Start Date	EndDate	Please select which best fits your experience.		Please select what most closely matches how you got started with block themes +
		Response	Other Text	Response
2021-07-30 07:19:10	2021-07-30 07:42:20	I have built and launched block themes.		I started from scratch.
2021-07-30 08:43:25	2021-07-30 08:45:56	I have built and launched block themes.		I started from scratch.
2021-07-30 07:15:14	2021-07-30 09:53:40	I have built and launched block themes.		I started from scratch.
2021-07-30 10:01:24	2021-07-30 10:13:12	I have built and launched block themes.		I used the empty theme from theme experiments.
2021-07-30 09:37:54	2021-07-30 10:12:32		I have explored using theme.json with a classic theme and I have experimented with building block themes.	I forked an existing theme (share in the comments of this question which theme).
2021-07-30 09:24:27	2021-07-30 10:13:11	I have built and launched block themes.		I started from scratch.
2021-07-30 10:23:56	2021-07-30 10:26:14	I have explored using theme.json with a classic theme.		I started from scratch.
2021-07-30 17:32:26	2021-07-30 18:28:57	I have built and launched block themes.		I forked an existing theme (share in the comments of this question which theme).
2021-07-30 22:35:17	2021-07-30 22:49:43	I have built and launched block themes.		I forked an existing theme (share in the comments of this question which theme).
2021-07-31 06:55:30	2021-07-31 07:01:11	I have explored using theme.json with a classic theme.		
2021-07-31 07:27:29	2021-07-31 07:39:10		I'm currently building a hybrid theme for a fairly large SaaS client that relies on theme.json and Block Patterns	I started from scratch.
2021-08-01 07:34:13	2021-08-01 07:35:16	I have experimented with building block themes.		I used the empty theme from theme experiments.
2021-08-01 14:11:06	2021-08-01 14:14:31	I have experimented with building block themes.		
2021-07-31 00:39:14	2021-08-02 02:12:12	I have built and launched block themes.		I started from scratch.
2021-08-02 04:07:23	2021-08-02 04:20:34	I have built and launched block themes.		I started from scratch.
2021-08-02 04:14:21	2021-08-02 04:24:48	I have built and launched block themes.		I forked an existing theme (share in the comments of this question which theme).
2021-08-02 10:00:41	2021-08-02 10:02:35	I have explored using theme.json with a classic theme.		I forked an existing theme (share in the comments of this question which theme).
2021-08-02 16:54:44	2021-08-02 16:57:35	I have experimented with building block themes.		
2021-08-03 02:41:22	2021-08-03 02:45:54	I have built and launched block themes.		I forked an existing theme (share in the comments of this question which theme).
2021-08-03 03:29:52	2021-08-03 04:17:44		I've built 5 experimental block themes and explored theme.json with classic themes	
2021-08-04 03:37:00	2021-08-04 03:42:35	I have explored using theme.json with a classic theme.		I used the empty theme from theme experiments.
2021-08-05 03:22:24	2021-08-05 03:32:51	I have built and launched block themes.		I forked an existing theme (share in the comments of this question which theme).
2021-08-05 18:39:35	2021-08-05 18:45:31	I have built and launched block themes.		I started from scratch.
2021-08-08 18:42:14	2021-08-08 18:49:49	I have built and launched block themes.		
2021-08-09 10:38:15	2021-08-09 11:07:34		Built block theme, Multisite experimentation, focused mostly on "universal/transition" block theme/classic theme	I started from scratch.
2021-08-10 02:02:34	2021-08-10 02:08:02	I have built and launched block themes.		I started from scratch.
2021-08-10 09:53:35	2021-08-10 09:58:06	I have explored using theme.json with a classic theme.		I started from scratch.
2021-08-11 10:11:53	2021-08-11 10:26:10		I have experimented with block themes and fse. I am currently building a custom block theme for a client site.	I forked an existing theme (share in the comments of this question which theme).
2021-08-12 01:30:22	2021-08-12 01:51:04	I have built and launched block themes.		I started from scratch.
2021-08-13 05:48:54	2021-08-13 07:34:48		I tried using theme.json in a classic theme and also experimented with TT1-Blocks theme and FSE. But haven't dig in too deep to create a fully custom FSE solution.	I started from scratch.
2021-08-13 08:25:42	2021-08-13 08:36:41	I have experimented with building block themes.		I forked an existing theme (share in the comments of this question which theme).

Other Text	Other Text
	I have used all three. I no longer use empty theme because it is too basic for me. When I fork, I copy one of my earlier themes like Armando.
	TT1
	Started with Blockbase but ended up restarting on my own, as it was a bit too much how it worked as a parent theme.
	I have forked an existing theme, used an empty theme, and have started one from scratch. The bulk of my work has come from forking a theme that I last worked on in 2019. The original GitHub repo is here: https://github.com/justintadlock/exhale I plan to release this as a new theme for free at some point. I am mostly waiting for some of the FSE features to mature. Primarily, the biggest missing pieces are: - Navigation block to settle down - An array of post author blocks - A robust set of comment-related blocks - Post Featured Image block to have a sizing option I think I could realistically release a use-at-your-own-risk beta version of my theme today if those items were addressed.
	I followed instructions from https://fullsiteediting.com/
My own 'minimal theme' that I have developed as a starting point for custom Themes for the last twelve years	
Using my Genesis base child theme and implementing/testing what I have come across over several resources online.	
	I felt this would make for a better learning experience with regards to how FSE works, and I've only ever built themes from scratch.
	Gutenberg Starter Theme from Github
	I extended an existing custom theme from our agency.
I followed a tutorial by Frank Klein	
	TT1 Blocks
All of the above.	I started with experimental-theme.json and had to convert it to theme.json. For some themes I started with a completely empty file, others I copied. Search : https://github.com/bobbingwide/thisis/search?q=theme.json&type=issues
I converted an existing theme to a block theme.	
	I have done most of my experimenting with tt1. I have been referencing and digging into the code many block themes, trying to come up with the best methodology for my custom theme which will need to be production ready at end of month. Blockbase, seedlet-blocks, astra-blocks, genesis block theme.
	We started from a classic theme, created a blocks plugin (Nova Blocks) around it, and step-by-step evolve towards a block theme (Rosa2). It's not a fully block-based theme, but the goal is to get there soon.
	As an established theme developer, I wanted to try benefits of theme.json file when transiting from a classic theme to new FSE experience. I tried to incorporate theme.json into my current classic theme mostly.

What templates and template parts do you always include in your block-based themes?	**How do you use colors within theme.json?**	
	Response Other Text	Comments
Templates: Index 404 single page archive search I use both single and page instead of singular, because I want to show the post meta information for posts, but not for pages. There is no conditional for this, like there is for PHP based themes, so, I have to use two separate templates. The only template part that I use in every theme is the footer. I have also used: header main comments sidebars I no longer believe that template parts are suitable as sidebars, since you may want to change the content in the sidebar depending on the page / context. A column pattern may work better for this.	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	I have used names and slugs with color names because I thought they would be easier for users to understand. I can't verify this theory. Using semantic *slugs*, in theory, has the benefit of being more consistent when you switch themes.
A wide range.	I use semantic names for colors like “primary, secondary, foreground, background”	
Templates: Index Page 404 Single template-parts: Header Footer Main(Loop for the blog listing code)	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
Template Parts: - header, footer, main Templates: page, index, single, page-home, page-about, page-news, 404	I use semantic names for colors like “primary, secondary, foreground, background”	You can find more about it here: https://github.com/anarieldesign/clove/pull/1
Index, header and footer	I use semantic names for colors like “primary, secondary, foreground, background”	
header/footer really is all I've used so far.	I use semantic names for colors like “primary, secondary, foreground, background”	Semantic 100%. I use the slugs are semantic values, but keep the labels as actual relatable names (Blue/primary), (Purple/secondary). https://richtabor.com
Haven't got that far yet	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
I pretty much include all of the top-level templates at this point. Because the template part system is static rather than dynamic, it means that I often need to repeat code in the top-level templates. With PHP, it was far easier to follow the DRY principle and avoid duplicating code. For example, if I want to create two different "content" template parts for single pages and posts, I cannot use a top-level singular.html template alone. I must break that apart into separate single.html and page.html templates that call separate content-single.html and content-page.html template parts. Ideally, I would be able to dynamically include the content-*.html template parts from one top-level template. Common template parts I use (broken down by hierarchy): - Header - Content - Loop - Sidebar - Footer	I use semantic names for colors like “primary, secondary, foreground, background”	I use semantic names with a Tailwind-like shading system. For example, my "primary" color gets split between different shades from 100 (lightest) to 900 (darkest). So, you get primary-100, primary-200, ... primary-900. These can be reduced or added to on a per-project basis. The system is based on the popular Tailwind CSS framework, but it also closely models common systems of theme authors I have surveyed. They tend to have neutral, primary, and/or secondary colors across the average project. I have described this in more detail here: https://github.com/WordPress/gutenberg/issues/29568#issuecomment-851579180 I also have a theme.json Gist of this system here: https://gist.github.com/justintadlock/0003b82a1fc753b8bae54fad5b8cfd55
I’ve only done one, but used: - header.html - footer.html - index.html - page.html - single.html - archive.html - single-portfolio.html - archive-portfolio.html	Other, please explain...	I used a mix. For the main color: - dark-main, light-main, lightest-main For accent color: - dark accent, light-accent, lightest-accent For grays: - darkest-gray, dark-gray, light-gray, lightest-gray - Also - white and black
Just the regular header, footer, index, front-page, functions, single, page etc.	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
I believe my theme is not truly block based, but more of a hybrid. I’m using theme.json but have ` remove_theme_support( 'block-templates' );` enabled.	Other, please explain...	This project is for a SaaS company that has three products and each product has its own brand color palette. I’m using product-specific color naming conventions which more closely align with semantic approach.
None	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
n/a	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
Templates: page single search 404 Template parts: header footer query loop	I use semantic names for colors like “primary, secondary, foreground, background”	
This is an ecomemrce theme using Woocmmerce blocks and our custom block library. Templates: Front Page Single Post Archive Search 404 Page Several Post templates and Page temmplates with sidebars Template parts: Header Footer Shop Archive-product (categories, tags, atributes) Loop and a few other templates (e.g. loop with sidebar) depending on the setup	I use semantic names for colors like “primary, secondary, foreground, background”	
404 archive comments footer header index search single	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
I haven't built a block-based theme yet.	I use semantic names for colors like “primary, secondary, foreground, background”	
n/a	n/a	
Template Parts: Header, Footer Templates: Index, Archive, 404, Page, Single	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	I use name that color https://chir.ag/projects/name-that-color/#6195ED to generate unique name for colors.
A lot more than most of the experimental themes. See https://blocks.wp-a2z.org/oik-themes/ I failed to do this for written, but I should have included 404.html since 1. It's not easy to create in the Site Editor 2. The default template is useless for Not Found 2. Is probably true for search.html as well.	A mixture of the above	For both colours and font sizes I like to know the details. In Written I put this information into the name attribute for font sizes. eg. { "slug": "extra-small", "size": "12px", "name": "Extra small 12px" },
N/A - have only explored using theme.json in a classic theme so far.	A naming system like Tailwind, eg primary-500 etc	
CorelDRAW	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
single, page, page with sidebar header, footer, sidebar	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	The !important that is added to all theme colors is DRIVING ME CRAZY. I hope we can fix this.
Templates: index, page, blank, 404, header-footer-only, archive, single, search Template parts: header, footer	I use semantic names for colors like “primary, secondary, foreground, background”	
templates -> index, single, page, archive, 404 template parts -> header, footer	I use semantic names for colors like “primary, secondary, foreground, background”	
Template parts: - Header - Footer - Navigation Templates: - Index - Single - Page - Archive - 404	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
I don't know yet.	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	
header and footer	I add color presets with names like “Blue, Red, Green”, and refer to those directly to use them.	I have been adding the color names. But am keen to the idea of the primary, secondary, etc. The problem for me in doing so has always been that they are too limiting. The designs I work off of in client theming are usually pretty complex and my colors do not always neatly fall into those categories. The theme I am working with right now has 3 main colors and lighter versions of each, as well as a black, white and gray. I am not sure what I would call primary and secondary for example.
We primarily use template parts for Header and Footer but evolving steadily to all other theme parts. Here is a list of all the templates: https://github.com/pixelgrade/rosa2/tree/main/template-parts	Other, please explain...	We used it for semantic names and here is the initial version: https://github.com/pixelgrade/rosa2/blob/758328d722d333ea50b6a65431e5b6c82b2352ea/experimental-theme.json Nowadays, we dropped the theme.json in favor of a custom color system. The colors are generated in Customizer and used through a tool that we call Color Signal: https://docs.google.com/document/d/1lqESfc_M1Ji2GHjXfoj-QhE5llF0zGDRhNlsKTptyuk/edit?usp=sharing
I used what's provided in TT1-Blocks theme for my testing.	Other, please explain...	Surely, this is a tricky question as there is no standard cross-theme compatible way of naming colors. Nowadays I kind of use a mixed naming strategy: - primary, secondary and tertiary (,...) for base theme color scheme, - an black, white, gray, dark gray, light gray for standard grayscale colors. All of these colors populate editor palette too. I do not set other colors if not required by the project as I feel a theme should adhere to some limited color palette (while still providing some options for basic colors of grayscale for possible tinting). I definitely tend to stay away from "Blue, Red, Green,..." naming convention to prevent a "wild west" designs :) This way I feel I give users the best color options to keep the design language of the theme/website consistent, while they can always use custom color option to set up any occasional "wild" color in their content.
Parts: header, sticky-header, footer. Templates: page, page-no-title, page-canvas, single.	I use semantic names for colors like “primary, secondary, foreground, background”	

Beyond colors and typography, what other settings do you use most frequently in theme.json?	Have you included per-block settings or styles in your theme.json files?	[Optional] Any other feedback you'd like to provide around what would be helpful to include in Core long term when it comes to theme.json?	Which blocks do you tend to customize with per block settings or styles in theme.json and why?
	Response		
Resetting the body margin to 0. Removing the left spacing on blocks with lists like the latest posts and latest's comments blocks. Specifying margins for featured images, template parts, columns.	Yes	Long term, theme.json may replace style.css and hold all the data from the style.css header. It may also have settings for some common options: Front page display blog/post Comments are enabled/disabled	I need to adjust the typography for these: Site title, post title, query title, post terms, date, post author, post excerpt read more link. I often adjust the color and spacing for buttons, query pagination, post navigation links, so that they look the same.
Padding margin, borders, duotone and palettes are also something I use along with custom values.	Yes	We need as many tools as possible in theme.json but an ability to use as few of them as we need.	Most of them, but my foundations are: quote, list, code and anything with an image.
Container Width Background color	No	How we can generate theme.json using tools or from WordPress editor?	
blocks, elements	Yes	Responsive/Mobile styles	buttons - border radius etc.
layouts and spacing	No	Cusotm configuration other than the shema provided by core.	
Im setting font sizes for captions (would be great if they could be more global, as all captions are styled the same typically), vertical spacing, horizontal content padding. Also missing CSS properties for layout sizes.	No	I tried added per-block settings but Im having a hard time getting that to work honestly. Need to dive in further.	
layout (sizing), spacing (allow padding/margin), remove custom radius from buttons	Yes	Not sure I've done enough to judge yet - mostly just migrating my existing add_theme_support options to theme.json	button - remove custom radius
I think I've touched every piece of `{ settings: {} }`, configuring things to my liking. Some common `settings.custom` properties that I set are: - Spacing values, with the most-used being a `spacing.global` value. - Content and wide-layout values because these are not currently exposed as CSS properties via presets. - Google Fonts system. It's a bit too early to say what I will use the most going forward though. It is still early. The biggest thing that is missing is a global spacing value as a preset. I have a feeling that will be a common use case for most theme authors.	Yes	This is a comment on the previous "Have you included per-block settings or styles in your theme.json files?": There was no open comments section for this, but it is one of the areas I wanted to expand on. For per-block settings, I have not done much outside of enabling an option or two if it was disabled by default. However, for per-block (and per-element) styles, I am not sold on the system. Writing CSS in JSON, which is essentially what we are talking about, feels wrong on so many levels. There is the obvious issue that it is limited to merely a few styles that are actually configurable, so anything beyond that requires diving into an actual CSS file anyway. And, that is problematic. Why would I want half my CSS code in a JSON file and the other half elsewhere? From a development standpoint, it makes the codebase harder to maintain. Initially, I started down the path of configuring per-block and element styles from `theme.json`. However, I have since moved my styling back to CSS files. It feels more natural, and I have the added benefit of all the tooling I am accustomed to. Right now, I cannot imagine a scenario where I would move back. But, there is also no real benefit (or none that I can see) of adding styles via `theme.json`. If that changes in the future, we'll see. I may be a convert. --- ## Other Notes We need a PHP layer for overwriting `theme.json` values. I have written about this as part of the FSE Outreach Program #8: https://wptavern.com/fse-outreach-round-8-a-developer-centric-call-for-testing-theme-json-configuration There is also an open ticket here: https://github.com/WordPress/gutenberg/issues/33367 There are two main benefits to such a system. Having a PHP API for piecing together configuration will feel far more natural to traditional theme developers. I look at it as a bit of an olive branch, a show of good faith, that the core/Gutenberg developers recognize that many theme authors will have an easier time easing into FSE features when they can do it via an API in a language that they are comfortable with. The second benefit is that there is an untold number of plugin ideas to extend Global Styles, site editing, etc. if there is an easy way to hook into the theme JSON system and overwrite things based on plugin options. I already have a pretty awesome idea around a font-family system that is now only possible through some crazy, hacky workaround. A simple filter hook would make this painless.	I've mostly enabled border support for the Button block in the past (not sure if that is enabled by default now). Adding default spacing to various blocks is another thing I've done. But, the biggest use case has been settings some default styles for the root/body (colors, typography, and spacing) and for other elements rather than specifically individual blocks. I'm more interested in configuring per-block settings in the long-term than per-block styles though.
I used spacing, block settings for some blocks, font families I can't think of the others, but i want to learn to use more features	Yes	I love, love, love using the json file! It made customizing so easy. I'm looking forward to working more with FSE and building more themes.	'- I customized blockquote, buttons, code, site-title, post-author, post-date, post-terms, post-title, featured image, navigation I customized because I needed styles to match my theme, and not just what WP provides. - other json sections I used were layout, border too
Switch of dropCap and customFontSize, enable custom units, enable wideSize	No		
Limiting color palettes on a per block basis and also adding in some custom properties to scaffold on top of	Yes	The theme.json file can get super big and a bit hard to peruse. My current project's theme.json is 350+ lines long. I've seen two solutions already surface to try and address this a bit: Justin Tadlock shared a webpack tool that merges multiple json files in to one, which is probably the most suitable for my needs. I'm just mentioning this as I think it'll quickly become a common gripe for engineers that use theme.json heavily and especially in a team/agency setting.	Button block gets the heaviest attention by far.
None	No		
Not much to date, but waiting on more to become available so I can use it more.	No		
Mainly custom definitions: filling gaps that aren't currently supported as presets, and duplicating layout definitions for contentSize and wideSize to expose these values.	No		
width setting styles per block, (e.g. cover, column, button, etc.) We are reviewing this part depending on the 'Global styles' updates.	Yes		columns-column, buttons-button, cover and adding more...
I wanted to use custom block definitions, but this doesn't work at the moment. Hopefully soon.	No	add_theme_support('align-wide'); seems to break when using the theme.json file :(	
None. This is the first time I've used it.	No	I've added theme.json to my Theme and added some colors to the palette array. e.g. --wp--preset--color--green. Then I've specified that links without any specific .has-text-color class should use this color as a default. a:not(.has-text-color) {color: var(--wp--preset--color--green);} I've added this to my CSS files enqueued in the frontend and in the editor. This works fine in the frontend, because WP loads the custom properties in the head of the page. But the editor doesn't have the --wp--preset--color--green custom property loaded in the editor, so the links are all black.	
n/a	No	These questions aren't very good/ relevant for someone who ticks "I have experimented" or "I have used" options.	
Haven't seen a reason to - only defined template areas - and typography. If you consider layout width as an additional setting - that is one thing which is very useful.	No	Looking forward to see how / if global settings can be overwritten on a per page/post basis - think it would be quite helpful.	
version, layout and turning most of the settings such as customColor to true	Yes	Yes. In order to support merging of theme.json from child themes with the parent's theme.json it needs to be clearly defined how to implement overrides. That's for applying changes, adding settings and deleting settings. There needs to be a validation routine that: - nicely reports syntax errors - indicates correct or incorrect semantics There should be some shortcut mechanism that allows the same styling to be applied to multiple blocks. And a tool to merge the Global Styles data with the original theme.json It should be easy to discover that theme.json can't be used for some styling, so you have to use style.css or another CSS file. In the documentation there need to be many more examples: - showing how to define elements. - explaining how theme.json overrides settings in block.json - and how to run internationalisation routines to create language files. Finally, RESPONSIVENESS.	So far: paragraphs, tables, buttons. This is to enable me to attempt to match the style of the theme I'm replacing. I've been fairly unsuccessful with styling navigation menus and tables, so had to resort to either not doing it or using style.css
enabling/disabling theme supports and block based settings.	Yes		Mainly buttons and general typography at the moment.
CorelDRAW	Yes		Yes
"dropCap": false,	Yes	I would love to see a better way of handling responsive text sizes. Currently we set the size in theme.json but then override it like this: body { --wp--preset--font-size--small: 1rem; --wp--preset--font-size--medium: 1.25rem; @include breakpoint(large) { --wp--preset--font-size--small: 1.125rem; --wp--preset--font-size--medium: 1.5rem; } } It works, but it sure feels hacky!	Columns, Buttons, Separators and Lists to give a more controlled style that matches brand guidelines.
typography, fontsizes, per block level settings	Yes	I think a single theme.json file will become very long and it might be good to introduce more structure to creating it. It seems a bit unstructured right now. Also I would like to be able to add color palettes to background and color block palettes separately.	Being able to customize all of them and my custom blocks is essential to me. I think font blocks like heading and paragraph are important to customize, the button block, preformatted block. But it comes down to the theme you are building I guess. Being able to customize all of them is great :)
Block configuration, utility's (spacing), custom for globals as a utility use case. Ultimately, it'd be interesting if we had the equivalent of a template.json that would populate both WP_POST_TYPE obj w/block template & a corresponding html file potentially....translating markup to arrays is messy / error prone so far. It'd be far easier to set parameters in a JSON file, which then another program simply translates those parameters into array's for WP_POST_TYPE & into mark-up for html templates.	Yes	I have to think more - but definitely have thoughts on this. Multisite functionality with theme.json in particular is a quite powerful use case. An "import" function, similar to wordpress importer to pass settings from one theme to another or one install to another or main theme to child theme - and consequently an exporter, would be super fantastic.	Individually styling a lot of blocks leads to a lot of customization when done explicitly in raw css - and potentially unnecessary bloat. Most of my customizations are set on the most used core blocks for layouting / synchronizing design aspects. Hard to think of the top of my head of the core set, I'll have to look at it again.
For settings, layout (contentSize and wideSize) For styles margins and padding.	Yes		I do customize pretty much all of them, as the default block styles rarely fit. I also remove every and all default block variations, and a lot of the patterns. Overall I'm not a fan of Core having "opinionated" styles, because that should be the theme's job. A theme without proper block editor support and opinionated defaults don't result in good outcomes. So I continue to be puzzled why the Gutenberg team thinks having more defaults are a good thing.
contentSize & wideSize, fontFamilies, setting default styling for blocks	Yes	I would like to be able to configure the size of blocks through the theme.json file. I didn't find a way to do this yet.	Currently only changing colors of texts in various blocks
I like layout as well.	No	I am struggling with wrapping my head around how to manage the theme.json when it becomes really large with a lot of customizations, I.e. when all the options are definable and work. It seems like it does have the possibility of becoming an experience like that of managing just 1 style.css, as opposed to using pre-processors and splitting it up, like sass. Even right now with the theme.json being small I have a hard time following my code. It would be helpful if there was some kind of commenting available, but there are no comments in .json, right? Don't get me wrong, I super love the idea of the json and defining variables and having 1 place to go to make customize a client theme.... just thinking ahead.	
I think this file could work as a design system declaration. So besides colors and typography, spacing comes next to mind.	No	We plan to use theme.json to *disable* all low-level controls like color pickers, padding controls, and even custom font sizes. We want to introduce some system-level controls similar to the Global Styles but couldn't wait until that part of the project is finished. I don't think that our end-users should have access to "a user interface for CSS rules" (e.g. Elementor), and I appreciate the flexibility that Gutenberg is preparing. Hope it helps! Best	
Sorry, haven't played with it enough to provide valid feedback for this.	No	CUSTOMIZER I am missing customizer in FSE themes and feel like global styles are not there yet. I used customizer not only to set colors, fonts and content widths (which is possible in theme.json too), but also for several other options specific to a theme, such as: - toggle options, for a sticky header toggle for example, - dropdown select and/or radio buttons, such as choosing which custom post type should display default page header/title (but most likely this will be able to replicate in FSE in the future with creating custom template for individual post types, but this was just an example of how I use custom radio buttons option currently), - maybe even a default/fallback post featured image option for posts lists (but this might be possible to set up in Query Loop block in the future?), - and similar... PAGE/POST TEMPLATES In my themes I also add functionality (via PHP) to enable page/post templates for all public post types automatically (using a "Template Post Type: public-post-types" magic word in a page template header). As far as I know I can not replicate this with theme.json currently. But again, I can imagine just creating a dedicated FSE template for specific situations when FSE is ready for such "abuse" :) Here is a current example of such functionality from one of my themes for illustration: - post template setup: https://github.com/webmandesign/michelle/blob/1.2.0/templates/no-intro.php - actual functionality enabling post template for public post types: https://github.com/webmandesign/michelle/blob/1.2.0/includes/Entry/Page_Template.php#L69 BODY CLASSES I'm not sure how to implement custom <body> tag classes as JSON is not enough for this, in my opinion. So I still have to use PHP for this. This brings me to asking whether there is currently some filter hook I can use to check parsed values from theme.json and/or from global styles? THEME.JSON WITH CLASSIC THEMES I also don't really see a benefit of using theme.json file with a classic theme when there is customizer still available. The theme.json might be useful to set up each individual block styles, but I am not using this feature and still can not find a usecase for it currently. OTHERS Even when testing global styles with TT1-Blocks theme, it did not pick up my custom global styles I set via WordPress admin (not via theme.json) on a website front-end. But maybe that was only my mistake somewhere during testing? Or this functionality is not implemented yet? (I was testing on WP5.8 with Gutenberg plugin enabled.)	
So far almost all my edits to theme.json were changes to templateParts and templates in order to load newly defined templates into the editor.	No	Support for the removal of core block CSS on a block by block basis to enable easier styling. This would be a flag system like serveBlockUnstyled and then an array of blocks that you want served unstyled. This would only work initially for core blocks, but block makers could provide support for it by serving their styles conditionally and checking if their block is named in the serveBlockUnstyled array. The name of the option could be changed... originally I wanted to call it blockBlockStyles.	