Node.js

Node - צומת. כשיש רשת של מחשבים, מחשב בודד נקרא Node.

מנוע להרצת קוד JavaScript מחוץ לדפדפן.

בשנת 2009 היו שלוש אפשרויות עיקריות לפיתוח שרתים:

1. מייקרוסופט - עלתה הרבה כסף, והצריכה ידע בכלי מייקרוספוט, וגם הוכחה כלא יעילה ל-scale של אינטרנט (ראה מקרה MySpace למשל)
2. ג'אווה - מצד אחד ג'אווה הייתה שפה קשה ומסובכת, ורצה בעיקר על שרתי יוניקס יקרים.
3. לאמפ - LAMP - Linux, Apache, MySQL, PHP/Perl/Python

שרת ה-Web של ה-LAMP היה Apache

אפאצ'י, היה שרת CGI - Common Gateway Interface



נוד פותח על ידי בחור בשם ריאן דאל



מרכיבי Node.js:

1. מנוע V8 של Chromium
2. ספריית לינוקס שנקראת libuv כדי לממש את הפעולות האסינכרוניות
3. איבנט לופ - event loop

תוצר הלוואי המבורך:

1. ג'אווה סקריפט בפרונטאנד
2. בבקאנד PHP
3. דטהבייס: MySQL
4. פרונטאנד: ג'אווהסקריפט
5. בקאנד: ג'אווהסקריפט
6. דטהבייס (מונגודיבי): ג'אווהסקריפט

זה בעצם ה-FullStack המקורי. המושג נטבע ברגע שהייתה ג'אווהסקריפט בכל אחד מרכיבי ה-stack.

Nodemon

מאפשר להריץ תוכניות נוד רגישות לשינויי קוד. כלומר ההרצה תתרחש מחדש בכל פעם שנשמור שינויים לקוד.

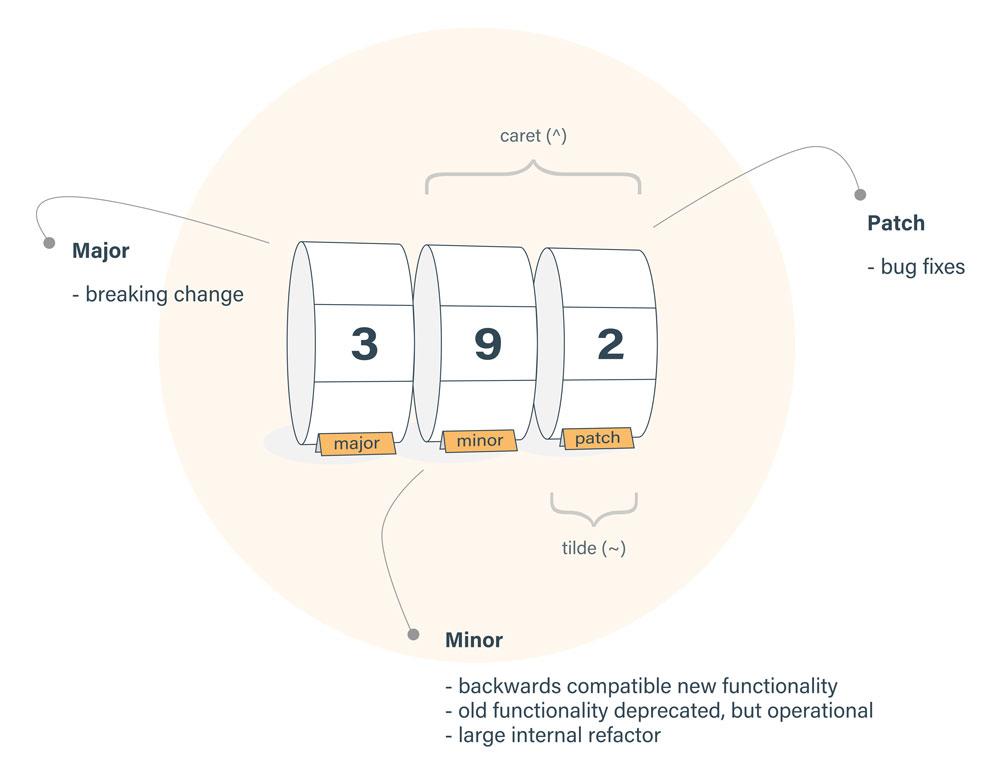
התקנה: npm i -g nodemon (התקינו כאדמין)

מודולים.

ישנם 3 סוגי מודולים:

1. מודולים שאנחנו מפתחים בעצמנו (custom modules)
2. מודולים שנגישים לנו מתוך Node.js עצמו
3. מודולים הזמינים ציבורית (או ארגונית) דרך NPM. אלו בעצם custom models שלא אנחנו פיתחנו.באמצעותם ניתן לנצל את כח הקהילה/הארגון כדי לייעל את הקוד שלנו ולא לפתח את הגלגל מחדש בכל פעם.

גרסאות מודולים ב-NPM



לכל מודול NPM יש גרסה בת 3 מספרים בפורמט:

Major.minor.patch

למשל:

1.4.5

גרסת ה-patch מקודמת בתיקוני באגים

גרסת ה-minor מקודמת בתוספות לקוד

גרסת ה-major מקודמת בתוספות לקוד ששוברות גרסאות קודמות.

בקבתי ה-package.json שלנו בחלק ה-dependencies נוכל לקבוע אילו גרסאות נרצה לקבל באמצעות סימנים מיוחדים:

1.4.5 - יביא לנו אך ורק את גרסה זו ספציפית

^1.4.5 - יביא לנו את גרסת ה-minor הגבוהה ביותר שימצא מעל הגרסה המצויינת

~1.4.5 - יביא לנו את גרסת ה-patch הגבוהה ביותר שימצא מעל הגרסה המצויינת

\* או latest יביא לנו את הגרסה האחרונה שימצא

תפקיד ה-package-lock.json

תפקידו לוודא שכל חברי הצוות שעובדים על פרוייקט אחד מסויים, יעבדו על אותן גרסאות של dependencies (בפועל אי אפשר לאכוף את זה, אבל בכל פעם שמישהו מריץ npm install ה-dependencies שלו יתעדכנו עד לרמת ה-package-lock ולא מעבר, ורק כשמישהו יריץ יזום npm update ויעלה גרסה מעל למה שמצויין ב-package-lock רק אז, אחרי שהוא יעשה קומיט לקובץ ה-lock כל שאר חברי הצוות יצטרכו להסתנכרן איתו

יצירה יזומה של קובץ package.json: npm init -y

פיבלוש מודולים ל-npm:

1. npm login
2. Npm publish

עידכון מודול ב-npm:

1. לקדם את הגרסה בקובץ package.json לפי השינוי שעשינו
2. npm publish

Debuuging

הפעולה שאנו המפתחים עושים כדי למצוא שגיאות בקוד. בעיקר שגיאות לוגיות.

כיום אנו יודעים לדבג באמצעות console.log

אבל ל-IDEs בדרך כלל יש debugger-ים יותר משוכללים, ועם המעבר לתוכנת צד שרת, הגיע הזמן להציג אותם.

Ts-node

npm i ts-node -D

זהו מודול המאפשר לקמפל טייפסקריפט בזכרון ולהריץ אותו מבלי שיישמר לקובץ.

משמעות @types

כל npm module הכתוב ב-JavaScript ורוצה לאפשר למפתחים להשתמש בו גם ב-TypeScript צריך לפרסם Types. למה:

1. כדי שהקומפיילר של TypeScript יוכל לאכוף את הטיפוסים שלו
2. וכדי לאפשר InteliSense ב-IDE

פירסום types נעשה באחת משתי דרכים:

1. הכללת קבצי \*.d.ts במודול המפובלש
2. או פרסום הטייפים כ-npm module נפרד בספרייה הגלובלית של @types. מי שבוחר בשיטה זו, ומפבלש ל-npm מודל בשם למשל myLovelyModule יצטרך לפבלש מודול בשם @types/myLovelyModule

Express

פריימוורק לכתיבת קוד שרת בנוד\טייפסקריפט. נועדה להקל עלינו את כתיבת השרת, להכליל פעולות שנרצה לעשות באופן די קבוע ושהשרת הבסיסי של נוד לא מאפשר לנו, למשל:

* לפענח יו אר אלים
* לפענח פרמטרים
* לפענח post bodys
* להחזיר לקליינט תשובות בפורמטים שונים
* לסדר את הקוד שלנו

בסה"כ אקספרס לנוד, היא כמו ריאקט לואנילה ג'אווהסקריפט. תמיד נעדיף להשתמש באקספרס על פני נוד בסיסי.

מודל העבודה של אקספרס, מודל ה-middleware

