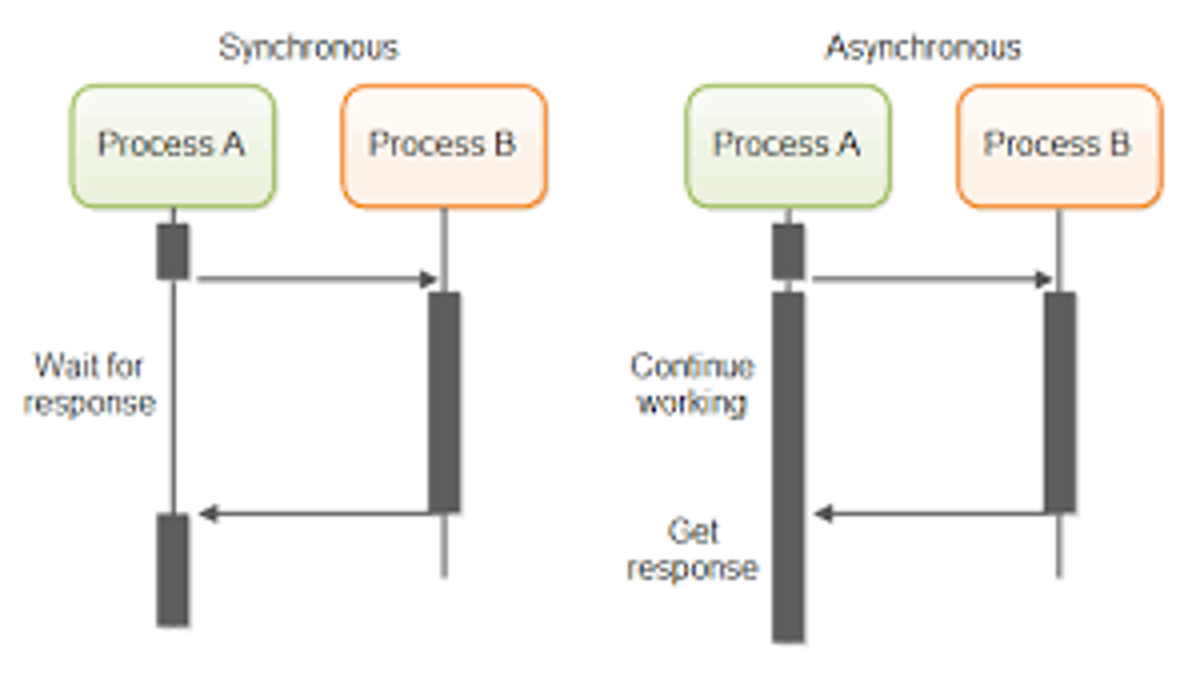
# Szinkron és aszinkron programozás



## Az aszinkron programozás előnyei

* Minden szkript egyenként töltődik be. Ez egyenlő a sebességgel, az érzékenységgel és a jobb felhasználói élménnyel.
* Megszünteti az oldalbetöltési késéseket. Tehát nincs szükség további oldalfrissítésekre az új kérések végrehajtása közben.
* Több funkciót is használhat egyszerre, még akkor is, ha más kérések még futnak.
* Az aszinkron alkalmazások nagymértékben méretezhetők, és kevés erőforrást igényelnek a működésükhöz.
* Még akkor is, ha egy kérés lassan válaszol, ez nem befolyásolja mások válaszidejét.
* Egy szál meghibásodása nem akadályozza a többiek feldolgozását.
* A beépített visszahívások lehetővé teszik a hibaüzenetek testreszabását.

## Az aszinkron programozás hátrányai

* Sok visszahívást és rekurzív funkciót igényel, amelyek a fejlesztés során nehézkesek lehetnek.
* Ha a visszahívásokat nem használják hatékonyan, akkor a felhasználó semmilyen módon nem tudja megtudni, hogy egy kérés sikertelen-e vagy sem, különösen a POST-kérések benyújtása közben.
* Az aszinkron betöltést használó webalkalmazásokat nehéz lehet feltérképezni olyan keresőmotorok számára, mint a Google és a Bing.
* Az aszinkron szkripteket nehéz lehet egyes programozási nyelvekben (*Python, C #, Java, PHP*) megvalósítani.
* A kód nehezebben átlátható, és nehéz a hibakeresés.
* Számolnunk kell a „***Race Condition”***-ökkel, azaz olyan eseményekkel, mikor a konkurens szálak ugyanahhoz az adathoz nyúlnak hozzá és módosítják úgy, hogy a módosítás sorrendje nem garantált. Ennek köszönhetően előállhat olyan helyzet, mikor egy feltétel az egyik szálon igazra értékelődik ki, de mire belép a feltétel igaz ágába, az adatot már megváltoztatta egy másik szál és ő erről nem tud semmit!

## A szinkron programozás előnyei

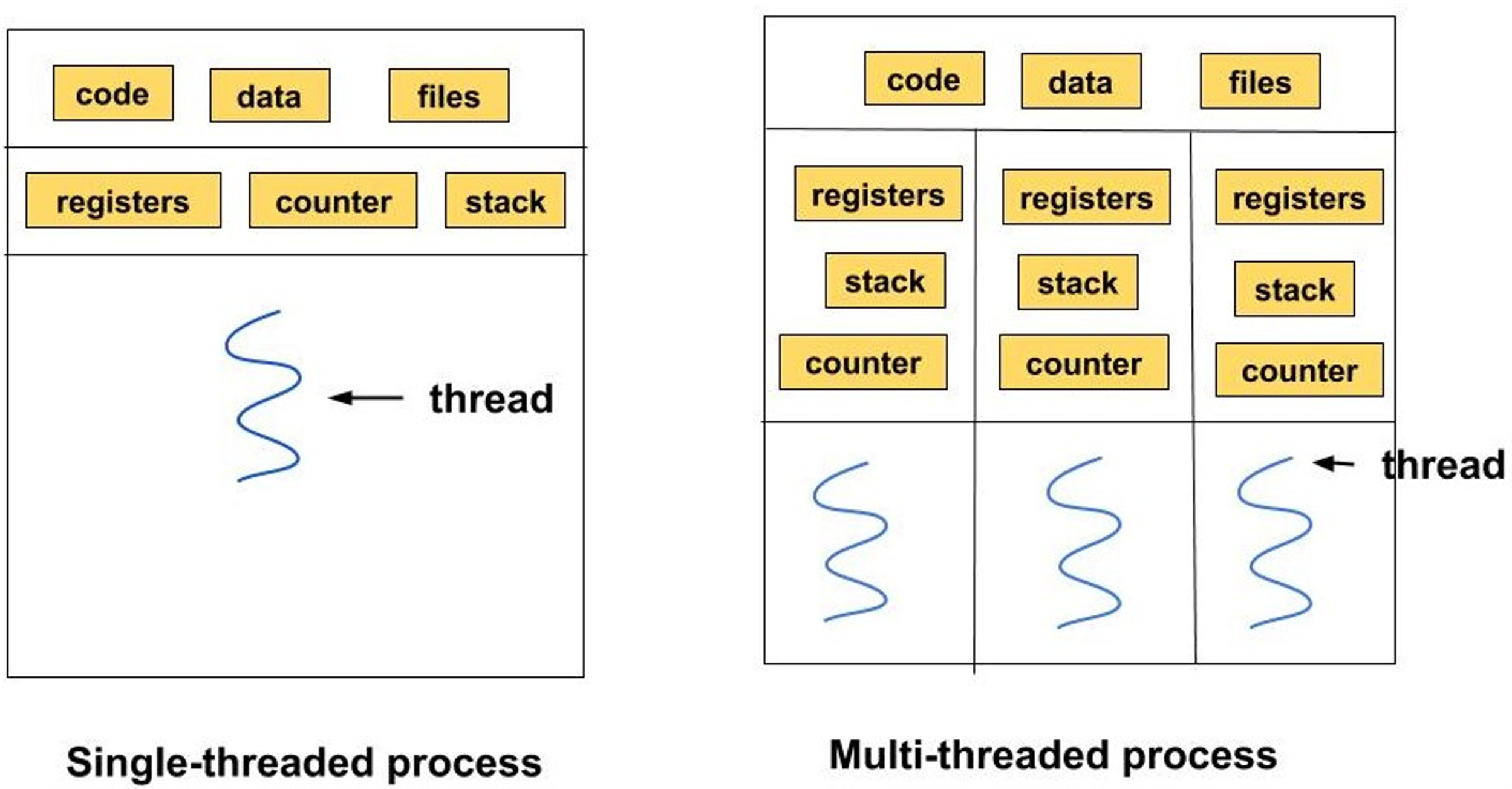
* Kevesebb kódolási know-how-t igényel, és minden programozási nyelv támogatja.
* Még akkor is, ha nincsenek személyre szabott visszahívások a kérelemhibákhoz, az ügyfél (böngésző) alapértelmezés szerint kezeli az ilyen hibákat.
* Jobb CPU feladatok végrehajtására.
* A keresőmotorok a szinkron weboldalakat könnyebben feltérképezik.
* Ideális egyszerű kérések benyújtásához.

## A szinkron programozás hátrányai

* A betöltési idő lassú lehet.
* Nincsenek beépített visszahívási módszerek.
* Amikor egy szál zárolva van, mások is blokkolódnak.
* Ha nem képes egyszerre több műveletet végrehajtani, csökkenhet a felhasználói élmény.
* Ha egy kérés sikertelen, akkor a teljes program is nem reagál.
* Hatalmas erőforrásokra lehet szükség további szálak kezeléséhez, ha a kérelmek száma megnő.

## melyik a jobb?

Feladattól függ!



# Node.js vagy PHP

Mindkét technológiát más-más időben és technológiai klímában vezették be (*PHP – 1995, node.js – 2009*), mindkettő erős alternatívának számít a webalkalmazások hátterében, nyílt forráskódú, és jelentős támogató közösséggel rendelkeznek.

A PHP a legmegbízhatóbb szerveroldali szkript nyelvek közé tartozik az olyan óriások alkalmazásában, mint a *Facebook*, a *Wikipedia* és a *Tumblr*. Ez a nyelv 1995 óta létezik, és folyamatosan profitál a körülötte kialakult nagy közösségből.

A 2009-ben elindított Node.js gyorsan megszerezte dominanciáját a legfontosabb webalkalmazás-kategóriákban. A *LinkedIn*, a *Netflix*, a *Medium* és más nagy szereplők által használt népszerű aszinkron architektúrája biztosítja a lekérdezések kezeléséhez elengedhetetlen hatékonyságot és sebességet.

## Kódolás

A Node.js több kódsort igényel ugyanazon funkciók végrehajtásához, mint a PHP, de az egységesen használt Javascript miatt kevesebb új ismeretet igényel, mint a PHP.

## Kliens oldali kódolás

Ha a projekt nagyobb részt statikus, amely kevés ügyfélkérést igényel, és a célközönsége kisebb teljesítményű eszközöket használ, a PHP a megfelelő választás.

Ha a használati eset olyan dinamikus webhelyeket tartalmaz, amelyek gyakran küldenek szerverkéréseket, és dinamikus felhasználói felülettel rendelkezik, akkor érdemes a Node.js-t használni.

## Sebesség

A Node.js aszinkron technológiája miatt egyértelműen gyorsabbá teszi.

## Teljesítmény

## Keretrendszerek

### PHP

* Laravel,
* CodeIgniter,
* CakePHP,
* Phalcon

### Node.js

* Meteor,
* Derby
* Sails
* Express.js
* Koa.js
* Meteor.js
* Socket.io - Egyszerűen használható csevegőalkalmazások a Socket.io segítségével.
* Nest.js
* Sails.js- Modernizált adatorientált MVC-keretrendszer.
* Total.js

## Adatbázis

## Közösség

## Modulok

## Tárhely

## Mikor válasszuk a Node.js-t a PHP helyett?

## Mikor válasszuk a PHP-t a Node.js helyett?

## A Node.js és a PHP korlátai

# Történet[[1]](#footnote-1)

A HTML 1.0-nál csak az alapvető szerkezet kialakítása volt megvalósítható, olyan elemeké, mint a bekezdések, hivatkozások, fejlécek és felsorolások. A 2-es verzió sem hozott sok újítást, de lehetőség volt például félkövér vagy dőlt betűk használatára, illetve képek beillesztésére. Ezen verzió kiegészítésében már megjelentek az űrlapok és a kiválasztható opciók lehetőségei. Ha pl. egy button megjelenését szerettük volna megváltoztatni használat után, akkor ahhoz az akkori szabvány semmilyen lehetőséget nem biztosított. Teljes egészében az akkori böngészőkön múlt a megvalósítás. Nagy verseny volt köztük.

1. Lynx (1993)
2. Netscape (1994)
3. Opera (1994)
4. InternetExplorer (1995)
5. Mozilla (1999)
6. Safari (2003)



1. ábra A fejlődésben szerepet játszó böngészők

## JavaScript

Eredetileg Brendan Eich, a Netscape Communications mérnöke fejlesztette ki; neve először Mocha, majd LiveScript volt, később „JavaScript” nevet kapott, és szintaxisa közelebb került a Sun Microsystems Java programozási nyelvéhez. A JavaScriptet először 1997–99 között szabványosította az ECMA „ECMAScript” néven. A jelenleg is érvényes szabvány az **ECMA-262** Edition 3 (*1999. december*), ami a JavaScript 1.5-nek felel meg. Ez a szabvány egyben ISO szabvány is.

# node.js alapok

## Install

* https://nodejs.org/en/download/
* <https://babeljs.io/>

## Alapmodulok

|  |  |
| --- | --- |
| Alapmodulok | Leírás |
| **http** | létrehoz egy HTTP-kiszolgálót a Node.js-ben. |
| **assert** | teszteléshez hasznos állítási függvények halmaza. |
| **fs** | fájlrendszer kezelésére használható. |
| **path** | módszereket tartalmaz a fájl elérési útjainak kezelésére. |
| **process** | információkat és vezérlést biztosít az aktuális Node.js folyamatról. |
| **os** | információkat nyújt az operációs rendszerről. |
| **querystring** | URL-lekérdezési karakterláncok elemzésére és formázására használt segédprogram. |
| **url** | modul segédprogramokat biztosít az URL-feloldáshoz és -elemzéshez. |

továbbiak: <https://www.npmjs.com/>

# debugger és sctrict mód

# async/await

# callback

1. https://www.youtube.com/watch?v=W4wWdmfOibY [↑](#footnote-ref-1)