GPU Web 2020-01-13

Chair: Corentin
Scribe: Ken, Austin, Kai
Location: Google Meet

TL;DR

- Changes to adapter discovery #521 #522 #523 #524
    - requestAdapter**s** needs to interact correctly with features like Privacy Budget and avoid spending all the pages' budget.
    - requestAdapter.s needs to interact with various device dimensions, their "tee shirt size", compute-only vs not, etc.
    - Discussion about what constraints there should be on the API so that pages don't break in a browser while working fine in another.
    - Consensus to land the change to requestAdapters with loose semantics that we'll tighten up later.
- Add read-only storage textures as new binding type and texture usage #532
    - Consensus: accept with both readonly-storage for texture usages and for binding types.

Tentative agenda

- Changes to adapter discovery #521 #522 #523
- Add a validation rule for buffer binding #526
- Add read-only storage textures as new binding type and texture usage #532
- Keeping data on chip #435
- PR burndown
- Agenda for next meeting

Attendance

- Apple
    - Dean Jackson
    - Justin Fan
    - Myles C. Maxfield
- Google
    - Austin Eng
    - Corentin Wallez
    - James Darpinian
    - Kai Ninomiya

    - Ken Russell
    - Shrek Shao
    - Ryan Harrisson
- Intel
    - Yunchao He
- Microsoft
    - Chas Boyd
    - Damyan Pepper
- Mozilla
    - Dzmitry Malyshau
    - Jeff Gilbert
- Mehmet Oguz Derin
- Timo de Kort

Administrivia

- CW: For F2F, connectivity system is Microsoft Teams
- CW: Normally, there would be a link that anyone can join, leading to the meeting, but it will be a different system.
- CW: Should start drafting an agenda for the F2F.
- CW: If you haven't told Rafael that you are coming to the F2F, please do.

Changes to adapter discovery #521 #522 #523 #524

- CW: discussed this last week. More important one is #524, but didn't resolve on any of these.
- CW: basically: #524 makes requestAdapter return a list -> requestAdapters. We believe no worse fingerprinting than current API, but would like more opinion on this, esp. from Apple's side.
- MM: not caught up yet. Apologies.
- KN: requestAdapters would be more flexible, and not any more prone to fingerprinting. Browser can always decide what it wants to return through it. Would let apps know what actual list of GPUs is, rather than apps having to come up with all possible adapter requests & make them. Expect high-perf apps to do this; they want info about available GPUs. Fingerprinting concern with exposing lots of adapters, but this doesn't change the fingerprinting concern. Brings closer to frameworks like Node & Electron too. Didn't come to any kind of conclusions on this last week, so not 100% sure what we need to discuss. Would be good to have more feedback.
- JG: last week, think we demonstrated that it doesn't have to be any worse for fingerprinting, and is more ergonomic to allow applications to peruse the list of adapters, even if they just want to look at both the high- and low-power adapters and choose one. Also helpful for compute nodes; want to run on anything they can, and would want to run

on "all" adapters. No solution for this right now. E.g., two dGPU video cards and want to run stuff on both, current API doesn't have a solution for this.

- KN: yes that's an imp't use case I forgot.
- MM: I think there's two pieces here, and sorry I couldn't join last week. The question is: In an API where the browser delivers a list of all the available adapters, (one world). On another world, the application asks for a set of criteria. The observation is correct that both have some set of reachable adapters. The second one, though, can interact with the privacy budget idea. The browser can watch what the application is asking for, and can start rejecting requests if the application asks for a ton. There can actually be more addressable adapters in the world where the application asks. In the world where the API returns a list, we might prune the list. Whereas if the app asks with some features, we might only say you get the first one or two you asked for.
- KN: interaction with the privacy budget idea is interesting.
- JG: one thing we discussed last week was: just because the API returns an array doesn't mean you have to enumerate all the adapters. Could offer as few as 0 or as many as there are.
- MM: Right, but the difference is the priorities. In the world where the browser delivers the array, it doesn't know how to prune the list. Whereas if the application explicitly asks for something, the browser knows what the app is looking for and can service those requests.
- JG: doesn't necessarily mean we're going to get rid of picking a priority when returning a list of adapters. Think the main proposal was to select low-power devices, and you get a list back of those, in sorted order. Similar to how D3D enumerates adapters. Typically, integrated GPU is first, but depending on power preferences that swaps with the discrete GPU.
- KN: MM, if I'm understanding correctly, the browser would sort of measure the number of identifying bits that the application has access to.
- MM: yes, that's the privacy budget idea.
- KN: think we should take that into consideration, and that requestAdapters (plural) is the right way to address it. Can easily add ways to request a shorter list (i.e., "I only want one") and apps can use that. Then they won't use so much of the privacy budget. Think it's worth it for flexibility / compatibility. Also don't think it makes things more significantly complex.
- DM: how does this interplay with budget idea?
- KN: if we've only informed web page of single adapter then we know it only knows about that single adapter. If we give it a full list then we know it knows that full list. If we wanted to clamp down on the amount of info we give thru adapter list, either make the list shorter (don't blow privacy budget on requestAdapters), or allow ways to have them make the list shorter.
- MM: Then there would still be a set of criteria that the app is asking the browser for. It's just that the response might have multiple adapters in the response.
- KN: yes. That's essentially the API I proposed, but a little bit more. Was intending that there would be a powerPreference. Don't want it to be too complicated. Should be easy

for developer to say, adapter 0 is a good default. Would put more limits on what comes back. Could get different lists back at different points in time.

- MM: good default idea is a good one. But for good ergonomics it should probably be a separate API, like getDefaultAdapter. Also should be able to request specific properties of adapters. Relying on the first element of the list probably isn't great.
- KN: depends on the criteria. Right now we don't have adapter selection criteria. Only powerPreference. In list model, only affects order of the list, not its contents.
- CW: eventually we might have constraints on the adapters. Have to see with WebML CG. Like, WebGPU adapter is compatible with WebML device. And if there are compute-only adapters, wouldn't be able to present to canvas, but might want that constraint (being able to present).
- MM: also buckets - group devices into relatively small number of buckets. Give me an extra-large device, for example.
- CW: also question on previous discussion: is there a public doc on privacy budget and how it works?
- MM: it's a document authored by Google.
- CW: thought there was one for Safari as well. The better we know how browsers might go, the better we can design our API.
- RC: are all browser vendors agreed that the privacy budget is the way to go?
- MM: Safari hasn't agreed to it yet. Shouldn't design in contradiction to it though.
- KN: I don’t think that requestAdapters (plural) is totally inconsistent. I think it can be made to work with it.
- DM: what would the budget be? You can make X requests and the total up to Y adapters to return?
- MM: the browser watches everything the page does. As soon as the budget is exceeded, the browser will start returning zeros, false information, noisy information, etc.
- JG: so, the thing we have right now, make it plural, keep the low/high power requests in consideration?
- KN: we have to return the whole list of adapters, unless there's a way to filter the results.
- JG: think we need to be able to pluralize this, but we're just figuring out how to.
- MM: probably OK to keep design we have, but allow browser to return multiple. Website should give criteria to the browser so the browser knows what the website cares about and which adapter would be a good one to hide
- KN: Yes, that is consistent with what I have. At some point I say that the hints can't affect the contents of the list, and maybe we want to remove that.
- DM: don't we want devs to be able to say how many at most devices they want?
- KN: yes I think so.
- JG: I'd like that to be something we reactively do later
- KN: we could add that later. I'm also OK with keeping the single-return-value versions. If there were a singular requestAdapter, maybe no point in passing powerPreference.
- MM: think the default in Metal and D3D there's a system default device. Not sure about Vulkan.
- DP: it's a little complicated in D3D, takes into account state of system and the app.

- RC: In general from an app point of view, there’s an enumerate adapters, and generally it’ll just give you one, and there’s EnumerateByPowerPreference, which will change that ordering.
- DP: wondering if you could limit the amount of information exposed in GpuAdapter by default. So, maybe not the name, only the list of extensions you care about.
- KN: I didn't think about that but it would be really useful to keep in line with privacy budget.
- MM: Similar to the bucketing idea, where we’d take many devices with similar properties, lie about them, and say they’re all the same.
- JG: this is the feature levels idea.
- KN: we can definitely do that, everything the browser does would allow that to be done in certain use cases. Can't see GPU differences except browser says one is higher-perf than the other.
- CW: sounds like there's appetite for requestAdapters, but need to keep path forward for privacy budget, and keep app in control of how much privacy budget they use with their requests?
- JG: yes that sounds right but we have different priorities. Must look out for ways we might make it hard to follow the privacy budget.
- KN: That’s what I was thinking. We should try not to design into a corner, but I don’t think it’ll be hard to do that. The rest of Web APIs have already been designed, and they will also have to do this privacy restriction. I feel it needs to stabilize outside of our group.
- CW: OK. What should we do with this set of pull requests?
- JG: could we come to some consensus that it's requestAdapters, but you have to specify a powerPreference style "options", and we don't mandate that the lists you get back for different options are the same size?
- KN: That’s what I’m in favor of, and we’ll figure out over time the contents of the options.
- JG: my only concern is that this doesn't address requestLowPowerAdapter/HighPowerAdapter/Big/SmallAdapter etc.
- KN: I think there should be an option for applications to get all of the information they can get, if they really want to, and if it uses up the budget, then so be it. They can have that and not get other information after. And we’ll resolve that when privacy budget is more developed.
- JG: OK. I think the first idea - you call requestAdapters, and based on powerPreference it returns some adapters, but maybe not all of them?
- KN: Only need to take out one line: “power preference only affects order, not contents”
- JG: great.
- MM: can we also say that if you don't specify any requirements / filtering criteria, that you might not get every device? Trying to avoid every website in the world using WebGPU tries to iterate all devices & pick one it wants. That's an architecture WebKit opposes, because we try to not provide all information in a single call. Want to make sure every WebGPU site people make doesn't assume Chrome's behavior.
- KN: If they request everything and get only one, that will be the same as Chrome behavior on almost every device (most have only one adapter). It’ll look the same.

- MM: there's a difference between "give me every device" and "give me what I need". In Chrome's approach, if you give them the one device they need, it'll work. If in Safari we give them one back and it isn't what they need, they might toss up a banner.
- KN: Not concerned about websites saying “you don’t have what I need” because they’ll get a single adapter, and if they want to use it they will. What is concerning, is if they provide no options because they think they’ll get everything -- and then turns out they only get one on Safari.
- MM: that's what I was trying to say.
- KN: Let’s think about that. I don’t have an answer right now.
- JG: to make progress should we say you can only request *with* option restrictions right now?
- MM: And also have a way to get some default adapter that will work. Important for new developers.
- JG: I think if you request low-power, we’re not going to not return anything. We’re just going to give you the lowest power device.
- CW: going more in that direction: how about having requestAdapter, and instead of taking "AdapterConstraints", it takes "WorkloadDescription"? Then we give them what we think will be the best thing. Sort of like Metal's CreateDefaultDevice.
- JG: I think that adds complexity. I don’t think it’s the easiest path forward.
- CW: my worry is that I don't see a good way for WebGPU to return a list of adapters, and correctly paper over browsers not returning the full list. Someone might say, I know Safari doesn't return the whole list, so I'll ask for the low-power and the high-power devices and take the union of them.
- JG: I will say that if we make progress on this, and don’t solve the fact that people have to ask for both high- and low-, that’s not super high on my list of priorities. I think being able to return multiple adapters is the biggest thing to solve now.
- CW: ok.
- JG: for follow-up: no way to request "I need float32 rendering" without enumerating the devices, and finding the lowest-power device supporting that feature.
- MM: If we do the t-shirt sizes thing, then you would just request the one that has float32 rendering.
- JG: I don't think that's compatible with the number of features we have .Have to be able to request multiple feature levels. Feature Level 3 isn't a straight linear line. Desktop line, mobile line.
- MM: I should do the work to come up with a proposal for the feature levels.
- RC: there's also not a great intersection. For example, "find the top GPU on this dell desktop". Then on this computer they put a high-end GPU from a few years ago and it doesn't have ray tracing support. Best feature support != always the highest-power GPU.
- CW: consensus I hear: requestAdapters returns a list. In the spec, remove any constraints about what the browser has to return. requestAdapters with no options might not return the whole list of adapters.
- RC: Can we at least say if there’s more than one, it’s in sorted order of what you put in the AdapterOptions? Like high → low if you specified high-power.

- CW: I think that's fair.
- MM: I thought we were going to say that the only one we would return in that case is the high-power one.
- KN: from a practical standpoint I don't think that works. App wanting high-power GPU doesn't want to exclude itself from low-power GPUs.
- RC: I don’t think what you’re suggesting will work on Windows. Listing adapters, does return a list, and I as a browser don’t want to get in the business of trying to filter that list based on what we think the app wants.
- DP: I checked with the team caring about this and they want the entire list returned, not have the browser filter it.
- MM: I agree with that sentiment, but we have to deal with privacy.
- DP: I understand why we have to limit what comes back in the adapter list. For example, the name. The app doesn't care about the name, but that it supports e.g. ray tracing. In terms of privacy / fingerprinting, telling them just that info should be good.
- CW: should have second discussion about removing the name.
- KN: We can also detect whether or not the app reads the name attribute. We don’t have to change anything.
- JG: you can request one adapter name per session? :)
- DP: is the number of adapters present a key thing we have to hide from the app?
- JG / CW: Potentially. Up to the user agent. Want to enable people who have a ton of powerful GPUs.
- KR: Let me point out one tangible issue: As Chrome changed power preference default to lowPower, the maps team wanted to use highPower, when available. So they had to ask high-performance for ALL user agents.
- MM: The maps team can change that. They can make multiple requests.
- JG: we're trying to learn from WebGL here and make it a request, not a statement. "Give me the highest performance GPU on the system". But if you only have one GPU then we give you that one.
- CW: Okay, can we extract some form of consensus?
- KN: I'd like to land the most underdefined version of what we've discussed and try to refine it later.
- JG: I think if you remove the line about the list being the same length, it’s mergeable.
- MM: I think that's OK - as long as we don't remove the part about the web site having to request specific properties.
- KN: We don’t have that right now. requestAdapter only have powerPreference.
- MM: so you're requesting a powerPreference.
- KN: but not requesting other capabilities.
- MM: is this an extensible set? It should be.
- JG / KN: it is.
- JG: in the WebGL world at least, it was powerPreference and failIfMajorPerformanceCaveat.
- KN: I added that to WebGPU too in this pull request.
- JG: think we might end up like "isXRPrimary" or similar.

- CW: let's land that change with the most undefined version, and figure out the constraints of giving the application what they want based on what they asked for. Surely can get one low-power device if no high-power device in the system.
- MM: when you do land it, appreciate it if you say that different requests may get you different adapters, and the set of adapters returned can be any subset of the system's.
- KN: we don't say anything about the system adapters in the spec right now, but will make sure.

Add read-only storage textures as new binding type and texture usage #532

- CW: think the binding type for read-only storage texture is noncontroversial. Similar to read-only storage buffer. Think we don't need read-only storage usage. More controversial. Don't think it makes a huge difference.
- JG: seems superficially valuable to me. If we don't use this, we have to infer whether something's read-only in the shader, which I think we discussed.
- DM: we don't have to infer. Based on the binding type we'd know it's read-only. "Storage is mutable if X, immutable if Y". Just specifying, and clarity of the spec, is one problem. Also, our impl can make decisions based on whether it has to be immutable or not. if it has to support mutable storage, have to create UAV in D3D12/11. Otherwise can get away with SRV. It's minor, but could make a difference.
- JG: also: ability to declare your usage as readonly-storage-buffer was potentially useful for buffer mapping stuff, because you can make different restrictions based on that. In one of the proposals there was a restriction about CPU-mappable UAVs. Seems useful to me to be able to say read-only.
- RC: what happens if someone says it's read-only and then writes to it? Error we need to catch? Check the shader in any case?
- DM: that would be an error at BindGroup creation. BindGroupLayout would say read-only, we'd see that it requires READ_ONLY capability, or vice-versa.
- RC: I guess my point is you have to check the body of the shader.
- DM: the shader part is independent of this question. Shader only concerns whether we'd need the binding point to be read-only.
- CW: to restate: notion of compatibility between resources & shader is encoded in BindGropuLayout & PipelineLayout. We know both match by comparing the layout. Question is whether to add read-only usage at resource creation. Either way is fine.
- MM: Sounds like a good idea. Maybe a related question, is should we do this for buffers? It can be valuable to know for inserting barriers, and forcing one shader to run after another vs. the same time if the resource is only readonly.
- CW: if we add readonly usage for one then we should for the other too. Not sure about barriers. At BGL time we know if it's readable/writable. Can add both, makes it more explicit. I withdraw my objection.
- JG: table this for now and talk about F2F planning?

- CW: can we say, we agree with that PR? Any concern about adding read-only usage?
- MM: r=me
- JG: r+
- CW: OK, great, and let's add read-only storage buffer later.
- DM: let's discuss the name later.

Keeping data on chip #435

-
-
-
-

PR burndown

-
-
-
-

Agenda for next meeting

- F2F:
  - CW: Anything to know about Microsoft Teams?
  - RC: conference room doesn't support WebEx, Hangouts, etc. If your laptop doesn't support Teams then you'll have to send me stuff to present. Everyone know address and date?
  - RC: we will be having breakfast, lunch & afternoon snack. Will order those later.
  - CW: Thank you; any questions?
- CW: Same agenda. More about adapters. Stretch to keeping data on-chip
- MM: Won’t be here next week.
- Also a US holiday.
- MM: let's defer keeping data on chip until the F2F.
- CW: OK. First thing on the agenda.
- CW: OK to cancel next week? (Yes)
- CW: two weeks: requestAdapters, PR burndown, F2F planning
- CW: ideas for F2F?
  - CW: setBufferSubData?
    - MM: good one
  - MM: would be helpful to have updates from all the teams
  - CW: shading language, will have something to discuss
    - MM: yes, let's put that agenda item down
- CW: SwapChains?

    - KN: not this time
- CW: James, anything from WebML?
    - JD: not this time
    - MM: a WebML synopsis would be helpful
- CW: I'll start an agenda for the F2F tomorrow. Can add stuff directly to it so we remember. Will send a link tomorrow. Add link with more ideas.
-
-
-