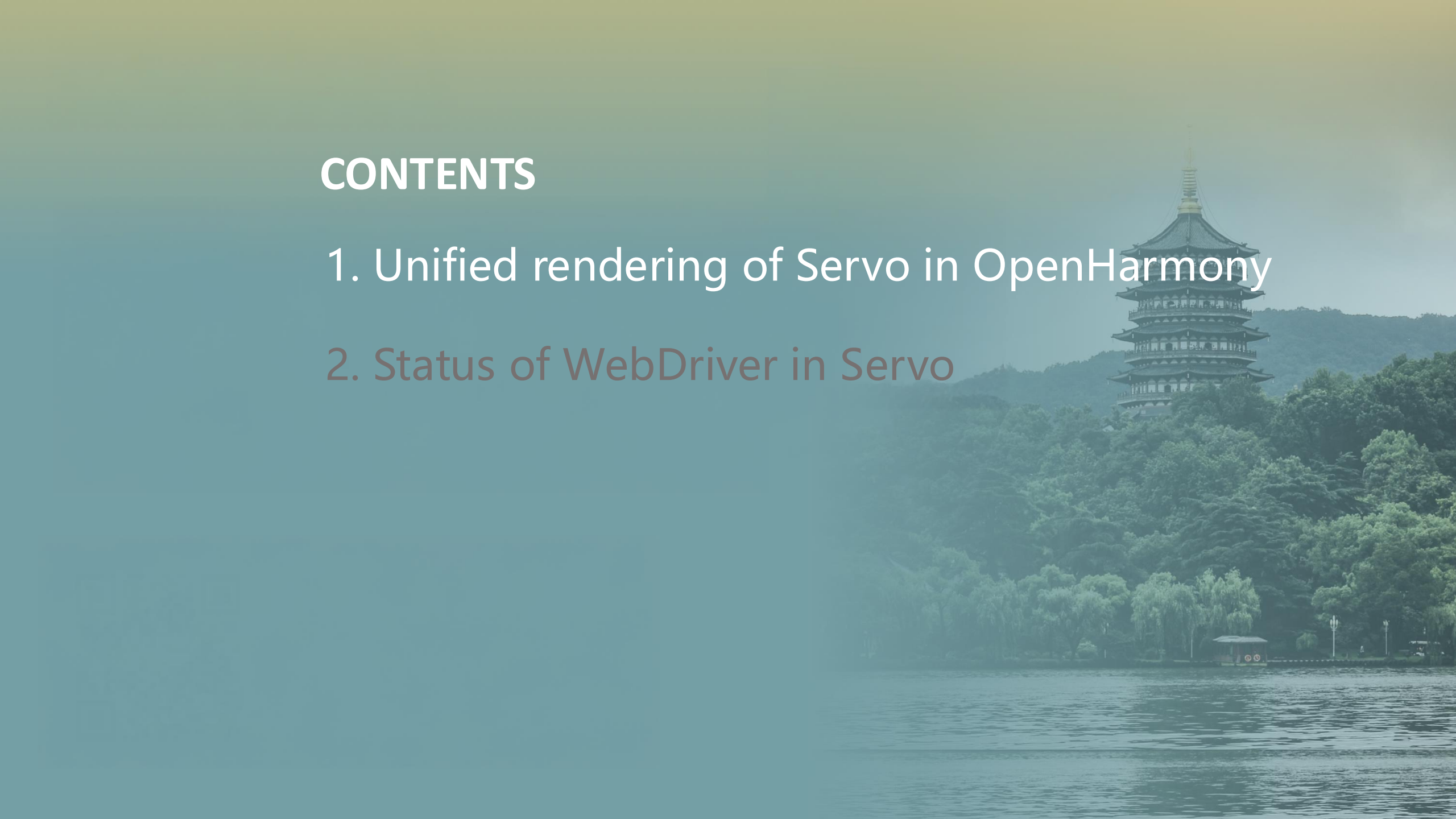
Driving Innovation with Servo and OpenHarmony: Unified Rendering and WebDriver

Authors: Jingshi Shangguan & Zhizhen Ye

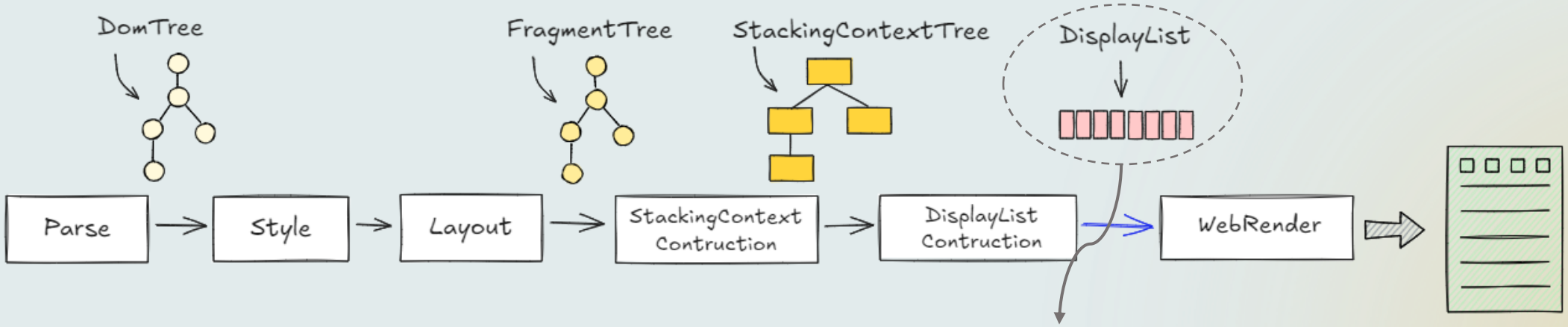
CONTENTS

1. Unified rendering of Servo in OpenHarmony

2. Status of WebDriver in Servo



Current Rendering solution in Servo: Powered by WebRender



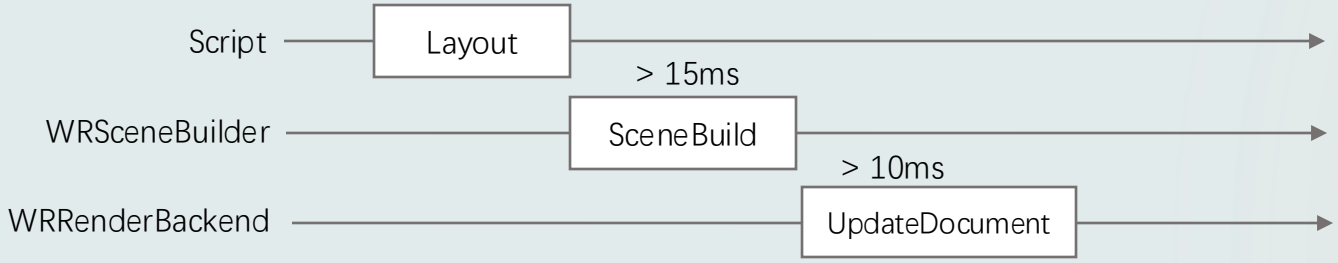

```
pub enum DisplayItem {
  // These are the "real content" display items
  Rectangle(RectangleDisplayItem),
  ClearRectangle(ClearRectangleDisplayItem),
  HitTest(HitTestDisplayItem),
  Text(TextDisplayItem),
  Line(LineDisplayItem),
  Border(BorderDisplayItem),
  BoxShadow(BoxShadowDisplayItem),
  RectClip(RectClipDisplayItem),
  ...
}
```

```
pub enum SpatialTreeItem {
  ScrollFrame(ScrollFrameDescriptor),
  ReferenceFrame(ReferenceFrameDescriptor),
  StickyFrame(StickyFrameDescriptor),
  Invalid,
}
```

After the DisplayList is created, it is sent to the Compositor, which then forwards it to WebRender.

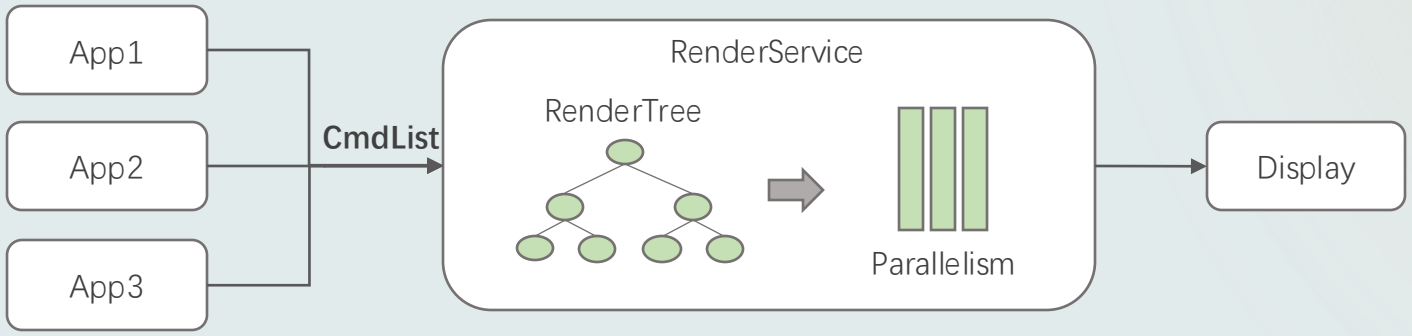
Vertical Integration of Servo with OpenHarmony

- WebRender demonstrates suboptimal performance in certain scenarios. Therefore, we are exploring vertical integration with OpenHarmony to optimize rendering performance.



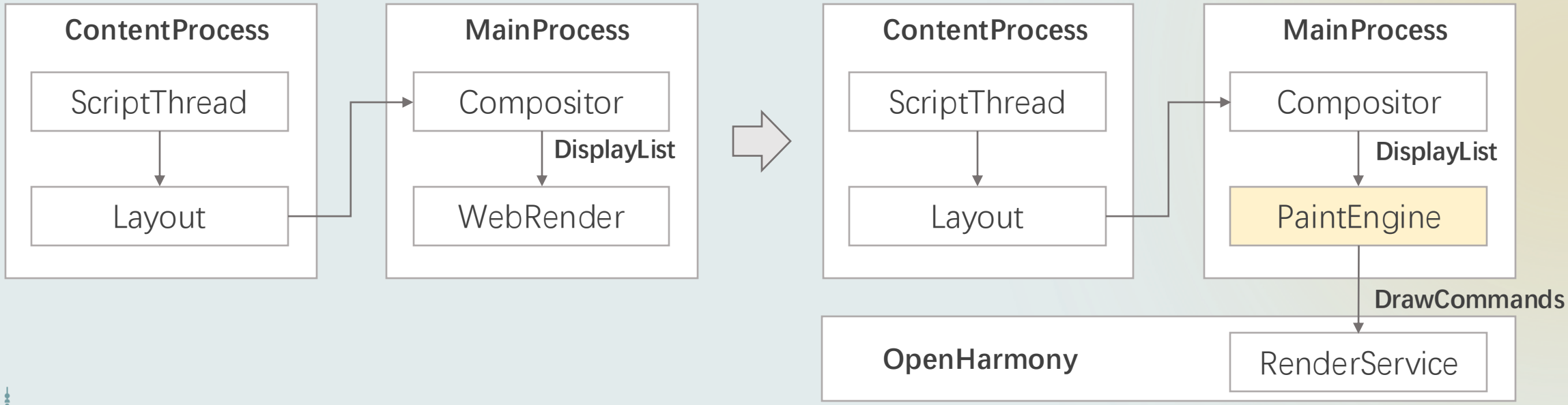
- The data is based on tests conducted on the Mossel H5 Shopping Homepage.
- Mossel is a mobile-friendly shopping page designed for premium retail.

- **RenderService** is a core component of the **OpenHarmony graphics subsystem**, leveraging occlusion culling and minimal region updating to reduce GPU workload and improve frame efficiency.

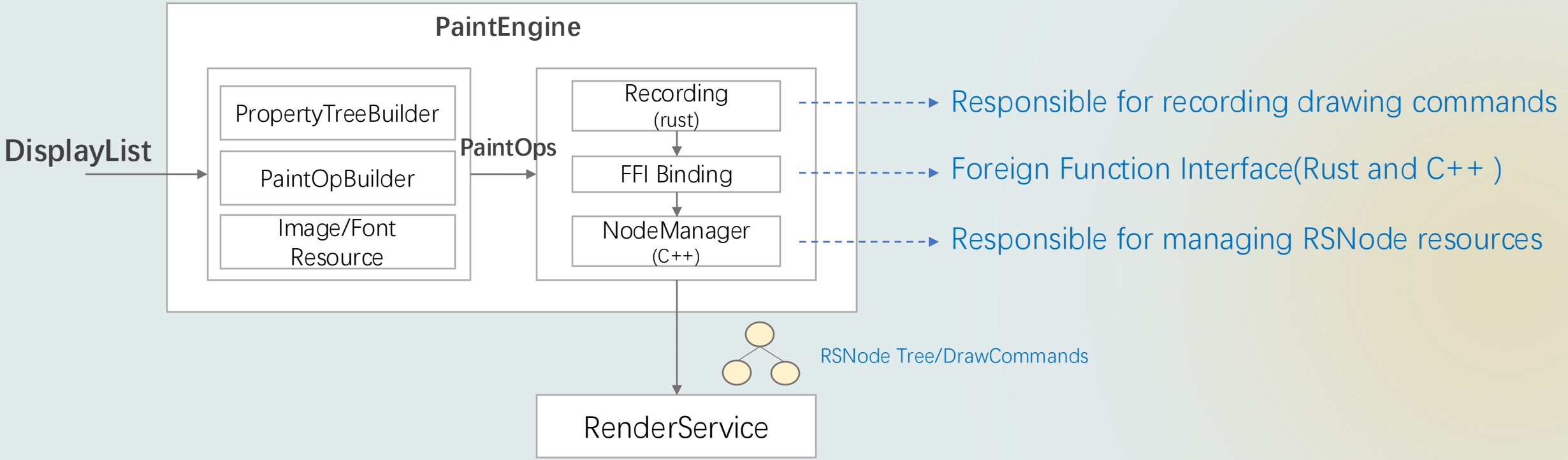


Enabe Unified Rendering

- Reuse the DisplayList defined by the WebRender API out of compatibility considerations.
- The design leverages WebRender's DisplayList format and **adds a crate of PaintEngine** to support command recording and bridge rendering operations with RenderService.



Baisc design of PaintEngine



FFI-Bindings for OpenHarmony: https://github.com/openharmony-rs/ohos-sys

Example：Record draw commands

```
RSCanvasNode::SharedPtr canvasNode = RSCanvasNode::Create();
DrawFunc func =
        [&](std::shared_ptr<Drawing::Canvas>) { // Call Rust Function };
canvasNode->DrawOnNode(RSModifierType::INVALID, func);
```

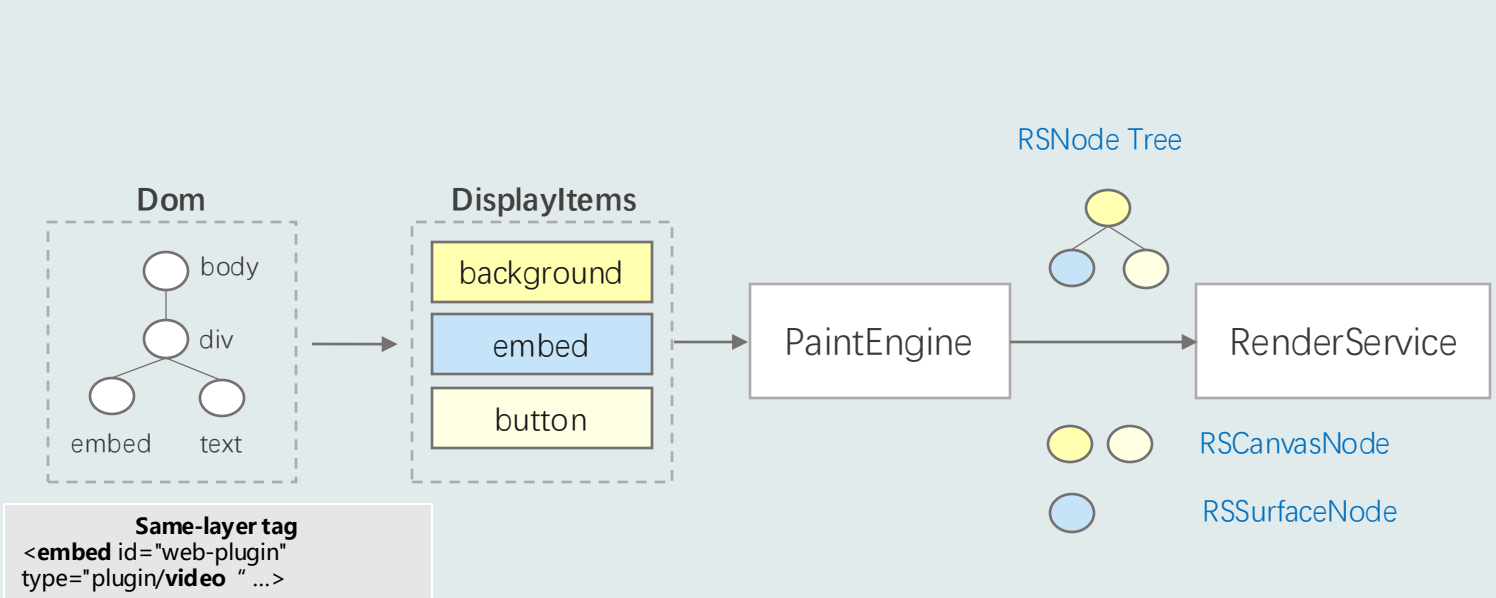
```
impl PaintOpRectangle {
    pub fn draw(&self, canvas: *mut OH_Drawing_Canvas, draw_rect: &LayoutRect) {
        unsafe {
            OH_Drawing_CanvasSave(canvas);
            let drawing_rect: *mut OH_Drawing_Rect = OH_Drawing_RectCreate(
                draw_rect.min.x,
                draw_rect.min.y,
                draw_rect.max.x,
                draw_rect.max.y,
            );
            let brush: *mut OH_Drawing_Brush = OH_Drawing_BrushCreate();
            OH_Drawing_BrushSetColor(brush, to_hex(&self.color));
            OH_Drawing_CanvasAttachBrush(canvas, brush);
            OH_Drawing_CanvasDrawRect(canvas, drawing_rect);
            OH_Drawing_CanvasDetachBrush(canvas);
            OH_Drawing_BrushDestroy(brush);
            OH_Drawing_CanvasRestore(canvas);
        }
    }
}
```

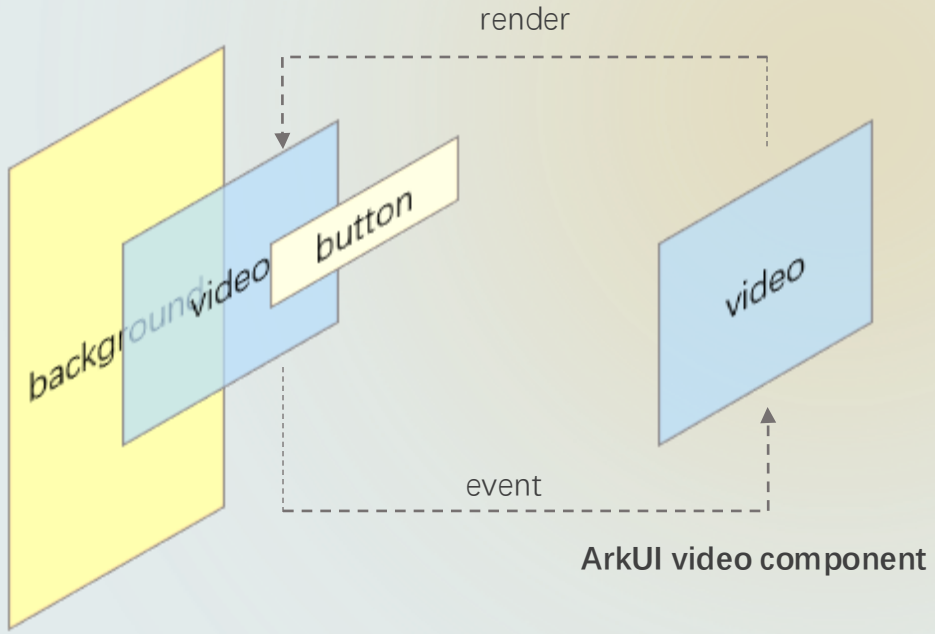
C APIs Drawing: https://docs.openharmony.cn/pages/v5.0/en/application-dev/reference/apis-arkgraphics2d/_drawing.md

OpenHarmony/graphic_2d: https://gitee.com/openharmony/graphic_graphic_2d

Support Same-Layer Rendering

- In the system, applications can use the Web component to load web pages. If the capability or performance of non-native UI components (same-layer components) is inferior, you can use the ArkUI component to render instead.
- On the web page, you can render the UI components (same-layer tags) such as **<embed> and <object>** at the same layer based on certain rules.





Example: Using Same-Layer Rendering

GOSIM

html

```
<!--HAP's src/main/resources/rawfile/text.html-->
<!DOCTYPE html>
<html>
<head>
    <title>Same-Layer Rendering Test HTML</title>
    <meta name="viewport">
</head>

<body style="background:white">

<embed id = "input1" type="native/view" style="width: 100%; height: 100px;
margin: 30px; margin-top: 600px"/>

<embed id = "input2" type="native/view2" style="width: 100%; height: 100px;
margin: 30px; margin-top: 50px"/>

<embed id = "input3" type="native/view3" style="width: 100%; height: 100px;
margin: 30px; margin-top: 50px"/>

</body>
</html>
```

ArkTs

```
// Load the local text.html page.
 Web({src: $rawfile("text.html"), controller: this.browserTabController})
   // Enable same-layer rendering.
   .enableNativeEmbedMode(true)
     // Register the same-layer tag of "object" and type of "test."
   .registerNativeEmbedRule("object", "test")
     // Obtain the lifecycle change data of the embed tag.
   .onNativeEmbedLifecycleChange((embed) => {
     console.log("NativeEmbed surfaceId" + embed.surfaceId);
     // If embed.info.id is used as the key for mapping nodeController,
     // explicitly specify the ID on the HTML5 page.
     const componentId = embed.info?.id?.toString() as string
     if (embed.status == NativeEmbedStatus.CREATE) {
       console.log("NativeEmbed create" + JSON.stringify(embed.info));
       // Create a NodeController instance, set parameters, and rebuild.
       ...
     } else if (embed.status == NativeEmbedStatus.UPDATE) {
       ...
     } else if (embed.status == NativeEmbedStatus.DESTROY) {
       ...
     } else {
       console.log("NativeEmbed status" + embed.status);
     }
   })
```

Using Same-Layer Rendering:
https://docs.openharmony.cn/pages/v5.1/en/application-dev/web/web-same-layer.md

Live Demo：Rendered by RenderService

Compared to WebRender, the single-frame rendering load(CPU instructions) is reduced by 15%.

Test Scenario: A 30-second scroll test was conducted on the Mossel H5 Shopping Homepage using the Mate60Pro device to measure CPU instruction count.

Issues:

- During each reflow, StackingContext and DisplayList construction takes over 15ms.
- The complete DisplayList is then sent to the PaintEngine (or WebRender), which incurs an additional cost of over 15ms.

The data was collected under a constrained test scenario (**Mossel H5 Shopping Homepage**).
Comparative tests across other scenarios are yet to be conducted.

Features planned for implementation

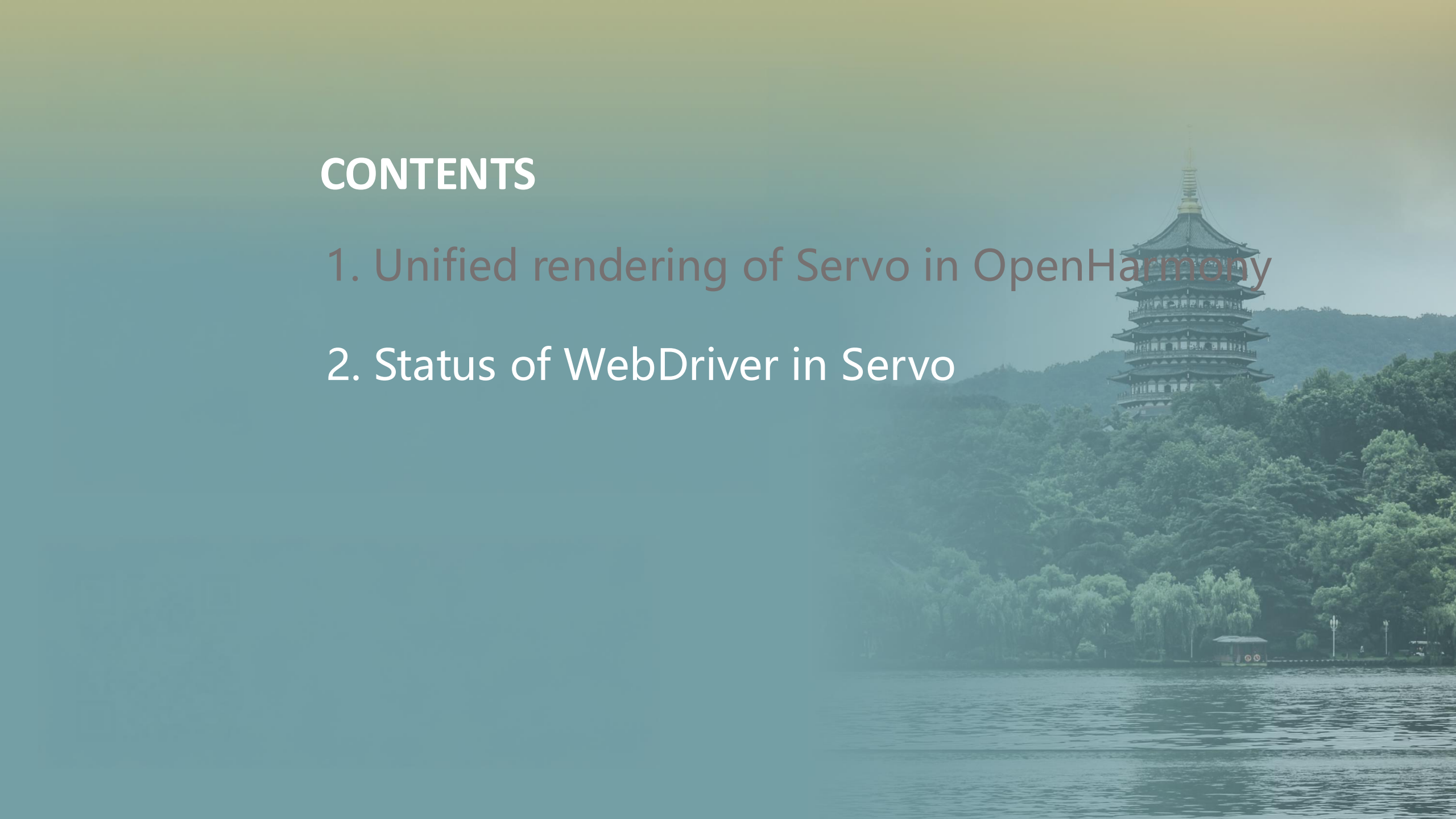
- Support rendering of additional DisplayItem types.
- Enable incremental construction of StackingContexts and DisplayList.
- Apply layering to the DisplayList for optimized rendering.

The unified rendering modifications have not yet been submitted to the community.

CONTENTS

1. Unified rendering of Servo in OpenHarmony

2. Status of WebDriver in Servo

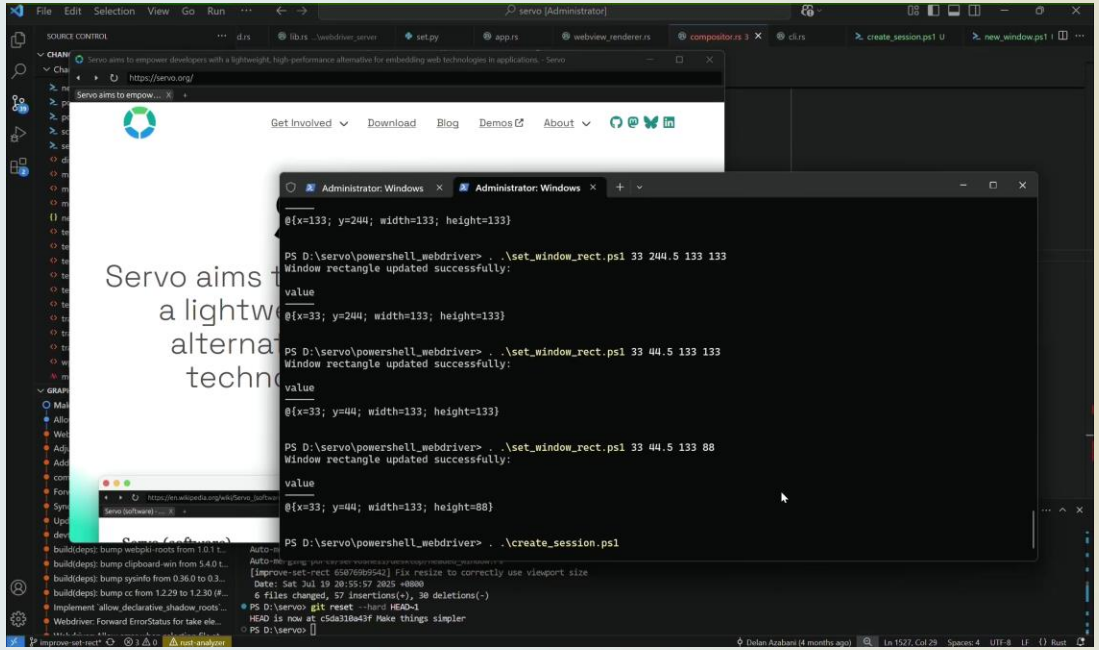


What is WebDriver?

- **WebDriver**

A platform & language neutral protocol for automating web browsers. It lets code to remotely control the browser—clicking, typing, and navigating—just like a human.

- **Quick Demo**



Why important for Servo & OpenHarmony?

1. Prerequisite for enable interactive test Coverage (testdriver) for all platforms

e.g., click, scroll, key input, back/forward, take screenshot

Why important for Servo & OpenHarmony?

GOSIM

2. Running WPT test on mobile platform

Motivation: The difference between mobile platform and Desktop is huge.

Challenge:
1. WPT server is run on Desktop
2. Hope to report result on Desktop
3. We do not run multiprocess mode for WPT in Servo yet. That is why we kill-and-reboot Servo for each test in CI. That would be too intensive for mobile phone and cause thermal issues.

How WebDriver helps:
1. Guide Servo on OHOS to reach the wpt server on desktop.
2. Run tests consecutively rather than kill-and-reboot.

Current Status of WebDriver in Servo

GOSIM

1. WebDriver Classic Conformance tests enabled in CI

3033 (pass) /3444 (total enabled), 88%

In comparison, Safari: 2359/2997; Chrome: 3266/3456; Firefox: 3420/3456

2. Migrated WebDriver support to mobile platform.

3. Able to run single WPT test in OpenHarmony and collect results in Desktop.

Future plan of WebDriver in Servo

1. Utilise WebDriver to support fast WPT runner by reusing the instance. Currently, Servo still kill-and-reboot between tests.

Example of test flex-flow-001.html shared in Zulip by Nico:

Chrome	Firefox	Safari	Servo	Ladybird	Flow	Blitz
400ms	135ms	166ms	1563ms	514ms	220ms	15ms

2. Enable testdriver tests in CI. Currently, it can be run locally with “--product servodriver”

Thank You