קבצים

HTML5 מאפשר לגשת בצד **הלקוח** לקבצים המקומיים (השמורים על מחשב הגולש) באמצעות חבילת פונקציות בשם File API.

מדוע יש צורך לטפל בקבצים בצד לקוח?

עד היום, מפתחים נאלצו להעלות את קבצים לשרת האתר כדי לפתוח אותם ולעבוד עם התוכן שלהם, לעיתים, רק כדי לגלות שלא מדובר בקובץ הנכון, או שהוא גדול מדי לדרישות האתר.

חוסר היכולת לגשת לקובץ בצד הלקוח, הביא לא פעם לעומס מיותר על הרשת, בזבוז שטח אחסון וניצול זמן מעבד מיותר בצד השרת.

באמצעות **File API**, ניתן לגשת לקבצים בצד הלקוח, לדוגמא כדי לאפשר למשתמש לבחור האם התמונה תשמר בשרת, עוד לפני שהתמונה הועלתה לשרת, כדי לשמור נתונים לקבצים מקומיים כאשר המשתמש לא מחובר לרשת ועוד.

אפשרויות הטיפול בקבצים:

1. קבלת פרטי הקובץ – שם, סוג, גודל ועוד.
2. קריאת הקובץ והצגתו על הדף

לא ניתן לכתוב לקבצים, כיון שלא ניתן לאפשר זאת מבחינת אבטחה.

אובייקטים לטיפול בקבצים:

File API מציע מספר ממשקים חדשים כדי לעבוד עם הקבצים המקומיים.

1. **File** – מייצג קובץ בודד. מספק מידע **לקריאה בלבד** אודות הקובץ. באמצעותו ניתן לגשת לפרטים כגון שם, גודל, סוג הקובץ וכן גישה ל- file handle.
2. **FileList** – מעין רצף/מערך של אובייקטים מסוג File.
3. **Blob** – מאפשר חיתוך קובץ למערך של בתים.
4. **FileReader** – מאפשר גישה לתוכן הקובץ.

<u>הממשק File</u>

ממשק זה מאפשר לקבל את הקובץ שנבחר בפקד מסוג input file, ולגשת לפרטיו.

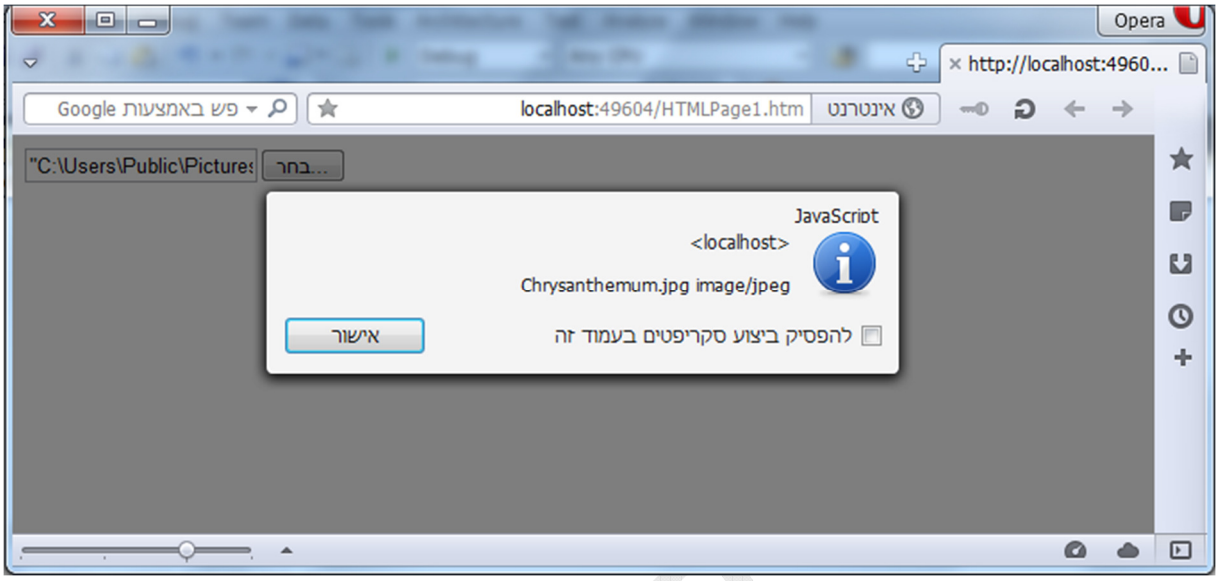
תכונות:

שם תכונה	תאור
name	שם הקובץ
type	סוג הקובץ
size	גודל הקובץ

lastModifiedDate	תאריך עדכון אחרון של הקובץ

דוגמא - בחירת קובץ והצגת פרטיו

בדף זה המשתמש בוחר קובץ ופרטיו מוצגים בתיבת הודעה.



קוד html

```
<input id="File1" type="file" onchange="getFileDetails(event)" />
```

קוד js

```
function getFileDetails(e) {
        if (window.File) {
                var myFile = File1.files;
                var details = myFile[0].name + " " + myFile[0].type
                alert(details);
            }
        else
                alert("לקבצים בגישה תומך אינו הדפדפן");
}
```

הסבר הקוד:

```
function getFileDetails(e)
```

אירוע זה מתרחש בעת בחירת קבצים. הפרמטר e מכיל את האובייקט ששלח לאירוע, פקד בחירת הקבצים.

```
if (window.File)
```

בפקודה זו בודקים האם הדפדפן תומך ב File API, וניתן לגשת לקבצים בצד לקוח. אם הדפדפן אינו תומך תופיע ההודעה הבאה:

```
alert("לקבצים בגישה תומך אינו הדפדפן");
```

```
var myFiles = File1.files;
```

הקובץ שנבחר מתקבל לתוך האובייקט myFiles. אובייקט זה מהווה מערך, כיון שניתן לאפשר לבחור מספר קבצים.

בדוגמא זו, אנו מאפשרים לבחור קובץ אחד, ולכן ניגש תמיד למערך במקום הראשון.

```
var details = myFile[0].name + " " + myFile[0].type;
```

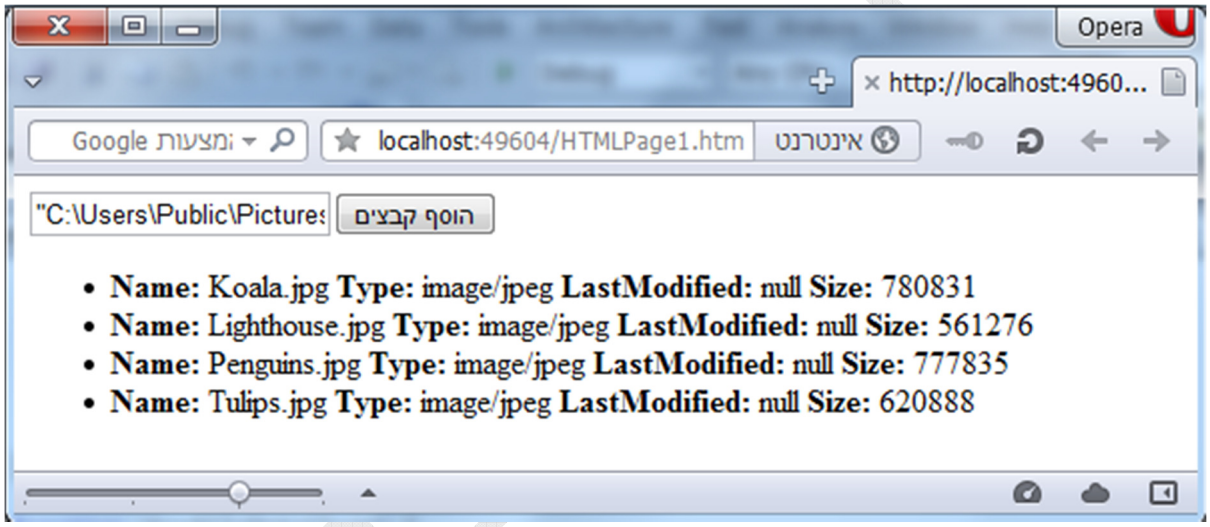
הצבת פרטי הקובץ (שם וסוג) במשתנה.

```
alert(details);
```

הצגת פרטי הקובץ בתיבת הודעה.

דוגמא - בחירת קבצים והצגת פרטיהם ברשימה

בדף זה המשתמש יכול לבחור מספר קבצים בפקד בחירת קבצים, לאחר מכן יוצגו פרטי הקבצים שנבחרו.



קוד html:

```
<input id="BrowserFile" type="file" multiple="multiple"
onchange="showFileDetails(event)" />
<section id="sec1"></section>
```

הסבר הקוד:

multiple="multiple"

תכונה זו מאפשרת בחירה מרובה של קבצים.

onchange="showFileDetails(event)"

בעת בחירת קבצים תקרא הפונקציה showFileDetails.

קוד js:

```
function showFileDetails(e) {
            if (window.File) {
                var myFiles = e.target.files;  // BrowserFile.files;
                var details = "<ul>";
                for (var i = 0, f; f = myFiles[i]; i++) {
                    details += "<li> <b>Name:</b> " + f.name +
                    " <b>Type:</b> " + f.type +
                    " <b>LastModified:</b> " + f.lastModifiedDate +
                    " <b>Size:</b> " + f.size + "</li>";
                }
                details += "</ul>";
                document.getElementById("sec2").innerHTML = details;
            }
              else
                alert("לקבצים בגישה תומך אינו הדפדפן");

        }
```

הסבר הקוד:

function showFileDetails(e)

אירוע זה מתרחש בעת בחירת קבצים. הפרמטר e מכיל את האובייקט ששלח לאירוע, פקד בחירת הקבצים.

if (window.File)

בפקודה זו בודקים האם הדפדפן תומך ב File API, וניתן לגשת לקבצים בצד לקוח. אם הדפדפן אינו תומך תופיע ההודעה הבאה:

alert("לקבצים בגישה תומך אינו הדפדפן");

var myFiles = e.target.files;

בפקודה זו אנו מקבלים את כל הקבצים שנבחרו לתוך אובייקט בשם myFiles. הקבצים נמצאים בתכונה files של הפקד File.

`e.target.files;` - כך ניגשים לקבצים בצורה כללית, מתוך הפקד שהפעיל את האירוע. ניתן לגשת לפקד ישירות, ולהמיר זאת בפקודה הבאה:

var files = BrowserFile.files;

אולם, באופן זה הקוד לא יהיה כללי, ולא יתאים לכל פקד מסוג File, אלא רק לפקד בעל id בשם "`BrowserFile`".

var details = "<ul>";

המשתנה details יכיל טקסט html שאותו נציג כתוצאה. רשימת הפרטים של הקבצים מוצגים כרשימת ul.

for (var i = 0, f; f = myFiles[i]; i++)

מעבר בלולאה על כל הקבצים שנבחרו. f הינו אובייקט שיכיל בכל איטרציה (סיבוב) את הקובץ הנוכחי.

```
details += "<li> <b>Name:</b> " + f.name + " <b>Type:</b> " + f.type + "<b>LastModified:</b> " +
f.lastModifiedDate + " <b>Size:</b> " + f.size + "</li>";
```

השורות הללו משרשרות למחרוזת ה html את התכונות של כל קובץ. כל קובץ מוצג כפריט ברשימה, לכן המחרוזת מוסיפה את האלמנט li.

- f.name - גישה לשם הקובץ. (f - מכיל את הקובץ הנוכחי)
- f.type - גישה לסוג הקובץ.
- f.lastNodifiedDate - גישה לתאריך עדכון אחרון של הקובץ
- f.size - גישה לגודל הקובץ.

```
details += "</ul>";
```

סגירת הרשימה של פרטי הקובץ.

```
document.getElementById("sec1").innerHTML = details;
```

הצגת הרשימה בפקד section הקיים בדף.

<u>הממשק FileReader</u>

בדוגמא למדנו על גישה לאובייקטים מסוג File, שאפשר לנו ללמוד פרטים בסיסיים על הקובץ הנבחר.

ברגע שיש לנו אובייקט מסוג File, אפשר ליצור מופע של אובייקט מסוג FileReader לצורך קריאת תוכן הקובץ באחת מארבע דרכים:

פונקציות:

- **FileReader.readAsBinaryString(File)** – קריאת תוכן הקובץ וקבלת ייצוג בינארי של הקובץ כמחרוזת בשדה result. כל תו במחרוזת מיוצג ע"י מספר בין 0 ל- 255.
- **FileReader.readAsText(File, opt_encoding)** – קריאת תוכן הקובץ כטקסט, לפי הקידוד שמועבר כפרמטר. השדה result יכיל מחרוזת עם התוכן הטקסטואלי של הקובץ. קידוד ברירת המחדל הוא UTF8, והשדה הזה הוא אופציונאלי.
- **FileReader.readAsDataURL(File)** – השדה result יכיל ייצוג של הקובץ כ- data URL (מחרוזת בסגנון: .
- **FileReader.readAsArrayBuffer(File)** – השדה result יכיל את תוכן הקובץ כאובייקט ArrayBuffer.

אירועים:

שם אירוע	תאור
onload	בעת טעינת קובץ

onloadstart	בעת התחלת טעינת הקובץ
onloaded	לאחר שהקובץ נטען
onerror	בעת שגיאה בקריאת הקובץ

דוגמא:

קריאת קובץ טקסט וקובץ תמונה והצגתם על הדף

```
function readFile(e) {
            var files = e.target.files;
            var file = files[0];
            var reader = new FileReader();
            if (file.type.match("text*")) {
                reader.onload = function (e) {
                    p1.innerText = e.target.result;
                }
                reader.readAsText(file);
            }
            if (file.type.match("image*")) {
                reader.onload = function (e) {
                    document.getElementById("sec1").innerHTML = "<img src='" +
e.target.result + "' width='100px' height='100px'/>";
                }
                reader.readAsDataURL(file);
            }
        }
```

הסבר הדוגמא:

קריאת קובץ טקסט-

var files = e.target.files;

קבלת הקבצים/הקובץ שנבחר בפקד

var file = files[0];

הכנסת הקובץ לתוך אובייקטfile

var reader = new FileReader();

הגדרת אובייקט לקריאת תוכן קבצים

if (file.type.match("text*"))

בדיקה האם הקובץ הנבחר הוא קובץ טקסט

reader.onload = function (e)

קישור אירוע לאובייקט, האירוע לא יתרחש כעת, אלא לאחר הפונקציה readAsText

באירוע טעינת הקובץ תתרחש הפונקציה הבאה:

{ p1.innerText = e.target.result;}

הצבת תוכן הטקסט של הקובץ באלמנט פסקה

reader.readAsText(file);

קריאת הקובץ בפועל

קריאת קובץ תמונה-

בדיקה האם הקובץ הנבחר הוא קובץ תמונה

if (file.type.match("image*"))

קישור אירוע לאובייקט, האירוע לא יתרחש כעת ,אלא לאחר הפונקציה readAsDataURL

reader.onload = function (e)

באירוע טעינת הקובץ תתרחש הפונקציה הבאה:

document.getElementById("sec1").innerHTML = "";

הוספת פקד תמונה שיכיל את הנתיב של התמונה שנבחרה

reader.readAsDataURL(file);

פונ 'לקריאת קבצי תמונה

סיכום:

יכולת קריאת קבצים בצד לקוח (באמצעות js) משפרת מאוד את רמת הביצועים של האתר.

תרגיל מס' 6

1. הוסיפי לתרגיל 7 אפשרות בחירה תמונה להצבה כתמונת רקע.

הדרכה:

באירוע הקורא את הקובץ יש להוסיף פקודה זו:

```
document.body.style.backgroundImage = " url('"+e.target.result+"')";
```

2. גלריית תמונות

הדף יאפשר לבחור תמונות ולהציגם בדף בצורת גלריה.

הנחיות:

א. הכיני דף במבנה הבא:

במרכז הדף תמונה גדולה (נניח בגודל 600X600)

בשורה מתחת פקדי תמונה קטנים יותר (נניח בגודל 100X100)

ב. אפשרי למשתמש לבחור תמונות באמצעות פקד file. התמונות הנבחרות יוצגו על פקדי התמונה הקטנים שבדף.

ג. בעת לחיצה על תמונה קטנה, התמונה תוצג בפקד התמונה הגדול שבדף.

ד. שימי לב, אין הגבלה על מספר התמונות הקטנות, הן יווצרו באופן דימני על פי בחירת המשתמש.

ה. הציגי בכל פעם רק 8 תמונות בדף, הוסיפי לחצני הבא וקודם מימין ושמאל לתמונות, והציגי את התמונות הבאות שנבחרו (בלחיצה על הבא, מתווספת התמונה הבאה וכל התמונות הקודמות זזות ימינה, התמונה הראשונה נשמטת).

הדרכה:

שמרי במערך את שמות כל התמונות שנבחרו.

שמרי במשתנים גלובליים (מחוץ לפונקציות) את אינדקס ההתחלה ואינדקס הסיום של התמונות המוצגות.

הציגי בכל פעם את התמונות לפי ערך האינדקסים.

בלחיצה על לחצני הניווט, הזיזי את האינדקסים ושני בהתאמה את התמונות.

קוד ליצירת פקד באופן דינמי:

בדוגמא זו, על הדף קיים פקד section בשם mySec, בו ניצור את הפקד הדינמי. ניצור פקד תמונה, ונוסיף אותו לפקד mySec.

יצירת פקד תמונה והכנסתו לפקד element

```
var element = document.createElement("img");
```

הצבת שם התמונה בפקד

```
element.src = "...";
```

הוספת הפקד

```
mySec.appendChild(element);
```