JS 2강. 콘솔에 출력, script async와 defer의 차이점

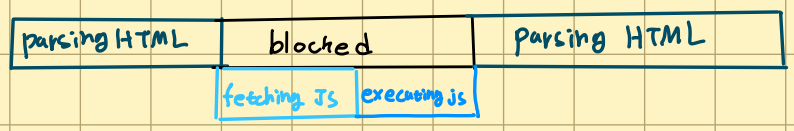
- Wep apis는 JS 언어 자체에 포함된 요소가 아니라 브라우저가 제공하는 기능(함수)이다.

  ex) console.log()

- async vs defer

① html문서에 JS 연결 기존 방법 1 : head에 포함

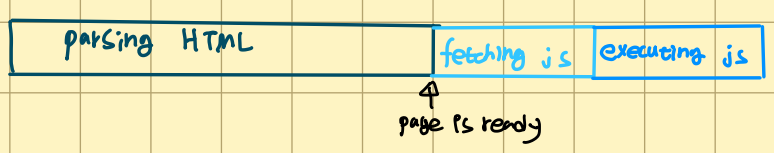
```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <script src="main.js"></script>
  </head>
  <body></body>
</html>
```



단점 : JS 파일 크기가 크거나 인터넷이 느린경우, JS를 다운받고 실행시키는데 많은시간이 소요됨 ⇒ 소요되는 시간 만큼 사용자들이 웹 페이지 화면을 볼수 없음 + JS의 DOM 조작 문제등 (HTML과 JS관계) 기능 문제 발생

② html문서에 JS 연결 기존 방법 2 : body에 포함

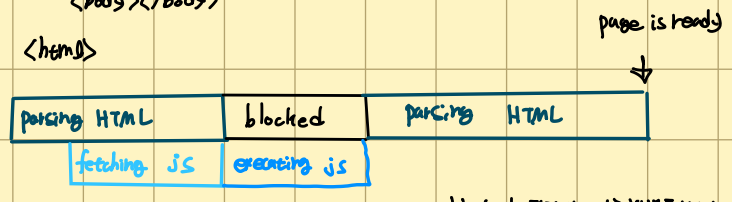
```
<!DOCTYPE html>
<html lang="ko">
  <head></head>
  <body>
    <script src="main.js"></script>
  </body>
</html>
```



- 장점 : JS를 다운 받는건 사용자가 웹 페이지 화면을 볼수 있음
- 단점 : 웹사이트가 제대로 동작하기 위해 JS가 필요하다면, 정상적인 웹페이지를 보는데 JS를 다운받고 실행 할때까지 기다려야 함

③ head + async

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <script async src="main.js"></script>
  </head>
  <body></body>
</html>
```

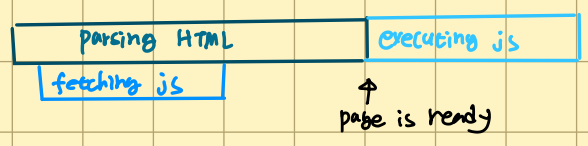


- 장점 : JS를 백그라운드로 다운받고, 실행시킬때는 blocked 되기에 기존방법들보다 기다리는 시간이 줄어듦. (executing js 하는데)
- 단점 : 사용자가 웹페이지를 보는데 시간이 다소 걸림 + JS의 DOM 조작 문제등 (HTML과 JS관계) 기능 문제 발생

④ head + defer

head + async

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <script defer src="main.js"></script>
  </head>
  <body></body>
</html>
```

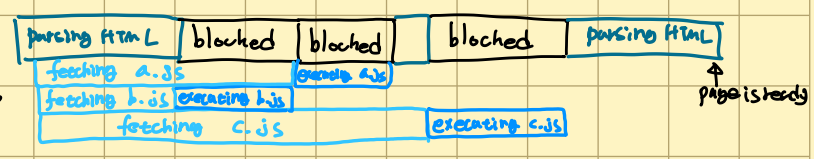


- 장점 : JS를 다운받는것을 병렬처리하여 시간 감소.

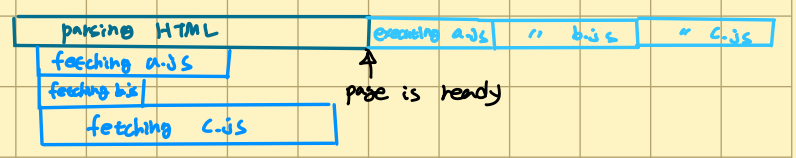
  HTML parsing 후 JS를 실행하기에 사용자가 웹페이지 화면을 볼수있고 DOM 조작 등에서 문제가 발생하지 않게됨

- async vs defer
  - async는 JS 파일을 다운받으면 바로 JS를 실행시키기에, 사용자가 웹화면을 보는데 시간이 오래걸리고 각 JS 코드 실행 순서를 예측하기 어렵.

실행되는 순서가 제각각



  - defere는 HTML을 parsing하는동안 JS 파일들을 다운받고, parsing이 끝나면 순서대로 각 JS를 실행시키기에 웹 화면도 빠르게 볼수 있고 JS 코드 실행 순서 예측가능



- 'use strict';
  - JS는 유연한 문법을 가지는데, 문법적으로 틀렸지만 브라우저에서 실행하면 에러가 나지 않는다. 하지만 후에 문제를 야기 시킬수 있으므로 이 유연한 문법을 제한하고자 할때 'use strict'; 를 사용할수 있다.

ex)

a = 6; ⇒ 선언문이 없지만 에러가 일어나지 X

→

'use strict';

a = 6; ⇒ 에러발생