Rendering Modes & API Routes

Default Rendering Mode

```
// /astro.config.mjs
import { defineConfig } from 'astro/config';

export default defineConfig({});
```

Projekt bauen

```
npm run build
```

Ergebnis: Statische Seite
pre-rendered
Auch bekannt als SSG - Static Site Generation

`output` Setting

```
// /astro.config.mjs
import { defineConfig } from 'astro/config';

export default defineConfig({
  output: 'static', // default
});
```

Gleiches Ergebnis: Statische Seite


On-demand Rendering

`output` = `server`

```
// /astro.config.mjs
import { defineConfig } from 'astro/config';

export default defineConfig({
  output: 'server',
});
```

Nützlicher Default für sehr dynamische Seiten, bei denen die meisten Inhalte (oder sogar alle) on-demand gerendert werden sollen.

SSR Opt-out für einzelne Pages möglich.

Benötigt einen eingerichteten SSR Adapter.

`output` = `hybrid`

```
// /astro.config.mjs
import { defineConfig } from 'astro/config';

export default defineConfig({
  output: 'hybrid',
});
```

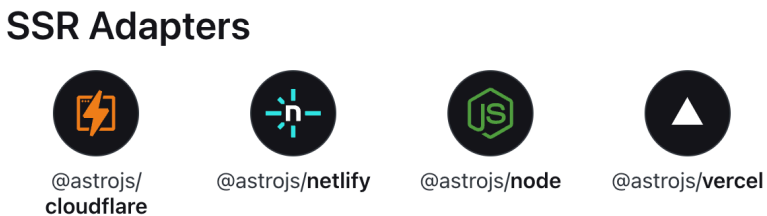
Nützlicher Default für Seiten, bei denen die meisten Pages statisch sind.

SSR Opt-in für einzelne Pages möglich.

Benötigt einen eingerichteten SSR Adapter.

SSR Adapters

Offizielle Adapters maintained vom Astro Team



& Deno

Community Adapters

- Bun
- AWS Lambda
- AWS Amplify

`server` output - Pre-rendering Opt-in

```
// /astro.config.mjs
import { defineconfig } from 'astro/config';
export default defineconfig({
  output: 'server',
});
```

```
---
// pages/my-page.astro
export const prerender = true;
---
```

`hybrid` output - Pre-rendering Opt-out

```
// /astro.config.mjs
import { defineconfig } from 'astro/config';
export default defineconfig({
  output: 'hybrid',
});
```

```
---
// pages/my-page.astro
export const prerender = false;
---
```

On-demand rendering features

- HTML Streaming (Demo)
- Request/Response API
- Cookie API
- API Routes (auch bekannt als Server Endpoints)

Request/Response support

Entpricht den `Request` und `Response` classes, die bekannt aus der `fetch` Web API sind.

```
---
import { getProduct } from '../api';

const response = Astro.response; // Response Web API

const product = await getProduct(Astro.params.id); // params support

// No product found
if (!product) {
  return new Response(null, {
    status: 404,
    statusText: 'Not found'
  });
}
---
<html>
  <!-- Page here... -->
</html>
```


HTML Streaming Demo

API Routes / Server Endpoints

- Sind im `pages` Ordner zu finden.

```
export const GET: APIRoute = ({ params, request }) => {
  return new Response(JSON.stringify({
      message: "This was a GET!"
    })
  )
}
export const POST: APIRoute = ({ request }) => {
  return new Response(JSON.stringify({
      message: "This was a POST!"
    })
  )
}
// ALL -> matched auf alle HTTTP methods
export const ALL: APIRoute = ({ request }) => {
  return new Response(JSON.stringify({
      message: `This was a ${request.method}!`
    })
  )
}
```

API Routes / Server Endpoints

Use case: Images generieren

```
// src/pages/my-image.png.ts
// URL: /my-image.png
export async function GET({ params, request }) {
  const response = await fetch("https://docs.astro.build/assets/full-logo-light.png");
  const buffer = Buffer.from(await response.arrayBuffer());
  return new Response(buffer, {
    headers: { "Content-Type": "image/png" },
  });
}
```


Aufgabe