chrome side

content side

platform: Linux
branch: servo 0.0.1

In-process interface to Servo.
It does everything necessary to render the web, primarily orchestrating the interaction between JavaScript, CSS layout, rendering, and the client window.

servo::Browser

- start()

Maintains the pipelines and navigation context and grants permission to composite

constellation::Constellation

- recv()

mpsc::Receiver<FromScriptMsg>

ipc_channel::router::ROUTER

IpcReceiver<ConstellationMsg>

- send(IpcSender<CanvasMsg>)

IpcSender<IpcSender<CanvasMsg>>

- spawn_named()

util::thread

- recv()

- start()

CanvasPaintThread

- new()
- get_size()
- get_format()
- snapshot()
- set_transform()
- push_clip()
- pop_clip()
- fill()
- fill_rect()
- clear_rect()
- stroke()
- stroke_rect()
- stroke_line()
- create_path_builder()
- create_similar_draw_target()
- create_source_surface_from_data()
- copy_surface()
- draw_surface()
- draw_surface_with_shadow()

azure::azure_hl::DrawTarget

- alloc_image()
- delete_image()
- update_image()

webrender_traits::RenderApi

IpcReceiver<CanvasMsg>

- send(CanvasData::Pixels())

IpcSender<CanvasData>

CanvasPixelData

webrender_traits::ImageKey

IpcSharedMemory

Freely-copyable reference to a rooted global object.

global::GlobalRef

- constellation_chan()

dom::window::Window

IpcSender<ConstellationMsg>

IpcReceiver<ConstellationMsg>

- send(CreateCanvasPaintThread)

IpcSender<IpcSender<CanvasMsg>>

- recv()

- send(CanvasMsg::Canvas2d)
- clone()

IpcSender<CanvasMsg>

IpcReceiver<CanvasData>

- recv()

CanvasPixelData

webrender_traits::ImageKey

- send(CanvasMsg::FromLayout(
FromLayoutMsg::SendData(sender)))

FragmentDisplayListBuilding for Fragment

create

gfx::display_list::ImageDisplayItem

Mutable field in the DOM

Traced reference to a DOM object
This should only be used as a field in other DOM objects.

web content

- getContext('2d')
- toDataURL()

DOMRefCell<Option<CanvasContext>>

HTMLCanvasElement

- new()
- encode()

JS<CanvasRenderingContext2D>

CanvasRenderingContext2D

image::png::PNGEncoder

LayoutJS<CanvasRenderingContext2D>

- get_ipc_renderer()

LayoutJS<HTMLCanvasElement>

layout_thread::LayoutThread

traversal::RecalcStyleAndConstructFlows

DomTraversalContext<N> for RecalcStyleAndConstructFlows

- new()
- process()

construct::FlowConstructor

Object that knows how to create flows

create

- data()

LayoutNode

HTMLCanvasData

ThreadSafeLayoutNode

- canvas_data()

CanvasFragmentInfo

OpaqueNode

- opaque()

- canvas_inline_size()
- canvas_block_size()

Fragment

- new()

- new()

Fragments (struct Fragment) are the leaves of the layout tree. They cannot position themselves. In general, fragments do not have a simple correspondence with CSS fragments in the specification: