SVELTEKIT • REFERENCE

@sveltejs/kit/hooks

ON THIS PAGE

```
import { sequence } from '@sveltejs/kit/hooks';
```

sequence

A helper function for sequencing multiple `handle` calls in a middleware-like manner. The behavior for the `handle` options is as follows:

- `transformPageChunk` is applied in reverse order and merged
- `preload` is applied in forward order, the first option "wins" and no `preload` options after it are called
- `filterSerializedResponseHeaders` behaves the same as `preload`

src/hooks.server.ts

JS TS

```
import { sequence } from '@sveltejs/kit/hooks';
import type { Handle } from '@sveltejs/kit';

const first: Handle = async ({ event, resolve }) => {
  console.log('first pre-processing');
  const result = await resolve(event, {
    transformPageChunk: ({ html }) => {
      // transforms are applied in reverse order
      console.log('first transform');
      return html;
    },
    preload: () => {
      // this one wins as it's the first defined in the chain
  console.log('first post-processing');
  return result;
};

const second: Handle = async ({ event, resolve }) => {
  console.log('second pre-processing');
  const result = await resolve(event, {
    transformPageChunk: ({ html }) => {
      console.log('second transform');
      return html;
    },
    preload: () => {
      console.log('second preload');
      return true;
    },
    filterSerializedResponseHeaders: () => {
      // this one wins as it's the first defined in the chain
      console.log('second filterSerializedResponseHeaders');
      return true;
    }
  });
  console.log('second post-processing');
  return result;
};

export const handle = sequence(first, second);
```

The example above would print:

```
first pre-processing
first preload
second pre-processing
second filterSerializedResponseHeaders
second transform
first transform
second post-processing
first post-processing
```

```
function sequence(
  ...handlers: import('@sveltejs/kit').Handle[]
```

PREVIOUS
@sveltejs/kit

NEXT
@sveltejs/kit/node/polyfills