ΠΕΡΙΕΧΟΜΕΝΑ:

Λειτουργία fullscreen:

- Ένα στοιχείο (element) μπορεί να μπει σε λειτουργία fullscreen με τη μέθοδο:

Μέθοδος	Επεξήγηση
requestFullScreen()	Αίτημα για να μπει το στοιχείο σε fullscreen. Επιστρέφει Promise, που όταν επιλύεται το στοιχείο καταλαμβάνει όλη την οθόνη

Παρατηρήσεις:

- Η μέθοδος πρέπει να είναι μέσα σε event που προκαλείται από τον χρήστη, όπως το να κάνει κλικ σε ένα κουμπί, να πατήσει πλήκτρο, ή με κάποιο touch event.
  - Αλλιώς ο browser θα μπλοκάρει το fullscreen

Έξοδος από Λειτουργία Fullscreen:

- Με την μέθοδο του **document**:

Μέθοδος	Επεξήγηση
exitFullscreen()	Γίνεται έξοδος από τη λειτουργία πλήρους οθόνης

- Χρήσιμο είναι το property του **document**:

Property	Επεξήγηση
fullScreenElement	Το στοιχείο που είναι σε fullscreen (ή null αν κανένα στοιχείο δεν είναι σε fullscreen)

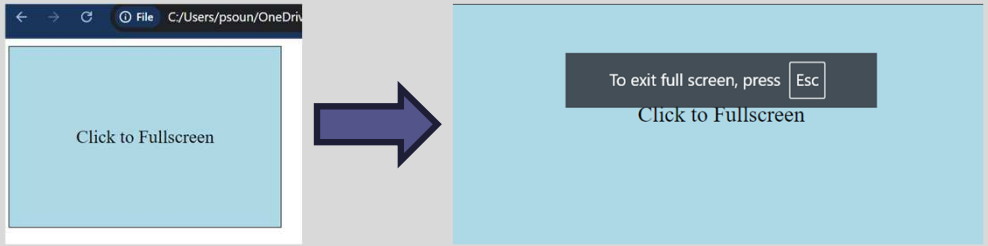
Παράδειγμα 1: fullscreen

```html
<div id="fullscreenDiv">
  Click to Fullscreen
</div>
```

```js
const fullscreenDiv = document.getElementById('fullscreenDiv');

function toggleFullscreen() {
  if (!document.fullscreenElement) {
    fullscreenDiv.requestFullscreen();
  } else {
    document.exitFullscreen();
  }
}

fullscreenDiv.addEventListener('click', toggleFullscreen);
```



Παρατηρήσεις:

- Είναι καλό να έχουμε διαχείριση της περίπτωσης που το Promise απορριφθεί με ένα κατάλληλο μήνυμα λάθους
- Η προσθήκη στυλ όταν μπαίνουμε σε λειτουργία fullscreen μπορεί να γίνει με εναλλαγή κλάσεων, όταν ενεργοποιείται το event fullscreenchange.

Το event fullscreenchange:

- Ενεργοποιείται στο **document**:

event	Επεξήγηση
fullscreenchange	Ενεργοποιείται όταν μπαίνουμε ή βγαίνουμε από λειτουργία πλήρους οθόνης

Παράδειγμα 2: fullscreenchange

```css
#fullscreenDiv {
  width: 300px;
  height: 200px;
  background-color: lightblue;
  border: 1px solid #000;
  text-align: center;
  line-height: 200px;
  font-size: 20px;
}

.fullscreen {
  width: 100%;
  height: 100%;
}
```

```html
<div id="fullscreenDiv">
  Click to Fullscreen
</div>
```

```javascript
const fullscreenDiv = document.getElementById('fullscreenDiv');

// Function to toggle full-screen mode
function toggleFullscreen() {
  if (!document.fullscreenElement) {
    fullscreenDiv.requestFullscreen().catch(err => {
      alert(`Error attempting to enable full-screen mode: ${err.message}
(${err.name})`);
    });
  } else {
    document.exitFullscreen();
  }
}

// Add event listener to the button to trigger full-screen mode
fullscreenDiv.addEventListener('click', toggleFullscreen);

// Event listener to handle fullscreen change
document.addEventListener('fullscreenchange', () => {
  if (document.fullscreenElement) {
    fullscreenDiv.classList.add('fullscreen');
    fullscreenDiv.textContent = 'Exit Fullscreen';
  } else {
    fullscreenDiv.classList.remove('fullscreen');
    fullscreenDiv.textContent = 'Enter Fullscreen';
  }
});
```

Console API:

- Εργαλείο για debugging του κώδικα (αλλά και για logging πληροφοριών, λαθών κ.λπ.)
- Υλοποιείται από το αντικείμενο **console του window**
- Περιέχει τις μεθόδους:

Μέθοδος	Επεξήγηση
log(...args)	Γενικές Πληροφορίες (άσπρο χρώμα)
error(...args)	Μηνύματα λάθους (κόκκινο χρώμα)
warn(...args)	Προειδοποιήσεις (κίτρινο χρώμα)
info(...args)	Πληροφοριακά μηνύματα (άσπρο χρώμα)

- Όλες οι μέθοδοι δέχονται μεταβλητό πλήθος ορισμάτων που στην εκτύπωση θα διαχωρίζονται με κενό

Παράδειγμα 3: console.html

```
// General log
console.log("Debugging message: Script loaded successfully");

// Log an object
let user = { name: "Alice", age: 30 };
console.log(user);

// Log an error
console.error("An error occurred while loading data");

// Log a warning
console.warn("Deprecated function called");

// Log informational message
console.info("User data loaded successfully");
```

```
Debugging message: Script loaded successfully
▶ Object
▶ An error occurred while loading data
▶ Deprecated function called
User data loaded successfully
```

Ομαδοποίηση εκτυπώσεων:

- Μπορούμε να ομαδοποιήσουμε τις εκτυπώσεις ώστε να έχουν μία δενδρική - ιεραρχική μορφή.
- Εκκινούμε ένα γκρουπ εκτυπώσεων με την μέθοδο του **console**:

Μέθοδος	Επεξήγηση
group(name)	Εκκίνηση γκρουπ εκτυπώσεων (με όνομα name)

- Ακολουθούν οι εκτυπώσεις με τις μεθόδους της προηγούμενης διαφάνειας
- Και απαιτείται να τελειώσουμε το γκρουπ εκτυπώσεων με τη μέθοδο:

Μέθοδος	Επεξήγηση
groupEnd()	Τέλος γκρουπ εκτυπώσεων

- Πρόσθετα μπορούμε να επιλέξουμε αν η ομάδα θα εμφανίζεται ανεπτυγμένη ή όχι στην κονσόλα ανάλογα με την μέθοδο που χρησιμοποιούμε για να εκκινήσουμε την ομάδα εκτυπώσεων:

Μέθοδος	Επεξήγηση
group(name)	Εκκίνηση γκρουπ εκτυπώσεων (ανεπτυγμένο)
groupCollapsed(name)	Εκκίνηση γκρουπ εκτυπώσεων (μη ανεπτυγμένο)

- Επίσης μπορούμε να έχουμε εμφωλιασμένα γκρουπ εκτυπώσεων (βλέπε παράδειγμα)

Παράδειγμα 4: console-group.html

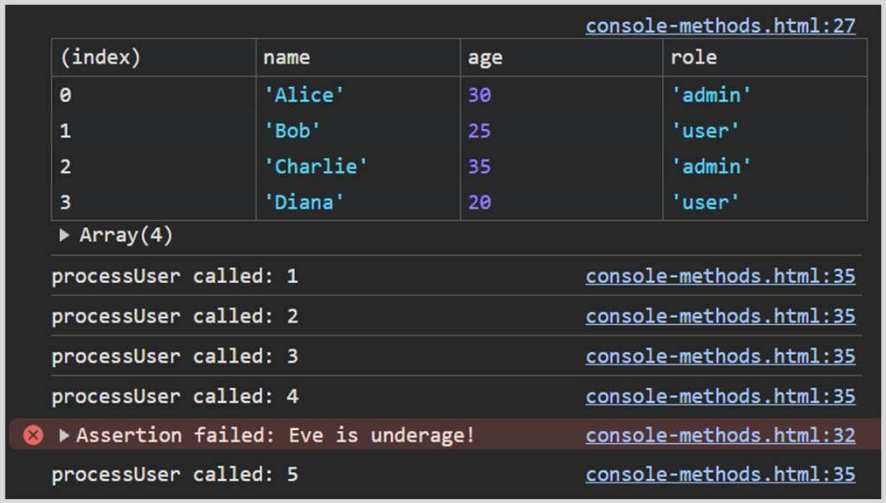
```
console.group("Parent Group");
console.log("Parent log 1");
console.groupCollapsed("Child Group Collapsed");
console.log("Child log 1");
console.log("Child log 2");
console.log("Child log 3");
console.groupEnd();  // Ends the child group
console.group("Child Group");
console.log("Child log 1");
console.log("Child log 2");
console.groupEnd();  // Ends the child group
console.log("Parent log 2");
console.groupEnd();  // Ends the parent group
```

Προχωρημένες Μέθοδοι Κονσόλας:

- Παρέχονται και πιο προχωρημένες μέθοδοι για την αλληλεπίδραση με την κονσόλα, χρήσιμες για debugging, με πρόσθετες μεθόδους του **console**:

Μέθοδος	Επεξήγηση
table(array)	Εκτυπώνει πίνακα. Το όρισμα πρέπει να είναι πίνακας ομοειδών objects
assert(condition, log)	Εκτυπώνει το log, μόνο αν η συνθήκη είναι ψευδής (με το μήνυμα λάθους
count(message)	Εκτυπωνει το μήνυμα και πρόσθετα τυπώνει το πλήθος των φορών που έχει γίνει η εκτύπωσή του

Παράδειγμα 5: console-methods.html




```
// Simulating a list of users
const users = [
  { name: "Alice", age: 30, role: "admin" },
  { name: "Bob", age: 25, role: "user" },
  { name: "Charlie", age: 35, role: "admin" },
  { name: "Diana", age: 20, role: "user" }
];

// 1. Using console.table() to display user data
console.table(users);

// 2. A function to simulate processing user data
function processUser(user) {
  // Using console.assert() to validate user data
  console.assert(user.age > 18, `${user.name} is underage!`);

  // Using console.count() to count how many times this function is called
  console.count("processUser called");
}

// Processing each user
users.forEach(user =>
      processUser(user));

processUser({
        name: "Eve",
        age: 17,
        role: "guest" });
```

Άλλες Μέθοδοι Κονσόλας:

- Παρέχονται και πιο προχωρημένες μέθοδοι για την αλληλεπίδραση με την κονσόλα, χρήσιμες για debugging, με πρόσθετες μεθόδους του **console**:

Μέθοδος	Επεξήγηση
time(label)	Ξεκινάει timer (με αναγνωριστικό label)
timeEnd(label)	Λήγει τον timer με αναγνωριστικό label
dir(object)	Εκτυπώνει μία λίστα με τις ιδιότητες του αντικειμένου (expandable με τα μέλη που είναι objects κ.λπ.)
trace(message)	Εκτυπώνει το μήνυμα και έπειτα την stack trace (τις διαδοχικές μεθόδους που κλήθηκαν μέχρι την κλήση της trace()

Παράδειγμα 6: console-debug.html

```
console.time("Loop Timer");

let sum = 0;
for (let i = 0; i < 1000000; i++) {
  sum += i;
}

console.timeEnd("Loop Timer");

console.dir(document.body);

function firstFunction() {
  secondFunction();
}

function secondFunction() {
  thirdFunction();
}

function thirdFunction() {
  console.trace("Trace Message");
}

firstFunction();
```

```
Loop Timer: 43.90185546875 ms
▶ body
▼ Trace Message
  thirdFunction  @ console-debug.html:38
  secondFunction @ console-debug.html:34
  firstFunction  @ console-debug.html:30
  (anonymous)    @ console-debug.html:41
```