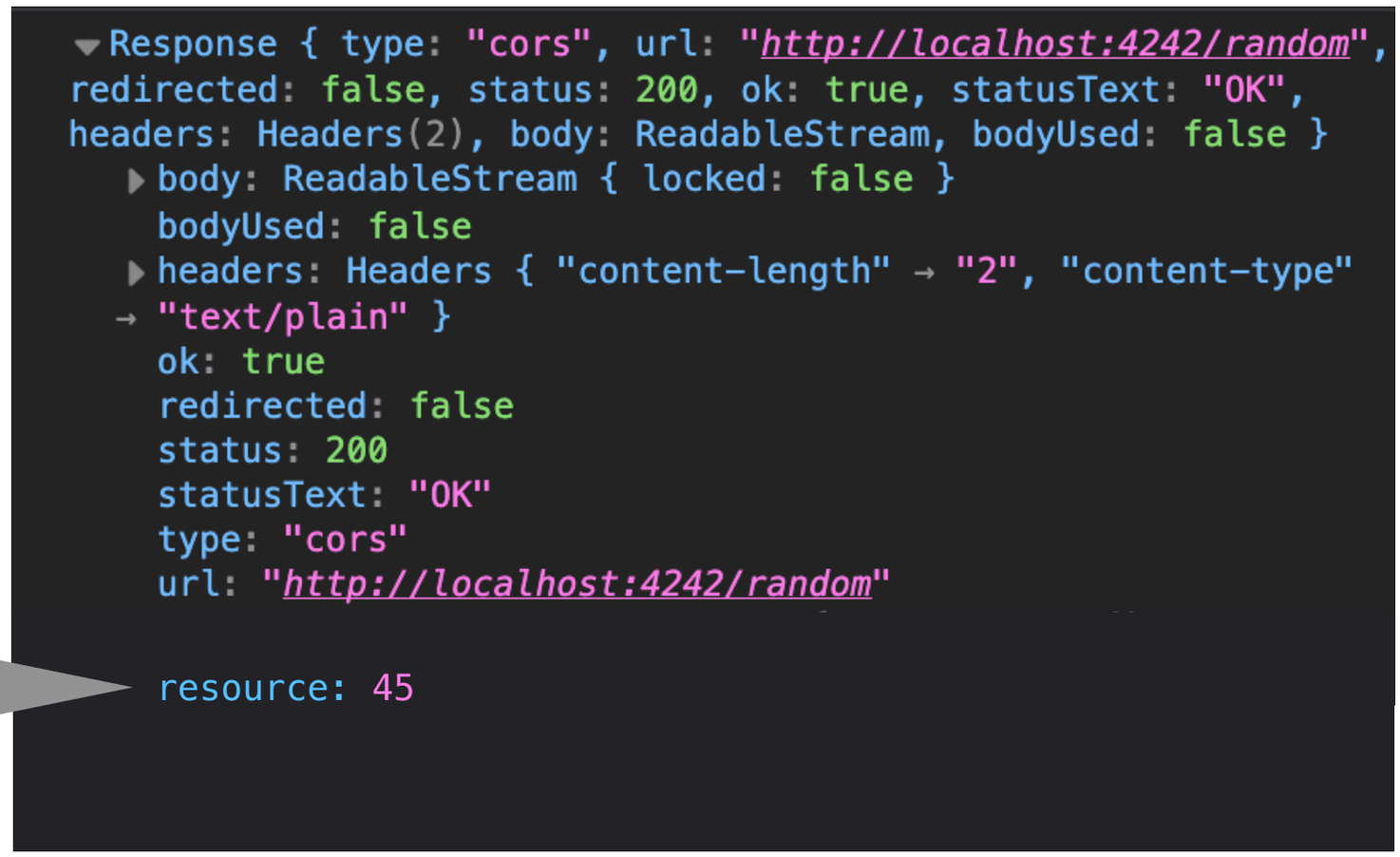
Från Response till Resurs

Response-objektet innehåller en del information om hur processen gick, som tex. status och statusText.

Men varför har inte *response* ett attribut som innehåller resursen vi efterfrågade i förfrågan?

Detta hade varit fint!



Från Response till Resurs (forts.)

paket på internet

För att förstå vad som har hänt behöver vi få en **mycket grundläggande översikt** över vad internet är och hur det fungerar.

Internet är en ofantligt stor samling datorer och routrar (noder) som är kopplade med varandra på ett relativt oordnat sätt (alltså ett enormt nätverk).

När en agent (t.ex. klient) ska kommunicera med en annan (t.ex. server) skickar den ett meddelande.

Meddelandet kommer att delas upp i mindre datapaket och skickas via internet, dvs. från nod till nod tills det kommer fram. Varje paket kan ta sin egen väg på internet, de behöver inte gå via samma noder.

Med andra ord: Ett meddelande kommer inte helt fram till mottagaren i ett stycke, utan mottagaren måste sätta ihop alla paket, som kan komma i oordning.

Från Response till Resurs (forts.)

Översatt till fetch och vårt exempel:

Servern har tagit emot förfrågan och skickat en respons:

```
// Kod från servern
...
return new Response (randomInt, { headers: headersCORS });
```

Som vi tidigare sett består en response av statusrad, headers och body. Lite förenklad kan vi säga att:

Statusrad och headers skickas först och kommer till webbläsaren först.

Body, som innehåller resursen, delas upp i datapaket, som skickas var för sig.

När webbläsaren får statusrad och headers så skapar den ett objekt av klassen Response. Resursen ingår alltså inte i Response-objektet, för den kommer (som body) vid sidan om.

En till *promise*

Löftet om resursen (till skillnad från löftet om responsen)

Resursen (body) kommer, som sagt i paket, som måste sättas samman.

Detta är ju också en process som kan ta tid och som därför hanteras asynkront. Som vanligt i JS, hanteras den med hjälp av en Promise.

Response-objekt har en metod *text()*, som hanterar detta.

Metoden text() returnerar en ny *promise*, som representerar processen med att sätta upp resursen från alla paket. I processen ingår att vänta på att alla paket har kommit in och sedan sätta samman dem till ett värde.

> Metoden text() ska användas om responsen har headern Content-Type satt till "text/plain" (som är standard).
>
> Det finns även en metod json() när headern Content-Type är "application/json".
>
> Det finns andra metoder för andra typer av data, men de är inte aktuella för oss.

En till *promise*

Löftet om resursen (till skillnad från löftet om responsen)

Resursen (body) kommer, som sagt i paket, som måste sättas samman.

Detta är ju också en process som kan ta tid och som därför hanteras asynkront. Som vanligt i JS, hanteras den med hjälp av en *promise*.

Påminnelse: fetch returnerar en *promise* som kommer att lösas till en *response*.

Det finns en annan funktion (en metod, för att vara mer exakt) som returnerar en *promise* som kommer att lösas till resursen.

> Det är en metod av Response-objektet, som heter *text()*.
>
> Metoden text() returnerar en ny *promise*, som representerar processen med att sätta ihop resursen från alla paket. I processen ingår att vänta på att alla paket har kommit in och sedan sätta samman dem till ett värde, som är ju själva resursen.
>
> > text() ska användas när Response-objektets header Content-Type är "text/plain".
> >
> > Det finns även en metod **json()**, som används när Content-Type är "application/json".
> >
> > Det finns andra metoder för andra Content-Type, men de är inte aktuella för oss.

Kod

En version av koden som hanterar hämtning av en resurs från en server

```
const request = new Request(server–URL);
```

Skapa förfrågan

```
const promiseResponse = fetch(request);
```

Skicka förfrågan. fetch returnerar en *promise* som kommer att lösas till en *response*

```
promiseResponse.then(handleResponse);
```

Ange funktionen som ska anropas när promiseResponse har lösts till *response*

```
function handleResponse (response) {
```

Deklaration av funktionen som anropas när *response* är klar.
Argumentet i anropet kommer att vara *response*.

```
  const promiseResource = response.text();
```

response har metoden .text(), som returnerar en promise som kommer att lösas till resursen.

```
  promiseResource.then(handleResource);
}
```

Ange funktionen som ska anropas när promiseResource har lösts till resursen

```
function handleResource (resource) {
  // Do something with the resource
}
```

Deklaration av funktionen som anropas när resursen är klar.
Argumentet i anropet kommer att vara resursen.