ODENCODE

Refactoring renderDocument with separation of concerns

□C v0.8.0 **A** claude-sonnet-4-20250514 Sep 14, 2025, 3:42 PM

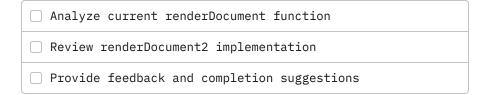
9	I want mind:	to	rewrite	the	renderDocument	function	with	these	goals	in
	Show mor	:e								

A\ ANTHROPIC

claude-sonnet-4-20250514

I'll analyze the current `renderDocument` function and your new `renderDocument2` approach to provide feedback and suggestions for completing the work.

□ CREATING PLAN



A\ ANTHROPIC

- GREP "renderDocument"
 - 25 matches >
- GREP "renderDocument2"
 - 2 matches >
- A\ ANTHROPIC

/25, 10	, 10:33 PM Refactoring re	nderDocument with separation of co
	claude-sonnet-4-20250514	
	READ lib/render.ts Show preview >	
Α\	\ ANTHROPIC	
	claude-sonnet-4-20250514	
	UPDATING PLAN	
	Analyze current renderDocument fund	tion
	☐ Review renderDocument2 implementati	on
	☐ Provide feedback and completion sug	gestions
Α\	N ANTHROPIC	
	claude-sonnet-4-20250514	
	READ lib/render.ts Show preview >	
Α\	∖ ANTHROPIC	
	claude-sonnet-4-20250514	
Pa	GREP "RenderingResults RenderingDa 11 matches >	ta createRenderingData
Α\	N ANTHROPIC	
	claude-sonnet-4-20250514	
P	READ lib/render.ts	
	Show preview >	
Α\	N ANTHROPIC	
	claude-sonnet-4-20250514	
	UPDATING PLAN	
	Review renderDocument2 implementati	on
	☐ Provide feedback and completion sug	gestions
	The state of the s	I I

✓ Analyze current renderDocument function

claude-sonnet-4-20250514

READ lib/render.ts

Show preview >

A\ ANTHROPIC

claude-sonnet-4-20250514

■ UPDATING PLAN

- Provide feedback and completion suggestions
- ✓ Analyze current renderDocument function
- ▼ Review renderDocument2 implementation

A\ ANTHROPIC

claude-sonnet-4-20250514

Analysis and Feedback

Your `renderDocument2` approach is excellent and addresses the key issues with the original function. Here are my observations:...

Show more

- ☐ Implement CSS renderFormat handling in renderDocument2
 ☐ Implement non-HTML asset copying in renderDocument2
- ☐ Add comprehensive error handling with structured error collection
- ☐ Create helper functions for CSS and asset processing
- ☐ Update return data structure

A\ ANTHROPIC

claude-sonnet-4-20250514

[∨] Kev

Key missing implementations:

1. **CSS processing branch** (lines 277-278) - needs to render CSS content and write to output...

Show more

UPDATTNG	

☐ Implement CSS renderFormat handling in renderDocument2
☐ Implement non-HTML asset copying in renderDocument2
Add comprehensive error handling with structured error collection
☐ Create helper functions for CSS and asset processing
Add performance calculation utilities
☐ Update return data structure
✓ Analyze current renderDocument function
▼ Review renderDocument2 implementation
✓ Provide feedback and completion suggestions

A\ ANTHROPIC

claude-sonnet-4-20250514

are you awake? there's been no response for awhile

A\ ANTHROPIC

claude-sonnet-4-20250514

Yes, I'm here! I provided a detailed analysis of your `renderDocument2` approach about a minute ago.

Your approach looks excellent - it addresses the key issues with clean...

Show more

- implement the CSS processing branch. Do this by adding a new function named renderCSSFile
- **A**\ ANTHROPIC

claude-sonnet-4-20250514

■ UPDATING PLAN

☐ Implement CSS renderFormat handling in renderDocument2

```
    Implement non-HTML asset copying in renderDocument2
    Add comprehensive error handling with structured error collection
    Create helper functions for CSS and asset processing
    Add performance calculation utilities
    Update return data structure
```

claude-sonnet-4-20250514

READ lib/render.ts
Show preview >

A\ ANTHROPIC

claude-sonnet-4-20250514

```
return dest;
    return dest;
}
                                              }
                                            + async function renderCSSFile(ret:
                                              RenderingData): Promise<void> {
                                                  try {
                                                       // Render the CSS content
                                                       ret.results.renderFirstStart =
                                              performance.now();
                                                       // Add necessary items to the
                                              metadata
                                              ret.renderFirstContext.metadata.config
                                              = ret.config;
                                               ret.renderFirstContext.metadata.partial
                                              = (fname, metadata) => {
                                                           return
                                              ret.config.akasha.partial(ret.config,
                                              fname, metadata);
                                                       };
                                              ret.renderFirstContext.metadata.partialS
                                               ync = (fname, metadata) => {
```

return

```
ret.config.akasha.partialSync(ret.config
                                                     , fname, metadata);
                                                             };
                                                     ret.renderFirstContext.metadata.akasha
                                                     = ret.config.akasha;
                                                     ret.renderFirstContext.metadata.plugin
                                                     = ret.config.plugin;
                                                             // Render the CSS content
                                                             ret.renderedFirst = await
                                                     ret.renderer.render(ret.renderFirstConte
                                                             ret.results.renderFirstEnd =
                                                     performance.now();
                                                             // Write the rendered CSS to
                                                    output
                                                             const renderDest =
                                                    path.join(ret.config.renderTo,
                                                     ret.docInfo.renderPath);
                                                             await
                                                     fsp.mkdir(path.dirname(renderDest), {
                                                     recursive: true });
                                                             await fsp.writeFile(renderDest,
                                                     ret.renderedFirst, 'utf-8');
                                                             ret.results.renderEnd =
                                                    performance.now();
                                                         } catch (error) {
                                                             ret.results.errors =
                                                    ret.results.errors || [];
                                                             ret.results.errors.push(error
                                                    instanceof Error ? error : new
                                                    Error(String(error)));
                                                             throw error;
                                                   +
                                                         }
                                                   + }
                                                   + /**
       * Attempt to rewrite renderDocument
                                                      * Attempt to rewrite renderDocument
      with cleaner code, and a
                                                     with cleaner code, and a
       * different method for collecting
                                                      * different method for collecting
      performance/timing data.
                                                     performance/timing data.
A\ ANTHRUPIC
```



Let me check the imports and context around this area:

READ lib/render.ts
Show preview >

A\ ANTHROPIC

claude-sonnet-4-20250514

Now let me add the function correctly:

```
}
                                               }
async function renderCSSFile(ret:
                                               async function renderCSSFile(ret:
RenderingData): Promise<void> {
                                               RenderingData): Promise<void> {
   try {
                                                   try {
                                                       ret.results.renderFirstStart =
                                               performance.now();
                                                       // Add necessary items to the
                                               metadata
                                               ret.renderFirstContext.metadata.config
                                               = ret.config;
                                               ret.renderFirstContext.metadata.partial
                                               = (fname, metadata) => {
                                                           return
                                               ret.config.akasha.partial(ret.config,
                                               fname, metadata);
                                             +
                                                       };
                                               ret.renderFirstContext.metadata.partialS
                                              ync = (fname, metadata) => {
                                                           return
                                               ret.config.akasha.partialSync(ret.config
                                               , fname, metadata);
                                                       };
                                             +
                                               ret.renderFirstContext.metadata.akasha
                                               = ret.config.akasha;
                                               ret.renderFirstContext.metadata.plugin
                                               = ret.config.plugin;
```

```
// Render the CSS content
                                                       // Render the CSS content
                                                       ret.renderedFirst = await
                                               ret.renderer.render(ret.renderFirstConte
                                                       ret.results.renderFirstEnd =
                                            +
                                               performance.now();
                                                       // Write the rendered CSS to
                                              output
                                                       const renderDest =
                                              path.join(ret.config.renderTo,
                                               ret.docInfo.renderPath);
                                                       await
                                               fsp.mkdir(path.dirname(renderDest), {
                                               recursive: true });
                                                       await fsp.writeFile(renderDest,
                                              ret.renderedFirst, 'utf-8');
                                                       ret.results.renderEnd =
                                              performance.now();
                                                  } catch (error) {
                                                       ret.results.errors =
                                              ret.results.errors || [];
                                                       ret.results.errors.push(error
                                              instanceof Error ? error : new
                                              Error(String(error)));
                                                       throw error;
                                                  }
                                            + }
                                            + async function renderCSSFile(ret:
                                               RenderingData): Promise<void> {
                                                   try {
                                                       // Render the CSS content
                                                       ret.results.renderFirstStart =
        ret.results.renderFirstStart =
performance.now();
                                               performance.now();
        // Add necessary items to the
                                                       // Add necessary items to the
metadata
                                               metadata
ret.renderFirstContext.metadata.config
                                               ret.renderFirstContext.metadata.config
= ret.config;
                                              = ret.config;
```

```
ERROR [239:16] Duplicate function implementation.

ERROR [271:16] Duplicate function implementation.
```

```
Let me check what's already there:
```

GREP "renderCSSFile"
2 matches >

A\ ANTHROPIC

claude-sonnet-4-20250514

It looks like there are already duplicate functions. Let me read around those lines:

READ lib/render.ts
Show preview >

A\ ANTHROPIC

claude-sonnet-4-20250514

- I see there are duplicate functions. Let me remove the second duplicate and keep the first one:
- READ lib/render.ts
 Show preview >
- **A**\ ANTHROPIC

claude-sonnet-4-20250514

Let me remove the duplicate function:

```
throw error;
}

throw error;
}

async function renderCSSFile(ret:
RenderingData): Promise<void> {
    try {
        // Render the CSS content
        ret.results.renderFirstStart =
    performance.now();
}
```

```
// Add necessary items to the
metadata
ret.renderFirstContext.metadata.config
= ret.config;
ret.renderFirstContext.metadata.partial
= (fname, metadata) => {
            return
ret.config.akasha.partial(ret.config,
fname, metadata);
        };
ret.renderFirstContext.metadata.partialS
ync = (fname, metadata) => {
            return
ret.config.akasha.partialSync(ret.config
, fname, metadata);
       };
ret.renderFirstContext.metadata.akasha
= ret.config.akasha;
ret.renderFirstContext.metadata.plugin
= ret.config.plugin;
        // Render the CSS content
        ret.renderedFirst = await
ret.renderer.render(ret.renderFirstConte
xt);
        ret.results.renderFirstEnd =
performance.now();
        // Write the rendered CSS to
output
        const renderDest =
path.join(ret.config.renderTo,
ret.docInfo.renderPath);
        await
fsp.mkdir(path.dirname(renderDest), {
recursive: true });
        await fsp.writeFile(renderDest,
ret.renderedFirst, 'utf-8');
        ret.results.renderEnd =
performance.now();
    } catch (error) {
        ret.results.errors =
ret.results.errors || [];
```

- Now let me update the renderDocument2 function to use the renderCSSFile function. First let me find where the CSS branch is:
- READ lib/render.ts
 Show preview >
- ANTHROPIC claude-sonnet-4-20250514
- Let me find the CSS branch in renderDocument2:
- READ lib/render.ts
 Show preview >
- ANTHROPIC claude-sonnet-4-20250514
- Perfect! Now I'll update the CSS branch to call the renderCSSFile function:
- EDIT lib/render.ts

```
const ret: RenderingData =
    createRenderingData(config, docInfo);

// Peel off to mode-specific functions

const ret: RenderingData =
    createRenderingData(config, docInfo);

// Peel off to mode-specific functions
// Peel off to mode-specific functions
```

```
(ret?.renderer?.renderFormat.toString()
                                                     (ret?.renderer?.renderFormat.toString()
      === 'CSS') {
                                                     === 'CSS') {
         // render for CSS
                                                         await renderCSSFile(ret);
                                                         return ret.results;
      } else if (!ret.renderer
                                                     } else if (!ret.renderer
      (ret.renderer.renderFormat.toString()
                                                     (ret.renderer.renderFormat.toString()
      !== 'HTML')
                                                     !== 'HTML')
      ) {
                                                     ) {
                                                         // Copy asset
          // Copy asset
AN ANT
```

claude-sonnet-4-20250514

Now let me check if the renderCSSFile function needs to be correctly implemented to match the pattern. Let me check how the renderer.renderFormat is called:

GREP "renderFormat.*toString"
2 matches >

A\ ANTHROPIC

claude-sonnet-4-20250514

- Let me check how renderFormat is used in other places:
- GREP "renderFormat"

 10 matches >
- A\ ANTHROPIC

claude-sonnet-4-20250514

I see the issue. The `renderFormat` should be called as a function, not accessed as a property with `.toString()`. Let me fix this:

```
// Create the master object to hold all
data
const ret: RenderingData =
createRenderingData(config, docInfo);

// Peel off to mode-specific functions

// Create the master object to hold all
data
const ret: RenderingData =
createRenderingData(config, docInfo);

// Peel off to mode-specific functions
// Peel off to mode-specific functions
```

```
- if
                                               + if
 (ret?.renderer?.renderFormat.toString()
                                                 (ret?.renderer?.renderFormat(ret.renderF
 === 'CSS') {
                                                 irstContext) === 'CSS') {
     await renderCSSFile(ret);
                                                     await renderCSSFile(ret);
     return ret.results;
                                                     return ret.results;
 } else if (!ret.renderer
                                                 } else if (!ret.renderer
                                                 (ret.renderer.renderFormat(ret.renderFir
 (ret.renderer.renderFormat.toString()
 !== 'HTML')
                                                 stContext) !== 'HTML')
 ) {
                                                 ) {
                                                     // Copy asset
     // Copy asset
 }
                                                 }
```

claude-sonnet-4-20250514

■ UPDATING PLAN

- ☐ Implement non-HTML asset copying in renderDocument2
 ☐ Add comprehensive error handling with structured error collection
 ☐ Create helper functions for CSS and asset processing
 ☐ Add performance calculation utilities
 ☐ Update return data structure
 ☑ Implement CSS renderFormat handling in renderDocument2
- A\ ANTHROPIC

- Perfect! I've successfully implemented the CSS processing branch in `renderDocument2` by:

 1. Created a `renderCSSFile` function that:...

 Show more
- in renderCSSFile, two issues. There is no metadata to consider, so therefore setting up the metadata is not necessary. Second, the error handler should return the "ret" object rather than throw an error.
- A\ ANTHROPIC

claude-sonnet-4-20250514

☐ Fix renderCSSFile metadata setup and error handling
☐ Implement non-HTML asset copying in renderDocument2
Add comprehensive error handling with structured error collection
☐ Create helper functions for CSS and asset processing
Add performance calculation utilities
☐ Update return data structure

A\ ANTHROPIC

claude-sonnet-4-20250514

```
return dest;
 }
- async function renderCSSFile(ret:
 RenderingData): Promise<void> {
          ret.results.renderFirstStart =
 performance.now();
          // Add necessary items to the
 metadata
 ret.renderFirstContext.metadata.config
 = ret.config;
 ret.renderFirstContext.metadata.partial
 = (fname, metadata) => {
             return
 ret.config.akasha.partial(ret.config,
 fname, metadata);
          };
 ret.renderFirstContext.metadata.partialS
 ync = (fname, metadata) => {
              return
 ret.config.akasha.partialSync(ret.config
 , fname, metadata);
```

```
}
      return dest;
  }
+ async function renderCSSFile(ret:
  RenderingData): Promise<RenderingData> {
          ret.results.renderFirstStart =
  performance.now();
```

```
ret.renderFirstContext.metadata.akasha
= ret.config.akasha;
ret.renderFirstContext.metadata.plugin
= ret.config.plugin;
                                                       // Render the CSS content
        // Render the CSS content
                                                       ret.renderedFirst = await
        ret.renderedFirst = await
ret.renderer.render(ret.renderFirstConte
                                               ret.renderer.render(ret.renderFirstConte
        ret.results.renderFirstEnd =
                                                       ret.results.renderFirstEnd =
performance.now();
                                               performance.now();
        ret.results.renderEnd =
                                                       ret.results.renderEnd =
performance.now();
                                               performance.now();
    } catch (error) {
                                                   } catch (error) {
        ret.results.errors =
                                                       ret.results.errors =
ret.results.errors || [];
                                               ret.results.errors || [];
        ret.results.errors.push(error
                                                       ret.results.errors.push(error
instanceof Error ? error : new
                                               instanceof Error ? error : new
Error(String(error)));
                                               Error(String(error)));
        throw error;
    }
                                                   }
                                                   return ret;
}
                                              }
```

ANTHROPIC

claude-sonnet-4-20250514

Now I need to update the call to `renderCSSFile` in `renderDocument2` to handle the returned object:

```
const ret: RenderingData =
  createRenderingData(config, docInfo);

// Peel off to mode-specific functions
  if
  (ret?.renderer?.renderFormat(ret.renderF
  irstContext) === 'CSS') {
    await renderCSSFile(ret);

    const ret: RenderingData =
    createRenderingData(config, docInfo);

    // Peel off to mode-specific functions
    if
    (ret?.renderer?.renderFormat(ret.renderF
    irstContext) === 'CSS') {
        const cssResult = await
        renderCSSFile(ret);
    }
}
```

claude-sonnet-4-20250514

■ UPDATING PLAN

- ☐ Implement non-HTML asset copying in renderDocument2
 ☐ Add comprehensive error handling with structured error collection
 ☐ Create helper functions for CSS and asset processing
 ☐ Add performance calculation utilities
 ☐ Update return data structure
 ✔ Fix renderCSSFile metadata setup and error handling
- A\ ANTHROPIC

claude-sonnet-4-20250514

Perfect! I've fixed both issues in the `renderCSSFile` function:

1. Removed unnecessary metadata setup - Since CSS files don't need access to partials, config, akasha, or plugin objects, I removed all that...

Show more

Now implement the branch for copying non-HTML-non-CSS files to the output directory. The function copyAssetToOutput does this same task, but it doesn't know about the new RenderingData and RenderingResult ...

Show more

A\ ANTHROPIC

claude-sonnet-4-20250514

■ UPDATING PLAN

Implement non-HTML asset copying in renderDocument2

Add comprehensive error handling with structured error collection
☐ Create helper functions for CSS and asset processing
☐ Add performance calculation utilities
☐ Update return data structure

claude-sonnet-4-20250514

Let me first look at the existing `copyAssetToOutput` function to understand what it does:

READ lib/render.ts

Show preview >

A\ ANTHROPIC

claude-sonnet-4-20250514

Now I'll create a new function `copyAssetFile` that follows the same pattern as `renderCSSFile`:

```
}
                                                   }
   return ret;
                                                   return ret;
}
                                               }
                                             + async function copyAssetFile(ret:
                                               RenderingData): Promise<RenderingData> {
                                                   try {
                                                       ret.results.renderFirstStart =
                                               performance.now();
                                                       // Copy the asset file to
                                               output directory
                                                       const renderDest =
                                               path.join(ret.config.renderTo,
                                               ret.docInfo.renderPath);
                                                       await
                                               fsp.mkdir(path.dirname(renderDest), {
                                               recursive: true });
```

```
await
                                              fsp.copyFile(ret.docInfo.fspath,
                                              renderDest);
                                                      ret.results.renderEnd =
                                              performance.now();
                                                  } catch (error) {
                                                      ret.results.errors =
                                              ret.results.errors || [];
                                                      ret.results.errors.push(error
                                              instanceof Error ? error : new
                                              Error(String(error)));
                                                  }
                                                 return ret;
                                            + }
                                            + /**
/**
* Attempt to rewrite renderDocument
                                               * Attempt to rewrite renderDocument
with cleaner code, and a
                                              with cleaner code, and a
* different method for collecting
                                               * different method for collecting
performance/timing data.
                                              performance/timing data.
```

claude-sonnet-4-20250514

Now I need to update the asset copying branch in `renderDocument2` to use the new function:

EDIT lib/render.ts

```
return cssResult.results;
                                                   return cssResult.results;
 } else if (!ret.renderer
                                               } else if (!ret.renderer
 (ret.renderer.renderFormat(ret.renderFir
                                               (ret.renderer.renderFormat(ret.renderFir
 stContext) !== 'HTML')
                                               stContext) !== 'HTML')
                                               ) {
 ) {
// Copy asset
                                             + const assetResult = await
                                               copyAssetFile(ret);
                                                  return assetResult.results;
 }
                                               }
 // Otherwise it is HTML
                                               // Otherwise it is HTML
 // This is where we render the content,
                                               // This is where we render the content,
 then render that
                                               then render that
```

A\ ANTHROPIC

claude-sonnet-4-20250514

Add comprehensive error handling with structured error collection
☐ Add performance calculation utilities
☐ Update return data structure
✓ Implement non-HTML asset copying in renderDocument2
✓ Create helper functions for CSS and asset processing

AN ANTHROPIC

claude-sonnet-4-20250514

Perfect! I've successfully implemented the asset copying branch by:

1. Created a `copyAssetFile` function that:

• Uses `performance.now()` for precise timing like the CSS function...

Show more

- What remains to be done in renderDocument2? Capture errors and append to ret.errors? Anything else?
- A\ ANTHROPIC

claude-sonnet-4-20250514

READ lib/render.ts
Show preview >

A\ ANTHROPIC

claude-sonnet-4-20250514

READ lib/render.ts
Show preview >

A\ ANTHROPIC

❖

Looking at the current `renderDocument2` implementation, here's what remains to be done:

CREAT	TTNC	DI	A N I
 CIVEA	LTING		\neg

- □ Wrap HTML rendering sections in try-catch blocks
 □ Replace throw statements with error collection
 □ Add elapsed time calculations to results
 □ Wrap file write operations in try-catch
- A\ ANTHROPIC

claude-sonnet-4-20250514

What remains to be done in `renderDocument2`:

High Priority:

1. Error handling for HTML rendering sections - Currently the function...

Show more

- Yes, implement those fixes
- A\ ANTHROPIC

claude-sonnet-4-20250514

■ UPDATING PLAN

- Wrap HTML rendering sections in try-catch blocks
 Replace throw statements with error collection
 Add elapsed time calculations to results
 Wrap file write operations in try-catch
- AN ANTHROPIC

claude-sonnet-4-20250514

EDIT lib/render.ts

// First Render // First Render

```
ret.results.renderFirstStart =
 ret.results.renderFirstStart =
 performance.now();
                                                 performance.now();
- // Add necessary items to the metadata
                                              + try {
- ret.renderFirstContext.metadata.config
                                                     // Add necessary items to the
                                                metadata
 = config;
ret.renderFirstContext.metadata.partial
 = doPartial;
                                                 ret.renderFirstContext.metadata.config
                                                 = confia:
- ret.renderFirstContext.metadata.partialS
                                                 ret.renderFirstContext.metadata.partial
 ync = doPartialSync;
                                                 = doPartial:
- ret.renderFirstContext.metadata.akasha
                                                 ret.renderFirstContext.metadata.partialS
 = config.akasha;
                                                 ync = doPartialSync;
- ret.renderFirstContext.metadata.plugin
                                              +
 = config.plugin;
                                                 ret.renderFirstContext.metadata.akasha
                                                 = config.akasha;
                                                 ret.renderFirstContext.metadata.plugin
                                                 = config.plugin;
- // Render the primary content
                                                     // Render the primary content
- ret.renderedFirst = await
                                                     ret.renderedFirst = await
 ret.renderer.render(ret.renderFirstConte
                                                 ret.renderer.render(ret.renderFirstConte
 xt);
                                              + } catch (error) {
                                                     ret.results.errors =
                                                 ret.results.errors || [];
                                                     ret.results.errors.push(error
                                                 instanceof Error ? error : new
                                                Error(String(error)));
                                                     // Use empty string as fallback if
                                                 rendering fails
                                                     ret.renderedFirst = '';
                                              + }
 ret.results.renderFirstEnd =
                                                 ret.results.renderFirstEnd =
 performance.now();
                                                 performance.now();
                                                 // END First Render
 // END First Render
```

ANTHROPIC

claude-sonnet-4-20250514

■ UPDATING PLAN

■ Replace throw statements with error collection

- Add elapsed time calculations to results ☐ Wrap file write operations in try-catch ✓ Wrap HTML rendering sections in try-catch blocks
- A\ ANTHROPIC

claude-sonnet-4-20250514

Now let me fix the layout render error handling:

```
// Layout Render
ret.results.renderLayoutStart =
performance.now();
if (ret?.docInfo?.metadata?.layout) {
    const layouts =
config.akasha.filecache.layoutsCache;
    // await layouts.isReady();
                                             +
```

```
// Layout Render
  ret.results.renderLayoutStart =
  performance.now();
 if (ret?.docInfo?.metadata?.layout) {
     try {
          const layouts =
 config.akasha.filecache.layoutsCache;
          // await layouts.isReady();
          let found = await
 layouts.find(ret.docInfo.metadata.layout
  );
          if (!found) {
              const error = new Error(`No
  layout found in
  ${util.inspect(ret.config.layoutDirs)}
  for ${ret?.docInfo?.metadata?.layout}
  in file ${ret.docInfo.vpath}`);
              ret.results.errors =
  ret.results.errors || [];
 ret.results.errors.push(error);
              // Skip layout rendering,
 use first render result
              ret.renderedLayout =
  ret.renderedFirst;
          } else {
              const renderer =
 config.findRendererPath(
 ret.docInfo.metadata.layout
+
              );
```

```
let found = await
                                                           ret.renderLayoutContext =
layouts.find(ret.docInfo.metadata.layout
                                               <RenderingContext>{
);
   if (!found) {
                                                               fspath:
                                               ret.docInfo.metadata.layout,
                                                               content:
        throw new Error(`No layout
                                               found.docContent,
found in
${util.inspect(ret.config.layoutDirs)}
for ${ret?.docInfo?.metadata?.layout}
in file ${ret.docInfo.vpath}`);
    }
                                                               body: found.docBody,
                                                               metadata: {}
                                             +
                                                           };
    const renderer =
config.findRendererPath(
                                               ret.renderLayoutContext.metadata
        ret.docInfo.metadata.layout
                                                               = copyProperties(
                                             +
    );
                                               ret.renderLayoutContext.metadata,
                                             +
                                                                    found.metadata,
                                                                    false
                                             +
                                                               );
                                               ret.renderLayoutContext.metadata
                                                               = copyProperties(
                                             +
                                               ret.renderLayoutContext.metadata,
                                               ret.docInfo.metadata,
                                                                    true
                                                               );
    ret.renderLayoutContext =
<RenderingContext>{
                                               ret.renderLayoutContext.metadata.content
                                                = ret.renderedFirst;
        fspath:
ret.docInfo.metadata.layout,
        content: found.docContent,
        body: found.docBody,
        metadata: {}
    };
    ret.renderLayoutContext.metadata
                                               ret.renderLayoutContext.metadata.config
                                               = config;
        = copyProperties(
                                               ret.renderLayoutContext.metadata.partial
                                                = doPartial;
```

```
ret.renderLayoutContext.metadata.partial
ret.renderLayoutContext.metadata,
                                               Sync = doPartialSync;
            found.metadata,
                                               ret.renderLayoutContext.metadata.akasha
                                               = config.akasha;
            false
                                               ret.renderLayoutContext.metadata.plugin
                                               = config.plugin;
        );
    ret.renderLayoutContext.metadata
        = copyProperties(
ret.renderLayoutContext.metadata,
            ret.docInfo.metadata,
            true
        );
                                                           // Render the layout content
\verb"ret.renderLayoutContext.metadata.content"
= ret.renderedFirst;
                                                            ret.renderedLayout = await
                                               renderer.render(ret.renderLayoutContext)
                                                       }
                                             +
ret.renderLayoutContext.metadata.config
= config;
ret.renderLayoutContext.metadata.partial
= doPartial;
ret.renderLayoutContext.metadata.partial
Sync = doPartialSync;
ret.renderLayoutContext.metadata.akasha
= config.akasha;
ret.renderLayoutContext.metadata.plugin
= config.plugin;
    try {
        // Render the primary content
        ret.renderedLayout
                = await
renderer.render(ret.renderLayoutContext)
    } catch (e) {
                                                   } catch (e) {
```

```
let ee = new Error(`Error
                                                             const error = new Error(`Error
                                                     rendering ${docInfo.vpath} with
      rendering ${docInfo.vpath} with
      ${docInfo?.metadata?.layout} ${e.stack
                                                     ${docInfo?.metadata?.layout} ${e.stack
                                                     ? e.stack : e}`);
      ? e.stack : e}`);
                                                             ret.results.errors =
              console.error(ee);
                                                     ret.results.errors || [];
                                                             ret.results.errors.push(error);
              throw ee;
                                                             // Use first render result as
                                                     fallback
                                                             ret.renderedLayout =
                                                     ret.renderedFirst;
          }
                                                         }
                                                     }
ANT
   cl;ret.results.renderLayoutEnd =
                                                     ret.results.renderLayoutEnd =
      performance.now();
                                                     performance.now();
```

Now let me fix the Mahabhuta error handling:

```
ret.docInfo.metadata,
                                                         ret.docInfo.metadata,
                                                         false
         false
      );
                                                     );
- if
 (ret.docInfo?.metadata?.config?.mahabhut
 aConfig) {
 mahabhuta.config(ret.docInfo?.metadata?.
 config?.mahabhutaConfig);
- }
- try {
                                                 try {
                                              + if
                                                 (ret.docInfo?.metadata?.config?.mahabhut
                                                 aConfig) {
                                                mahabhuta.config(ret.docInfo?.metadata?.
                                                config?.mahabhutaConfig);
                                              +
      ret.renderedMaha = await
                                                     ret.renderedMaha = await
 mahabhuta.processAsync(
                                                 mahabhuta.processAsync(
          ret.renderMahaContext.content,
                                                         ret.renderMahaContext.content,
 ret.renderMahaContext.metadata,
                                                 ret.renderMahaContext.metadata,
          ret.config.mahafuncs
                                                         ret.config.mahafuncs
      );
                                                     );
 } catch (e2) {
                                                 } catch (e2) {
```

```
let eee = new Error(`Error with
                                                  const error = new Error(`Error with
Mahabhuta ${ret.docInfo.vpath} with
                                               Mahabhuta ${ret.docInfo.vpath} with
${ret.docInfo?.metadata?.layout}
                                               ${ret.docInfo?.metadata?.layout}
${e2.stack ? e2.stack : e2}`);
                                               ${e2.stack ? e2.stack : e2}`);
                                                   ret.results.errors =
   console.error(eee);
                                               ret.results.errors || [];
                                                   ret.results.errors.push(error);
   throw eee;
                                                  // Use layout result or first
                                               render as fallback
                                                   ret.renderedMaha =
                                               ret.renderMahaContext.content;
}
ret.results.renderMahaEnd =
                                               ret.results.renderMahaEnd =
performance.now();
                                               performance.now();
// END Mahabhuta
                                               // END Mahabhuta
```

ANTHROPIC

claude-sonnet-4-20250514

■ UPDATING PLAN

- Wrap file write operations in try-catch
 Add elapsed time calculations to results
 ✓ Wrap HTML rendering sections in try-catch blocks
 ✓ Replace throw statements with error collection
- **ANTHROPIC**

claude-sonnet-4-20250514

Now let me add error handling for the file write operations:

```
recursive: true
                                                 fsp.mkdir(path.dirname(renderDest), {
                                                         recursive: true
- });
- await fsp.writeFile(renderDest,
                                                     });
                                                     await fsp.writeFile(renderDest,
                      ret.renderedMaha,
  'utf-8');
                                                 ret.renderedMaha, 'utf-8');
                                               + } catch (error) {
                                                     ret.results.errors =
                                                 ret.results.errors || [];
                                                     ret.results.errors.push(error
                                                 instanceof Error ? error : new
                                                Error(String(error)));
                                               + }
 ret.results.renderEnd =
                                                 ret.results.renderEnd =
 performance.now();
                                                 performance.now();
 return ret.results;
                                                 return ret.results;
```

claude-sonnet-4-20250514

■ UPDATING PLAN

- Add elapsed time calculations to results

 ✓ Wrap HTML rendering sections in try-catch blocks

 ✓ Replace throw statements with error collection

 ✓ Wrap file write operations in try-catch
- A\ ANTHROPIC

claude-sonnet-4-20250514

Now let me add elapsed time calculations. First, I need to check what fields are available in the RenderingResults type:

READ lib/render.ts

Show preview >

A\ ANTHROPIC



I need to add elapsed time fields to the RenderingResults type first:

EDIT lib/render.ts

```
renderMahaStart?: number;
                                                   renderMahaStart?: number;
    renderMahaEnd?: number;
                                                   renderMahaEnd?: number;
                                                   // Elapsed time calculations
                                                   renderFirstElapsed?: number;
                                             +
                                                   renderLayoutElapsed?: number;
                                                   renderMahaElapsed?: number;
                                                   renderTotalElapsed?: number;
                                                   errors?: Array<Error>;
   errors?: Array<Error>;
};
                                               };
// Collect all required data in an
                                               // Collect all required data in an
instance of this object.
                                               instance of this object.
```

A\ ANTHROPIC

claude-sonnet-4-20250514

❖

Now let me add the elapsed time calculations at the end of renderDocument2:

```
}
                                                  }
    ret.results.renderEnd =
                                                   ret.results.renderEnd =
performance.now();
                                               performance.now();
                                                  // Calculate elapsed times
                                                   if (ret.results.renderFirstStart &&
                                               ret.results.renderFirstEnd) {
                                                       ret.results.renderFirstElapsed
                                               = ret.results.renderFirstEnd -
                                               ret.results.renderFirstStart;
                                                  if (ret.results.renderLayoutStart
                                               && ret.results.renderLayoutEnd) {
                                                       ret.results.renderLayoutElapsed
                                               = ret.results.renderLayoutEnd -
                                               ret.results.renderLayoutStart;
                                                  }
```

```
+ if (ret.results.renderMahaStart &&
    ret.results.renderMahaEnd) {
+        ret.results.renderMahaElapsed =
        ret.results.renderMahaElapsed =
        ret.results.renderMahaStart;
+    }
+    if (ret.results.renderStart &&
    ret.results.renderEnd) {
+        ret.results.renderTotalElapsed
=        ret.results.renderEnd -
        ret.results.renderStart;
+    }

return ret.results;
}

/**
```

claude-sonnet-4-20250514

- ✓ Wrap HTML rendering sections in try-catch blocks
 ✓ Replace throw statements with error collection
 ✓ Add elapsed time calculations to results
 ✓ Wrap file write operations in try-catch
- A\ ANTHROPIC

claude-sonnet-4-20250514

Perfect! I've successfully implemented all the requested fixes to `renderDocument2`:

Completed Improvements:...

Show more

This looks good.

The next step is to create a render2 function by rewriting the render…

Show more

A\ ANTHROPIC

	claude-sonnet-4-20250514		
	CREATING PLAN		
	☐ Analyze existing render function structure		
	☐ Create render2 function with fastq integration		
	☐ Handle RenderingResults aggregation		
	☐ Maintain concurrency control with config.concurrency		
Α\	ANTURORIC		
Α\	ANTHROPIC claude-sonnet-4-20250514		
	Claude-Sonnet-4-20250514		
	READ lib/render.ts		
	Show preview >		
Α\	ANTHROPIC		
	claude-sonnet-4-20250514		
B	READ lib/render.ts		
	Show preview >		
Δ\	ANTHROPIC		
	claude-sonnet-4-20250514		
	UPDATING PLAN		
	☐ Create render2 function with fastq integration		
	☐ Handle RenderingResults aggregation		
	☐ Maintain concurrency control with config.concurrency		
	✓ Analyze existing render function structure		
Δ\	ANTURORIC		
Α\	ANTHROPIC claude-sonnet-4-20250514		
	C1auue-S0111161-4-20200014		
♦	Now I'll create the `render2` function by adapting the	existing	`render`
	function to use `renderDocument2` and handle `Rendering		

https://opencode.ai/s/bdruDOwS

EDIT lib/render.ts

// 5. return results

// 5. return results

```
return results;
};
```

```
return results;
  };
+ /**
+ * Render all the documents in a site
 using renderDocument2,
+ * limiting the number of simultaneous
  rendering tasks
+ * to the number in config.concurrency.
+ * Returns structured RenderingResults
  data instead of text strings.
+ * @param config
+ * @returns Array of RenderingResults
  with performance and error data
+ */
+ export async function render2(config):
  Promise<Array<{ result?:</pre>
  RenderingResults; error?: any; }>> {
      const documents =
  <DocumentsCache>config.akasha.filecache.
  documentsCache;
      // await documents.isReady();
      // console.log('CALLING
 config.hookBeforeSiteRendered');
      await
 config.hookBeforeSiteRendered();
      // 1. Gather list of files from
  RenderFileCache
      const filez = await
 documents.paths();
      // console.log(`render2 filez
  ${filez.length}`);
      // 2. Exclude any that we want to
  ignore
      const filez2 = [] as Array<{</pre>
          config: Configuration,
          info: Document
+
      }>;
      for (let entry of filez) {
+
          let include = true;
+
          // console.log(entry);
+
          let stats;
          try {
              stats = await
  fsp.stat(entry.fspath);
```

```
} catch (err) { stats =
 undefined; }
         if (!entry) include = false;
          else if (!stats ||
 stats.isDirectory()) include = false;
          // This should arise using an
 ignore clause
         // else if
 (path.basename(entry.vpath) ===
  '.DS_Store') include = false;
          // else if
 (path.basename(entry.vpath) ===
  '.placeholder') include = false;
         if (include) {
              // The queue is an array of
 tuples containing the
              // config object and the
 path string
              filez2.push({
                  config: config,
+
                  info: await
 documents.find(entry.vpath)
              });
          }
      }
     // console.log(`render2 filez2
 after ignore ${filez2.length}`);
      // 3. Make a fastq to process using
 renderDocument2,
     //
           pushing results to the
 results array
     // This sets up the queue processor
     // The concurrency setting lets us
 process documents
     // in parallel while limiting total
 impact.
     const queue: queueAsPromised<{</pre>
         config: Configuration,
         info: Document
     }> = fastq.promise(
          // This function is invoked for
 each entry in the
         // queue. It handles rendering
 the queue
          // The queue has config objects
 and path strings
          // which is exactly what's
  required by
```

```
// renderDocument2
          async function
 renderDocument2InQueue(entry)
              : Promise<{ result?:</pre>
 RenderingResults; error?: any; }>
          {
+
 console.log(`renderDocument2InQueue
  ${entry.info.vpath}`);
              try {
                  let result = await
  renderDocument2(
                      entry.config,
 entry.info
                  );
                  // console.log(`DONE
  renderDocument2InQueue
  ${entry.info.vpath}`);
                  return { result };
              } catch (error) {
+
                  // console.log(`ERROR
  renderDocument2InQueue
  ${entry.info.vpath}`, error.stack);
                  return { error };
+
              }
          },
          config.concurrency);
      // queue.push returns a Promise
  that's fulfilled when
      // the task finishes.
      // Hence waitFor is an array of
 Promises.
      const waitFor = [];
      for (let entry of filez2) {
          waitFor.push(queue.push(entry));
+
+
      // This automatically waits for all
 those
      // Promises to resolve, while
  making the results
      // array contain results.
      const results: Array<{ result?:</pre>
  RenderingResults; error?: any; }> = [];
      for (let result of waitFor) {
          results.push(await result);
+
+
      // 4. Invoke hookSiteRendered
+
      try {
+
```

```
+ // console.log('Invoking hookSiteRendered');
+ await config.hookSiteRendered();
+ } catch (e) {
+ console.error(e.stack);
+ throw new
Error(`hookSiteRendered failed because ${e}`);
+ }
+ // 5. return results
+ return results;
+ };
```


- ✓ Analyze existing render function structure
 ✓ Create render2 function with fastq integration
 ✓ Handle RenderingResults aggregation
 ✓ Maintain concurrency control with config.concurrency
- A\ ANTHROPIC

claude-sonnet-4-20250514

Perfect! I've successfully created the `render2` function with the following improvements:

Key Features of `render2`:

- 1. Uses `renderDocument2` for Structured Data
 - Calls `renderDocument2` instead of `renderDocument`
- Returns `RenderingResults` objects with performance metrics and error collections
- Provides structured data instead of text strings for better analysis
- 2. Maintains FastQ Concurrency Control
- Uses the same `fastq.promise` pattern as the original `render` function
- Respects `config.concurrency` setting for parallel processing control
- Handles queue processing with proper error isolation

3. Enhanced Error Handling

- Each document render is wrapped in try-catch within the queue processor
- Errors are captured without stopping the entire render process

• Returns both successful results and errors in a structured format

4. Proper Type Safety

- Explicitly typed return value: `Promise<Array<{ result?: RenderingResults; error?: any; }>>`
- Queue processor function properly typed to handle `RenderingResults`
- Maintains type consistency throughout the processing pipeline

5. Same Processing Flow

- Follows identical steps: gather files → filter → process in parallel → invoke hooks → return results
- Maintains compatibility with existing configuration and hooks
- Preserves all filtering and validation logic from the original

Benefits:

- Data-driven analysis: Get precise timing data for each render phase
- Error resilience: Individual file errors don't halt the entire site render
- **Performance insights**: Aggregate timing data across all files to identify bottlenecks
- Scalable concurrency: Same proven parallel processing with better data structure

The `render2` function is now ready to provide the structured performance and error data you wanted while maintaining the same reliable concurrency control as the original!

```
Sep 14, 2025, 5:01 PM
```

Connected, waiting for messages...

Cost \$0.00 Input Tokens 139 Output Tokens 18405 Reasoning Tokens -