> *The only thing worse than not knowing why some code breaks is not knowing why it worked in the first place! It's the classic 'house of cards' mentality: "it works, but I'm not sure why, so nobody touch it!" You may have heard, 'Hell is other people' (Sartre), and the programmer meme twist, 'Hell is other people's code.' I believe truly: 'Hell is not understanding my own code'.*
> —Kyle Simpson, You Don't Know JS: Async & Performance

Modules

1 JS splitsen in verschillende bestanden

We willen al een aantal basiscitaten klaarzetten in de JS-code. Dat zijn vele tientallen regels 'data'. Het is zinvol om deze data in een apart bestand te zetten, gescheiden van de programmeerlogica. De basiscitaten komen zo apart in het bestand 'citaten.js' terwijl de rest van de code in 'script.js' blijft staan. Je roept dan beide bestanden op in je HTML met het `script` element.

 Het volgende filmpje toont hoe het splitsen van onze JS code in verschillende aparte bestanden de logische opbouw en leesbaarheid van onze code ten goede komt. De juiste volgorde waarin je de bestanden oproept in de `head` van het HTML-document is uiteraard belangrijk!

2 Modules

> Meer info over modules: zie https://javascript.info/modules-intro.

In de vorige video hebben we onze code al opgesplitst in verschillende bestanden. Maar we kunnen nog een stap verder gaan. We kunnen onze code opdelen in *modules*. Een module is een *stukje code dat een bepaalde functionaliteit bevat* . Zo'n module kan je dan importeren in een ander bestand. Dit is handig omdat je zo je *code kan hergebruiken* en je *code overzichtelijk blijft* .

Waarom kunnen we niet gewoon verder blijven werken met aparte `script` elementen in de HTML-code? Onze beperkte code heeft nu nog maar twee JS-bestanden. Maar stel je voor dat de applicatie een heel stuk groter wordt en verspreid wordt over tientallen verschillende bestanden. Er onstaan dan heel wat moeilijkheden:

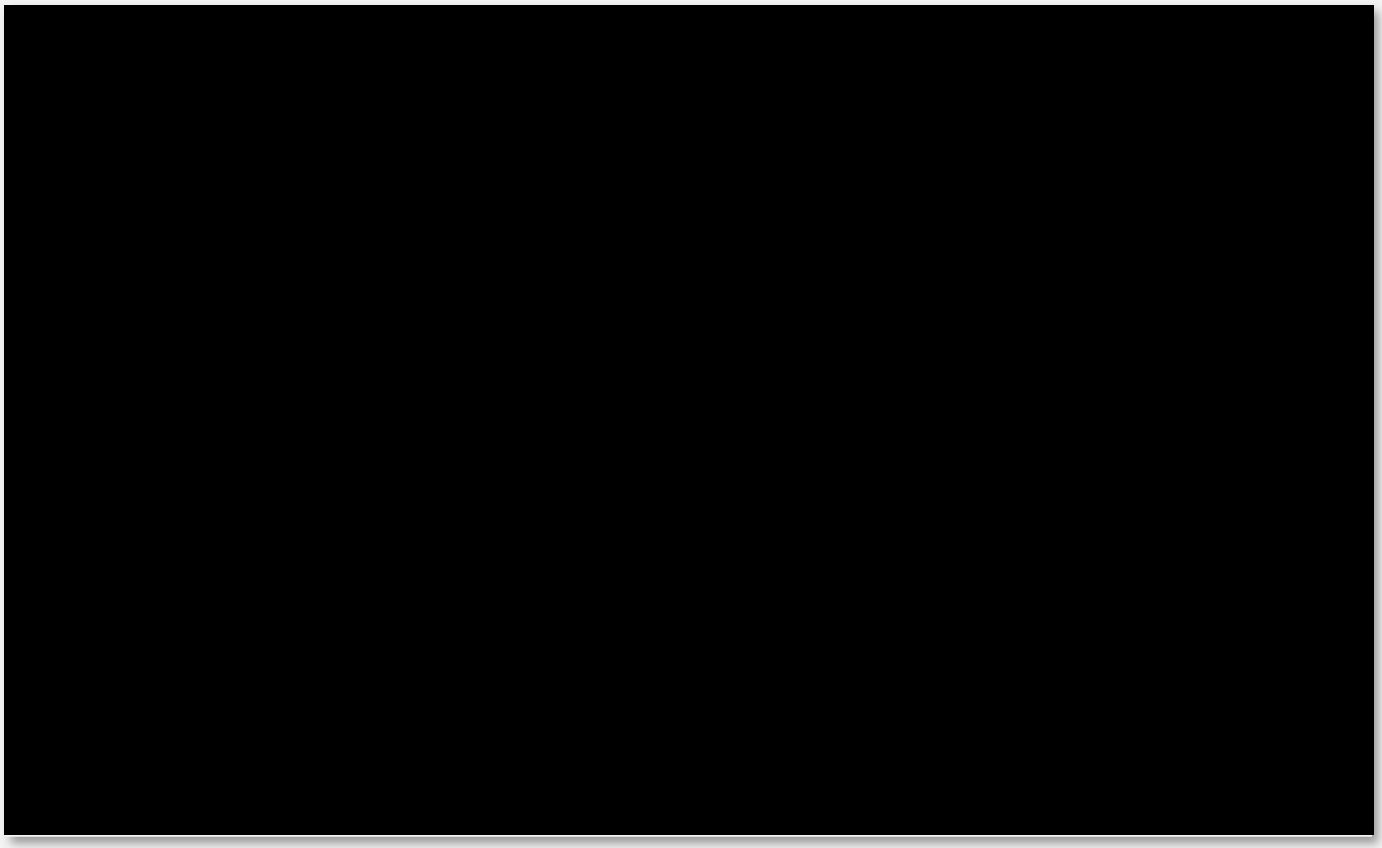
- 'global namespace pollution': misschien gebruik je in die tientallen bestanden twee keer de variabele 'stap', die dan mogelijk met elkaar in conflict komen.
- 'Afwezigheid van dependency management': een bepaald script in bestand 3 heeft iets nodig dat in bestand 7 gedefinieerd is. Je bent dus zelf

verantwoordelijk voor de juiste volgorde waarin je de bestanden in de HTML laadt.

- Moeilijker onderhoud en debuggen: De opeenvolging van losse bestanden vertelt je niets over hoe alles met elkaar samenhangt. Zoeken naar fouten wordt moeilijker naarmate er meer losse bestanden zijn.
- Performance overwegingen: Het importeren van veel scripts in HTML kan leiden tot een verhoogde laadtijd van de pagina, omdat elk script afzonderlijk moet worden gedownload en uitgevoerd. Dit kan de algehele prestaties van de applicatie beïnvloeden, vooral op langzamere netwerkverbindingen.

 Onderstaand filmpje gaat in op volgende topics:

- overstap naar modules;
- `import` / `export`;
- `defer` attribuut vervalt.



3 CSS update: classes en custom properties toevoegen

Het laatste filmpje van dit hoofdstuk legt de klemtoon weer helemaal op CSS. We voegen wat CSS-code toe en beklemtonen vooral volgende onderwerpen:

- het belang van klassen in CSS;
- CSS custom properties (‘variabelen’).

↑ Naar top