Jasne — problem z „text/html dla module script” bardzo często wynika z **reguł URL Rewrite**, które kierują **również żądania do plików** (.js, .mjs, .css, .wasm, itp.) do Node’a, a potem Express (przez źle ustawiony fallback) oddaje index.html. Poniżej masz „dobre” i „złe” wzorce wraz z gotowymi regułami.

**Dlaczego to się dzieje**

* Reguła „łap wszystko” w web.config (match url=".\*") bez wyjątków → **każdy** request idzie do server.js.
* W Express kolejność jest zła lub katalog dist nie ten → express.static nie zwraca plików → fallback SPA oddaje index.html.
* Przeglądarka chciała moduł JS, ale dostała HTML ⇒ błąd MIME.

**Wzorzec A (polecany): „Wszystko do Node, a statyki serwuje Express”**

Tu **IIS przepuszcza** wszystko do server.js, a ty pilnujesz kolejności w Express: najpierw express.static(...), potem fallback. To najprostsze i spójne dla SPA.

<!-- web.config -->

<configuration>

<system.webServer>

<handlers>

<add name="iisnode" path="server.js" verb="\*" modules="iisnode" />

</handlers>

<rewrite>

<rules>

<!-- 1 reguła: całość do Node -->

<rule name="all-to-node" stopProcessing="true">

<match url=".\*" />

<action type="Rewrite" url="server.js" />

</rule>

</rules>

</rewrite>

<!-- Dodatkowo typy MIME, jeśli IIS musi serwować cokolwiek statycznego -->

<staticContent>

<remove fileExtension=".mjs"/><mimeMap fileExtension=".mjs" mimeType="application/javascript"/>

<remove fileExtension=".wasm"/><mimeMap fileExtension=".wasm" mimeType="application/wasm"/>

<remove fileExtension=".map"/><mimeMap fileExtension=".map" mimeType="application/json"/>

</staticContent>

<iisnode loggingEnabled="true" devErrorsEnabled="true"/>

</system.webServer>

</configuration>

**Warunek powodzenia:** w server.js masz:

app.use(express.static(dist)); // najpierw statyki

app.get(/^\/(?!.\*\.\w+$).\*/, …send index.html…); // potem SPA fallback tylko dla ścieżek bez kropki

**Wzorzec B: „Hybryda” (IIS serwuje statyki, Node dostaje resztę)**

Jeśli wolisz, aby **IIS** obsługiwał pliki (.js/.css/.png/...), zrób **wyjątki** w regule rewrite:

<!-- web.config -->

<configuration>

<system.webServer>

<handlers>

<!-- standardowy StaticFileHandler musi zostać -->

<add name="iisnode" path="server.js" verb="\*" modules="iisnode" />

</handlers>

<rewrite>

<rules>

<!-- 1) NIE przepuszczaj realnych plików/katalogów do Node -->

<rule name="SkipIfFileOrDir" stopProcessing="true">

<match url=".\*" />

<conditions logicalGrouping="MatchAny">

<add input="{REQUEST\_FILENAME}" matchType="IsFile" />

<add input="{REQUEST\_FILENAME}" matchType="IsDirectory" />

</conditions>

<action type="None" />

</rule>

<!-- 2) (opcjonalnie) jawnie wyklucz rozszerzenia SPA assetów -->

<rule name="SkipAssets" stopProcessing="true">

<match url=".\*\.(?:js|mjs|css|map|wasm|png|jpg|jpeg|gif|svg|ico|json|txt)$" ignoreCase="true" />

<action type="None" />

</rule>

<!-- 3) Resztę przekaż do Node -->

<rule name="ToNode" stopProcessing="true">

<match url=".\*" />

<action type="Rewrite" url="server.js" />

</rule>

</rules>

</rewrite>

<staticContent>

<remove fileExtension=".mjs"/><mimeMap fileExtension=".mjs" mimeType="application/javascript"/>

<remove fileExtension=".wasm"/><mimeMap fileExtension=".wasm" mimeType="application/wasm"/>

<remove fileExtension=".map"/><mimeMap fileExtension=".map" mimeType="application/json"/>

</staticContent>

</system.webServer>

</configuration>

**Plusy:** URL-e do plików nie dotykają Node’a.  
**Uwaga:** musisz mieć rolę **Static Content** w IIS i poprawne MIME (jak wyżej), inaczej IIS odda 404 lub zły typ.

**Antywzorzec (często źródło błędu)**

Reguła „Angular routes / SPA” kierująca **wszystko** wprost do index.html **przed** iisnode:

<!-- ZŁA kolejność / zły cel -->

<rule name="SpaToIndex" stopProcessing="true">

<match url=".\*" />

<action type="Rewrite" url="/dist/your-app/index.html" />

</rule>

<add name="iisnode" path="server.js" …/>

Albo reguła do Node **bez wyjątków**, po której w Express fallback łapie też pliki (bo app.get('\*', ...) stoi PRZED express.static). Skutek w obu przypadkach: main.js dostaje HTML → błąd MIME.

**Wirtualny katalog / aplikacja pod pod-ścieżką**

Jeśli Twoja aplikacja siedzi w IIS pod np. /myapp, nic nie zmieniasz w samych regułach – są **względem katalogu aplikacji**. Natomiast **Angular musi być zbudowany** z:

ng build --base-href /myapp/ --deploy-url /myapp/

Inaczej klient zażąda /main.js z root’a, wpadnie w fallback i dostanie index.html.

**Szybkie debugowanie rewrite**

1. Włącz **Failed Request Tracing (FREB)** dla statusów 200/404/500 i modułu Rewrite. Zobaczysz, która reguła zadziałała dla main.\*.js.
2. Sprawdź log iisnode (iisnode\) — widać, czy żądania do \*.js trafiają do Node.
3. DevTools → Network → kliknij main.\*.js → jeśli **Response** zaczyna się od <!doctype html>, to rewrite/fallback łapie assety.

**„Gotowce” do skopiowania**

* Chcesz mieć **prosto** i kontrolę w Express? → Wzorzec **A**.
* Chcesz zrzucić pliki na IIS i Node tylko do API/SPA-fallback? → Wzorzec **B**.

Jeśli podeślesz swój obecny web.config + krótki server.js, wskażę dokładnie, która reguła przepycha assety do index.html i podmienię ją na bezpieczny wariant.

Jasne — jeśli IIS (albo Express) serwuje pliki z rozszerzeniami, których **domyślnie nie zna** (np. .mjs, .wasm, .map, .webmanifest), to przeglądarka dostaje application/octet-stream i **modułowy skrypt/wasm** jest odrzucany. Poniżej szybka diagnoza i „gotowce”.

**1) Jak szybko potwierdzić problem**

* W DevTools → Network kliknij np. runtime.mjs / main.mjs / \*.wasm → sprawdź **Response Headers** → Content-Type nie może być application/octet-stream.
* Z serwera:  
  PowerShell:
* iwr https://twoj.host/myapp/runtime.mjs -Method Head | select -ExpandProperty Headers

lub cURL:

curl -I https://twoj.host/myapp/runtime.mjs

Szukaj Content-Type: application/javascript (lub text/javascript) dla .mjs i application/wasm dla .wasm.

**2) Jeśli IIS serwuje assety (hybryda)**

Dodaj mapowania MIME w web.config **w katalogu aplikacji** (albo przez IIS Manager → MIME Types). Daj też <remove> przed <mimeMap>, bo podwójne mapowanie powoduje 500.19:

<configuration>

<system.webServer>

<staticContent>

<!-- JavaScript, ES Modules -->

<remove fileExtension=".js" />

<mimeMap fileExtension=".js" mimeType="application/javascript" />

<remove fileExtension=".mjs" />

<mimeMap fileExtension=".mjs" mimeType="application/javascript" />

<!-- WebAssembly -->

<remove fileExtension=".wasm" />

<mimeMap fileExtension=".wasm" mimeType="application/wasm" />

<!-- Source maps & JSON -->

<remove fileExtension=".map" />

<mimeMap fileExtension=".map" mimeType="application/json" />

<remove fileExtension=".json" />

<mimeMap fileExtension=".json" mimeType="application/json" />

<remove fileExtension=".webmanifest" />

<mimeMap fileExtension=".webmanifest" mimeType="application/manifest+json" />

<!-- Fonty -->

<remove fileExtension=".woff" />

<mimeMap fileExtension=".woff" mimeType="font/woff" />

<remove fileExtension=".woff2" />

<mimeMap fileExtension=".woff2" mimeType="font/woff2" />

<remove fileExtension=".ttf" />

<mimeMap fileExtension=".ttf" mimeType="font/ttf" />

<!-- Obrazy (na wszelki wypadek, gdyby brakowało) -->

<remove fileExtension=".svg" />

<mimeMap fileExtension=".svg" mimeType="image/svg+xml" />

<remove fileExtension=".webp" />

<mimeMap fileExtension=".webp" mimeType="image/webp" />

<remove fileExtension=".avif" />

<mimeMap fileExtension=".avif" mimeType="image/avif" />

<remove fileExtension=".ico" />

<mimeMap fileExtension=".ico" mimeType="image/x-icon" />

<!-- CSS -->

<remove fileExtension=".css" />

<mimeMap fileExtension=".css" mimeType="text/css" />

</staticContent>

</system.webServer>

</configuration>

Uwaga: w starszych Windows Server/IIS .mjs i .wasm **nie istnieją** domyślnie — bez tych mapowań dostaniesz octet-stream.

**3) Jeśli Express serwuje assety (wszystko do Node)**

Wtedy IIS-owe MIME nie są używane dla tych plików. Dopnij nagłówek w express.static (albo zaktualizuj pakiet mime):

app.use(express.static(dist, {

setHeaders: (res, filePath) => {

if (filePath.endsWith('.mjs') || filePath.endsWith('.js')) {

res.setHeader('Content-Type', 'application/javascript; charset=utf-8');

}

if (filePath.endsWith('.wasm')) {

res.setHeader('Content-Type', 'application/wasm');

}

if (filePath.endsWith('.map') || filePath.endsWith('.json')) {

res.setHeader('Content-Type', 'application/json; charset=utf-8');

}

if (filePath.endsWith('.css')) {

res.setHeader('Content-Type', 'text/css; charset=utf-8');

}

}

}));

**4) Dwie częste pułapki**

* **Rewrite łapie pliki** i oddaje index.html → wtedy zobaczysz text/html (to Twój pierwotny błąd). Napraw: express.static **przed** fallbackiem i/lub wyklucz rozszerzenia w web.config.
* **Zły baseHref/deployUrl** → klient próbuje /main.js z root’a, trafia w fallback i dostaje index.html. Zbuduj Angulara dla pod-ścieżki (/myapp/).

**5) Bonus: sensowne cache’owanie (gdy nazwy są z hashami)**

Dla plików typu main.[hash].js możesz śmiało dać długi cache (a dla index.html – nie):

**IIS (gdy serwuje statyki):**

<system.webServer>

<staticContent>…</staticContent>

<httpProtocol>

<customHeaders>

<add name="Cache-Control" value="public, max-age=31536000, immutable" />

</customHeaders>

</httpProtocol>

</system.webServer>

**Express (gdy serwuje statyki):**

app.use(express.static(dist, { maxAge: '1y', immutable: true }));

// a w fallbacku do index.html:

res.setHeader('Cache-Control', 'no-store');

Podsumowując: dodaj mapowania MIME (zwł. .mjs i .wasm) **tam, gdzie faktycznie serwujesz assety** (IIS albo Express). Potem jednym rzutem oka w DevTools/curl -I potwierdź, że Content-Type jest poprawny — i problem zniknie.