Scheduling Scripts Intuitively and Performantly

kouhei@chromium.org
bit.ly/script-scheduling

[PUBLIC]

Script scheduling is a complex space. This document summarizes the script scheduling primitives available today, categorizes its problems, and presents a roadmap to solve them.

Recap: Script scheduling primitives available today

Before going into detailed discussion, here is a summary of what existing primitives would do:

	Loading prio. (net/Blink)	**Exec sched prio.**	**Where kouhei@ thinks they should be used in Chrome M65**
`<script>` in `<head>`	Medium/High	VeryHigh - Blocks parser	• Scripts that affects layout of FMP content • Scripts that must be run before other scripts Examples: • Framework runtimes (if not server-side-rendered) • Polyfills • A/B testing that affects DOM structure of the entire page
`<script type=module async>` or	Medium/High	High - Interrupts parser	• Scripts that generates critical content (needed for FMP) ○ But shouldn't affect above-the-fold layout of the page • Scripts that triggers network fetches of dynamically inserted contents

`<link rel=preload>+ <script async>` hack			Examples: ● Draw something on <canvas>
`<script async>`	Lowest/Low	High - Interrupts parser	*kouhei@ personally can't find any convincing use-case for this :/*
`<script defer>`	Lowest/Low	VeryLow - Runs after `<script>`s at end of `<body>`	● Scripts that generates non-critical content ● Scripts that provides key interactive features that >50% of page view sessions would use Examples: ● Ad frameworks ● Framework runtimes (if server-side-rendered)
`<script>` at the end of `<body>`	Medium/High	Low - Waits parser end	*kouhei@ personally can't find any convincing use-case for this :/*
`<script defer>` at the end of `<body>`	Lowest/Low - end of the queue	VeryLow - Runs after `<script>`s at end of `<body>`	● Scripts that provides interactive features that may be used occasionally Examples: ● Load "Related articles" ● "Give feedback" feature

Note: All non-async <script>s' execution will block both DOMContentLoaded and OnLoad. Async <script>s can also be executed before DOMContentLoaded and OnLoad if it is loaded fast enough.

TODO: share the test page URL+code

Problem

The current script prioritization primitives, namely `{defer, async}`, are

- [A] Misleading

    - kouhei@, as one of blink script sched. infra maintainer, needed to lookup HTML spec and write some test code to verify their actual semantics (in order to fill in the table above)
    - Feedback from developer facing teams that they are hard to communicate
        -
    - Almost all page loading performance audit sessions expose their misuse
- [B] Inconsistent
    - `async` scripts are loaded at low priority, while its exec scheduled at high priority
        - `<link rel=preload as=script href=x.js><script async x.js>` hack was invented to make both {loading,scheduling} priority high.
    - `defer` isn't really different from "late scripts", but their loading/exec schedulers are swapped

The increasing complexity of web apps are demanding the following **flexibility**:

- [C] Dependency scheduling
    - A script may schedule load/exec of other scripts.
        - **ES modules**: A ES module may import other ES modules, and those dependent ES modules may also have imports.
        - **Loader scripts**: Some scripts act as a loader which select the right variation of the script needed, using information only available at runtime.
            - They often appear in 3rd party lib deploy "snippets"
    - The current primitives are only focused on specifying load/exec priority for top-level refs from <script> tags. We are currently lacking primitives to specify loading priority for dependency scripts.
- [D] Inter-resource type scheduling
    - Script loading shares the same network bandwidth / request queue / main thread time with other types of resources, like CSS/image/fonts. We need a primitive that can express the priority relative to other type of resources.
    - Inter-resource type dependencies: A script may depend on non-script resource to be meaningful.
        - Example: A script that draws interactive graph to <canvas> may involve painting PNG images.

There are also some **performance considerations** that are not captured by the current primitives:

- [E] Early priority indication
    - The correct priority intents need to be expressed as early as possible, preferable in priority descending order. (High priority first, low priority last)

    - We have little control over their priority after the requests are sent out. Round-trip times and TCP Recv buffers.

Roadmap

kouhei@ discussed various proposals at BlinkOn9, and this section outlines our roadmap. ☂crbug

Short-term (18Q2 or Q3)

Change `<script async>` execution scheduling priority to low

Problem:

- [A] Misleading: `<script async>` today, is more often used to indicate non-critical scripts.
- [B] Inconsistent: `<script async>` has an inconsistent priority of being
    - loaded at low priority, and
    - executed at high priority.

Solution:

- Change the execution scheduling priority of `<script async>` to low priority.
- Keep the execution scheduling priority at high iff:
    - Priority hint is specified. (e.g. `<script async importance=high>`), or
    - If the `<script async>` has a matching `<link rel=preload>`

Open Questions:

- What is “low” scheduling priority?
    - Just before flushing the defer script queue (“scripts to be executed before parser end”)

AI(OWNER):

- AI(kouhei): Create an experiment CL and Finch it -> Started

Bound priority of fetches to that of parent script

TBD

AI(OWNER):

- AI(tbansal): Share design sketch with kouhei@,hiroshige@

Change <script> at the end of the <body> loading priority to low

Problem:

- [A] Misleading: <script> tags at the end of the <body> is currently scheduled at Medium/High priority, but page authors' intent is for low priority scripts.

Solution:

- Implement a heuristic where:
  - If the <script>s look like they are at the end of <body>
  - Schedule them at Low priority

Design sketch by kouhei@:

- In HTMLDocumentParser::QueueTokenizedChunk(), if enchunk looks like a EOF/end-of-body, mark the continuous starting_script TokenizedChunks at the end of the speculations_ queue as they are likely to be end-of-body script chunks.
- In PumpPendingSpeculations(), plumb that if the chunk is a end-of-body script chunk, so that it can pass the info to HTMLScriptElement ctor via CreateElementFlags
- ScriptLoader query if the ScriptElementBase==HTMLScriptElement is at the end-of-body, deprioritize script fetch request if so.

AI(OWNER):

- AI(kouhei) sketch out implementation design here -> help tbansal@/bengr@ team work on it.
  - Still looking into this. The fact that we synchronously flush the tokenizer chunk when we hit </script> makes this non-trivial.
  - Update; Looks like most of the old logic can be moved away. kouhei@ to create minimal refactoring CLs here, and pass over to tbansal@

- AI(kouhei) deprecate tokenized_chunk_queue_ and remove chunking heuristics which no longer make sense in BackgroundHTMLParser actually running on the main thread. -> CL in flight -> landed

Priority Hints

WICG/priority-hints is a WICG effort sketching out new API/markup to indicate intent to relatively adjust the request priority from the web browser calculated default priority. Priority Hints introduces `importance` attribute, which can be specified on `<script>` and other elements which have external resource dependencies.

Problem:
In the script scheduling context, the Priority Hints proposal is much more intuitive than the existing primitive (solves [A] Misleading). The proposal also partially addresses inter-resource type scheduling [D] Inter-resource type scheduling by giving other types of resources which didn't have any scheduling primitive.

If we chose to align `<script async>`'s priority to low, `<script async importance=high>` may be a useful primitive to provide high loading/exec priority.

Solution:
- Add Priority Hints `importance` attribute support to `<script>`

AI(OWNER):
- domfarolino@gmail.com is currently working on Priority Hints implementation on Chromium (I2I thread)

`<script defer>`s should yield tasks in between

Problem:
- Blink currently schedules all `<script defer>` scripts in one task.

Solution:
- Yield in between `<script defer>`

AI(OWNER):

- AI(kouhei): Find an OWNER

Long term

WebPackaging: Native Bundles

Chrome loading team is currently exploring WICG/webpackage as a new way to distribute web apps.

The Web Bundle format can be used to bundle the entire web app into a single file. In such case, the order of the subresources in the bundle will exactly match the order the web browser loads the subresources.

Solves [A,B,C,D,E] in the sense that it forces a single loading schedule. How would execution scheduling look like?

Loading Manifests

A Loading Manifest communicates the entire dependency graph of an web app upfront.

AI: Kenji to expand his thoughts? Seems to solve [A,B,D,E] completely, while we need loading worklet to solve [C]

Loading/Execution Worklet

There is an idea of introducing loading worklet, which can intercept fetch request triggered from a document, and adjust their priority before dispatched to the network stack.

Loading checkpoints

On top of the “importance” indication that priority hints would provide, maybe we need a way to tell the browser *when* a script should be executed. Examples:

- Before first paint (but don’t block parser and don’t SPOF)

- After FCP
- Before DCL (current “defer”)
- After DCL
- Before onload
- After onload
- Other??

Open Questions

Is there a use case that benefits from specifying different loading/execution scheduling priority?

kouhei@ isn’t aware of such use-case, so would like to know about this more.

Proposals

Deprecated: Superseded by AIs section.

There are several proposals that aim to solve the issue. They are in the order of their difficulty/needed-time-to-ship (easy ones first).

Align `<script async>` loading/exec priorities to High

In crbug.com/690550, pmeenan@ propose raising `<script async>`’s loading priority to Medium/High, so that they match their High execution priority.

The proposal focus on addressing inconsistency ([B]), as a consequence this will hopefully help the confusion with the existing primitive ([A]).

Align `<script async>` loading/exec priorities to Low

Alternatively, we can align `<script async>`'s loading/exec priorities by lowering its execution priority. Like the original proposal in the previous section, the focus is on inconsistency ([B]).

In the audit sessions, we have seen many web developers trying to down-prioritize scripts by adding async. If “lowering” priority of the script is more popular developer intent, then this may increase the chance of Chromium matching developer intent.

If we take this route, then it would be useful to provide additional high loading/exec priority primitive for the completeness.

WebPackaging native bundles

TODO: write

Solve [A,B,C,D,E] in the sense that it forces a single loading schedule. How would execution scheduling look like?

Loading manifests

Kenji? Seems to solve [A,B,D,E] completely, while we need loading worklet to solve [C]

Loading/Execution Worklet