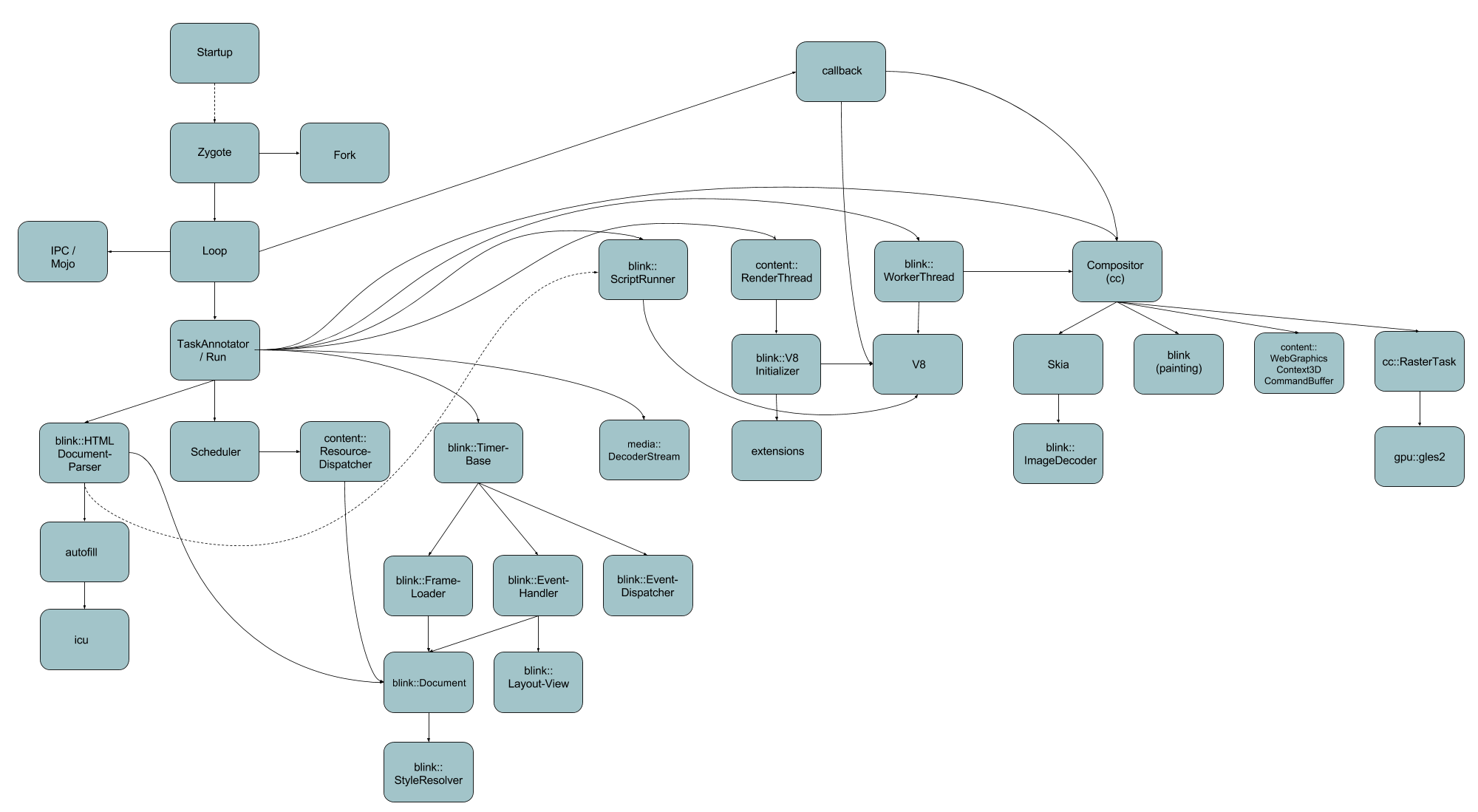
# *Zygote process - detailed architecture overview*

[](https://docs.google.com/drawings/d/153E_TQxq4NVSqzQyqqUtqkkCHrkEnbVDhqf_Ur6Tr-c/edit?usp=sharing)

* **Startup**
  + Startup box represents the parent process’s (Browser process) startup procedure
* **Zygote**
  + after Browser process startup, Zygote process is created
* **Fork**
  + Zygote process is forked in order to create Renderer process(es)
* **Loop**
  + MainMessageLoop
    - is used to process events for a particular thread
    - puts the incoming messages, tasks to a queue
    - pops a task from the queue and starts it
    - strong relationship with the IPC communication framework
    - has task reentrancy protection
      * second task cannot be started until first task finishes
* **IPC / Mojo**
  + framework which is used for inter-process communication
  + connects directly to the MainMessageLoop
  + provides communication channels through which the messages can be sent
  + message creating, sending and receiving
  + asynchronous message handling
* **TaskAnnotator**
  + all incoming tasks are going through a TaskAnnotator which annotates the task before the execution
  + implements common debug annotations for posted tasks. This includes data such as task origins, queueing durations and memory usage.
  + runs a previously queued task
* **callback**
  + this box represents system or message related callbacks
* **Scheduler**
  + package which contains multiple classes regarding task schedule
  + TaskQueueManager
    - The task queue manager provides N task queues and a selector interface for choosing which task queue to service next. Each task queue consists of two sub queues
      * Incoming task queue
      * Work queue
* **blink::HTMLDocumentParser**
  + parse the HTML document
  + build the DOM tree
* **autofill**
  + AutofillAgent deals with Autofill related communications between WebKit and the browser
  + there is one AutofillAgent per RenderFrame
  + Autofill encompasses:
    - single text field suggestions, that we usually refer to as Autocomplete,
    - password form fill, referred to as Password Autofill, and
    - entire form fill based on one field entry, referred to as Form Autofill.
* **icu**
  + ICU stands for International Components for Unicode
  + library which provides Unicode and Globalization support for software applications
  + in the current context icu is used for regular expression pattern matching to determine if an autofill matches a specific regular expression
* **content::ResourceDispatcher**
  + this class serves as a communication interface to the ResourceDispatcherHost in the browser process
  + it can be used from any child process.
  + dispatches incoming messages
* **blink::TimerBase**
  + basic Timer
* **blink::FrameLoader**
  + controls a load of a specific frame (page)
* **blink::EventHandler**
  + handle events like selection, drag and drop, gesture, mouse move, mouse press or release
  + perform hit test
* **blink::EventDispatcher**
  + dispatches simple events, scoped events or simulated clicks
* **blink::Document**
  + blink::DocumentLoader is responsible for the loading of the Document
  + after CSS style applying (**blink::StyleResolver**) and Layout calculation the paint layers and graphics layers are updated
* **media::DecoderStream**
  + wraps a DemuxerStream and a list of Decoders and provides decoded output to its client (e.g. Audio/VideoRendererImpl).
* **blink::ScriptRunner**
  + class for executing JavaScript instructions
* **content::RenderThread**
  + a thread which is used for rendering tasks in Renderer and Zygote process (only one can be present at a time)
* **blink::V8Initializer**
  + has only static methods in order to initialize v8 context
    - on the main thread
    - on a worker thread
* **extensions**
  + reusable extensions module
  + it implements the core parts of Chrome's extension system, and can be used with any host of the 'content' module.
* **blink::WorkerThread**
  + thread which can execute specific tasks
  + tasks can be posted to a worker
  + calls WorkerScriptController::initializeContextifNeeded in order to execute JavaScript via V8
* **Compositor (cc)**
  + uses multiple backing stores to cache and group chunks of the render tree
  + avoids unnecessary repainting
  + primary compositing tasks
    - determine how to group contents into backing stores (i.e. composited layers)
    - paint the contents of each composited layer
    - draw the composited layers to make a final image
  + paints the layer contents to display lists
  + handles layer updates
* **Skia**
  + Blink’s drawing library
  + the rasterization calls specific **Skia** functions in order to get the canvas drawn correctly
    - drawColor, drawPicture, drawRect, fillRect, etc.
* **blink::ImageDecoder**
  + ImageDecoder is a base for all format-specific decoders (e.g. JPEGImageDecoder). This base manages the ImageFrame cache.
* **content::WebGraphicsContext3DCommandBuffer**
  + 3D graphic related methods
  + forwards instructions to the **GpuChannelHost** which
    - encapsulates an IPC channel between the client and one GPU process
    - on the GPU process side there's a corresponding GpuChannel
    - every method can be called on any thread with a message loop, except for the IO thread
* **RasterTask**
  + task, which performs rasterization
  + tasks are represented as a task graph
    - edges: dependencies
    - node: tasks, priority is assigned to them
  + items in the display list are drawn to the surface
  + rasterization calls gpu::gles2::QueryTracker methods in order to create queries to the GPU
* **gpu::gles2**
  + QueryTracker
    - tracks queries for client side of command buffer.
  + QueryManager
    - keeps track of the queries and their state as Queries are not shared there is one QueryManager per context.
  + sends queries to content::CommandBufferProxyImpl
    - client side proxy that forwards messages synchronously to a CommandBufferStub

Sources:

<https://code.google.com/p/chromium/codesearch#chromium/src/components/scheduler/base/task_queue_manager.h&q=task_queue_manager.h&sq=package:chromium&type=cs&l=5>

<https://code.google.com/p/chromium/codesearch#chromium/src/base/message_loop/message_loop.h>

<https://code.google.com/p/chromium/codesearch#chromium/src/base/debug/task_annotator.h&q=task_annotator&sq=package:chromium&type=cs&l=1>

<https://code.google.com/p/chromium/codesearch#chromium/src/components/autofill/content/renderer/autofill_agent.h&q=autofill_agent&sq=package:chromium&type=cs&l=1>

<http://site.icu-project.org/>

<https://code.google.com/p/chromium/codesearch#chromium/src/content/child/resource_dispatcher.h&q=resource_dispatcher&sq=package:chromium&type=cs>

<https://code.google.com/p/chromium/codesearch#chromium/src/media/filters/decoder_stream.h&q=decoder&sq=package:chromium&type=cs&l=1>

<https://code.google.com/p/chromium/codesearch#chromium/src/extensions/README&q=extensions/Re&sq=package:chromium&type=cs&l=1>

<https://docs.google.com/presentation/d/1dDE5u76ZBIKmsqkWi2apx3BqV8HOcNf4xxBdyNywZR8/edit#slide=id.g9ade3ed5_017>

<https://code.google.com/p/chromium/codesearch#chromium/src/third_party/WebKit/Source/platform/image-decoders/ImageDecoder.h&q=imagedecoder.h&sq=package:chromium&type=cs&l=1>

<https://code.google.com/p/chromium/codesearch#chromium/src/content/common/gpu/client/gpu_channel_host.h&q=gpuchannelhost&sq=package:chromium&type=cs&l=1>

<https://code.google.com/p/chromium/codesearch#chromium/src/cc/raster/task_graph_runner.h&q=taskgraph&sq=package:chromium&type=cs&l=42>

<https://code.google.com/p/chromium/codesearch#chromium/src/gpu/command_buffer/client/query_tracker.h&q=querytracker&sq=package:chromium&type=cs&l=77>

<https://code.google.com/p/chromium/codesearch#chromium/src/gpu/command_buffer/service/query_manager.h&q=querymanager&sq=package:chromium&type=cs&l=34>

<https://code.google.com/p/chromium/codesearch#chromium/src/content/common/gpu/client/command_buffer_proxy_impl.h&q=commandbufferproxy&sq=package:chromium&type=cs&l=47>