**DOC­S – Advanced JavaScript 2023-01-09**

**IIFE**

הדרך הנכונה לעבוד ב native js, על מנת ששום פונקציה או משתנה ( אשר נמצאים ב global scope) , לא יידרסו לנו את השמות.

**window**

האובייקט הראשי ב JS (אבל ב run time של הדפדפן) – הכל נמצא בתוכו.

לדוגמה – alert, זו פונקציה שנמצאת ב window, ולכן אני יכול לעשות window.alert()

בגלל שכל התשתית של javascript נמצאת בתוך אובייקט ה window, לא צריך לכתוב window , כי js אוטומטית תחפש את הפקודה בתוך ה window.

**Strict mode**

העברת Javascript למצב "מחמיר" – בו JS לא מסכימה לבצע מספר פעולות שעלולות להוות באגים פוטנציאליים. לדוגמה: שימוש במשתנה ללא יצירה (הצהרה) ע"י var,let או const

**Synchronous function – פונקציה סינכרונית**

כל פונקציה (או שורת קוד) שכאשר קוראים לה, הקוד הקורא ממתין עד לסיומה. כלומר, הקוד אחר כך לא יוצא לדרך (התוכנית "תקועה), עד שהקוד הנוכחי לא מסיים לרוץ (single threaded).

בגלל שממתינים לה (לפונקציה) היא נקראת פונקציה סינכרונית. היא תמיד יכולה להחזיר את תוצאות החישוב שלה ע"י return, ולכן אנחנו יכולים לקבל את תוצאות החישוב לתוך משתנה.

**Asynchronous function**

זו היא כל פונקציה שכאשר קוראים לה, הקוד הקורא לא ממתין עד לסיומה וממשיך מיד לפקודה הבאה אחריה.

הפונקציה מתחילה לעבוד ברקע, אך הקוד שקרא לה, כבר ממשיך הלאה.

בגלל שלא ממתינים לפונקציה אסינכרונית , היא לא יכולה אף פעם להחזיר את תוצאת החישוב (או להחזיר return), לכן לא ניתן לבצע השמה מהפונקציה למשתנה שאמור להכיל את תוצאת החישוב.

החזרת התשובה מפונקציה אסינכורונית , חייבת להיות ע"י callback.

על הפונקציה לקבל בצורה כלשהי , callback, כך כאשר יעבור פרק הזמן והמידע הרלוונטי יהיה זמין לנו, נוכל לקרוא ל callback ולשלוח לתוכו את התוצאה.

Event loop explanation - <https://www.youtube.com/watch?v=8aGhZQkoFbQ&vl=en>