**المصفوفات**

**المصفوفة**

هي عبارة عن حاوية أو قائمة تحتوي على أكثر من قيمة.

**طريقة تعريف Array**

يتم تعريف المصفوفة بنفس الطريقة التي تعرف فيها المتغيرات وتتم كتابة القيم المسندة لها بين قوسين مربعة []، كما هو موضح في المثال التالي:

let colors = ["red", "green", "blue"];

نلاحظ أننا قمنا بكتابة ثلاث قيم بين القوسين []، وفصلنا بين كل قيمة والأخرى بفاصلة ,

**طريقة تعريف Array لاتحتوي على أي قيمة:**

يمكن تعريف مصفوفة فارغة بإسناد قوسين مربعة فارغة للمتغير كما هو موضح في المثال التالي:

myColors = [];

**طباعة قيمة Array**

لطباعة قيمة مصفوفة كل ما علينا فعله هو كتابة اسم المصفوفة بين قوسي أمر الطباعة، كما هو موضح في المثال التالي:

console.log(colors);

عند تنفيذ الأمر السابق ستكون المخرجات على النحو التالي:

['red', 'green', 'blue']

**الوصول لقيمة محددة من قيم المصفوفة:**

يمكن الوصول لكل قيمة من قيم المصفوفة من خلال رقم يدعى index وهو ترتيب القيمة بين القيم، يبدأ ترقيم خانات وقيم المصفوفة من اليسار لليمين، ويبدأ العد من رقم 0 لو أردنا الوصول للقيمة red لطباعتها مثلًا، فسنجد أن رقم index الخاص بها هو 0 لذلك سنستخدم اسم المصفوفة colors مع الرقم 0 للوصول إليها، كما هو موضح على النحو التالي:

console.log(colors[0]);

عند تنفيذ الأمر السابق ستكون المخرجات على النحو التالي:

red

نلاحظ أنها تمت طباعة القيمة red ولو أردنا طباعة أي من القيم الأخرى فكل ما علينا فعله هو كتابة رقم index الخاص بتلك القيمة بدلا من الرقم 0.

**تحديث قيم المصفوفة**

يمكننا تغيير وتحديث قيم المصفوفات باستخدام وتحديد رقم index الخاص بالقيمة المراد تحديثها ثم إسناد قيمة بديلة لها، ويتم ذلك على النحو التالي:

let colors = ['red', 'green', 'blue'];

colors[1] = 'black';

قمنا في السطر الثاني بتحديث القيمة الموجودة في index رقم 1 وهي القيمة green وجعلناها black، وللتحقق من ذلك سنقوم بكتابة أمرين طباعة أحدهما قبل التحديث والآخر بعده:

let colors = ['red', 'green', 'blue'];

console.log(colors[1])

colors[1] = 'black';

console.log(colors[1])

عند تنفيذ الأمر السابق ستكون المخرجات على النحو التالي:

green

black

في السطر الأول تمت طباع القيمة green أي قبل التحديث، وفي السطر الثاني تمت طباع القيمة بعد أن تم التحديث وهي black.

**الخاصية Array Length**

تستخدم length مع المصفوفات، ووظيفتها إعادة عدد عناصر المصفوفة، لتوضيح الفكرة نلاحظ المثال التالي:

let colors = ['red', 'green', 'blue'];

console.log(colors.length);

**المخرجات:**

3

نلاحظ أنه تم طباعة الرقم 3 وذلك لأن المصفوفة تحتوي على 3 قيم وهي red، green، blue.

**استخدام Array Includes**

تُستخدم includes مع المصفوفة وإرفاق قيمة ما، تقوم includes بالتحقق ما إذا كانت تلك القيمة موجودة في المصفوفة أم لا. في حال كانت القيمة موجودة في المصفوفة بالفعل، فسيتم إعادة true أما في حال كانت تلك القيمة غير موجودة في المصفوفة فسيتم إعادة false نلاحظ المثال التالي:

let colors = ['red', 'green', 'blue'];

console.log(colors.includes('green'));

**المخرجات:**

true

في المثال تم استخدام includes المصفوفة colors وإرفاق القيمة green للتحقق ما إذا كانت موجودة بداخل المصفوفة colors أم لا، نلاحظ أن المخرجات true وذلك لأن القيمة green بالفعل موجودة بداخل المصفوفة colors.

الآن سنقوم بإرفاق قيمة غير موجودة بالمصفوفة:

let colors = ['red', 'green', 'blue'];

console.log(colors.includes('black'));

**المخرجات:**

false

نلاحظ أنه تم إعادة false وذلك لأن القيمة black غير موجودة بالمصفوفة colors.

الآن دعنا نقوم بتعديل المصفوفة colors وإضافة اللون black لها:

let colors = ['red', 'green', 'blue', 'black'];

console.log(colors.includes('black'));

**المخرجات:**

true

نلاحظ أنه تم إعادة true وذلك لأن القيمة black الآن موجودة داخل المصفوفة colors.

لاترتبط includes مع أمر الطباعة ()console.log، فهي تقوم بإعادة القيمة true أو false فقط، أي يمكن استخدامها بأي طريقة حسب الاحتياج البرمجي، سنقوم باستخدامها بشكل آخر في المثال التالي:

let colors = ['red', 'green', 'blue', 'black'];

let result = colors.includes('black');

console.log(result);

نلاحظ في السطر الثاني استخدمنا includes وقمنا بحفظ النتيجة التي قام بإعادتها داخل المتغير resultبعدها قمنا بطباعة النتيجة وكانت المخرجات كما هو موضح أدناه:

true

**إضافة عنصر للمصفوفة**

لإضافة عنصر جديد إلى المصفوفة نقوم باستخدام push والتي تقوم بإضافة القيمة الجديدة إلى آخر المصفوفة، ويتم استخدامها بالشكل التالي:

let colors = ["red", "green", "blue"];

colors.push('black');

console.log(colors);

قمنا في السطر الثاني بكتابة اسم المصفوفة colors ثم push وبين القوسين كتبنا القيمة المراد إضافتها وهي blackوالآن عند تنفيذ أمر الطباعة السابق ستكون المخرجات على النحو التالي:

['red', 'green', 'blue', 'black']

كما هو موضح في المخرجات أعلاه تمت إضافة القيمة black إلى المصفوفة colors.

**حذف عنصر من المصفوفة**

لحذف عنصر من المصفوفة نقوم باستخدام ()pop والتي تقوم بحذف آخر عنصر من عنصاصر المصفوفة، ويتم استخدامها بالشكل التالي:

let colors = ["red", "green", "blue"];

colors.pop();

console.log(colors);

في السطر الثاني قمنا بكتابة اسم المصفوفة ثم ()pop، والآن لو قمنا بكتابة أمر طباعة لطباعة المصفوفة كاملة فستكون المخرجات على النحو التالي:

['red', 'green']

قامت()pop بحذف العنصر الأخير blue من المصفوفة colors.

يمكننا الاستفادة من ()pop أيضا بحفظ القيمة المحذوفة وتخزينها في متغير، وذلك بالطريقة التالية:

let blueColor = colors.pop();

console.log(blueColor);

عند تنفيذ أمر الطباعة السابق ستكون المخرجات على النحو التالي:

blue

أصبحت الآن قيمة المتغير blueColor هي العنصر المحذوف blue .

**إضافة عنصر باستخدام Unshift**

تُستخدم unshift مع مصفوفة ما بغرض إضافة عنصر جديد في بداية تلك المصفوفة.

let colors = ['red', 'green', 'blue'];

colors.unshift('black');

console.log(colors);

**المخرجات:**

['black', 'red', 'green', 'blue']

نلاحظ أنه تم إضافة اللون black في بداية المصفوفة، دعنا الآن نقوم بإضافة اللون white أيضًا:

let colors = ['red', 'green', 'blue'];

colors.unshift('black');

colors.unshift('white');

console.log(colors);

**المخرجات:**

['white', 'black', 'red', 'green', 'blue']

نلاحظ أنه تم إضافة القيمة white في بداية المصفوفة قبل القيمة black.

**حذف عنصر باستخدام Shift**

تستخدم shift مع مصفوفة ما وذلك لحذف أول عنصر فيها وعمل return لقيمته، افترض أنك تريد حذف العنصر red من المصفوفة colors كما هو موضح في المثال التالي:

let colors = ['red', 'green', 'blue'];

في هذه الحالة يمكن استخدام shift كما هو موضح في الشكل أدناه:

let colors = ['red', 'green', 'blue'];

colors.shift();

console.log(colors);

**المخرجات:**

['green', 'blue']

نلاحظ أنه قام بحذف العنصر الأول تلقائياً بدون الحاجة إلى تحديد اسم العنصر. الآن دعنا نقوم باستخدام shift مره أُخرى:

let colors = ['red', 'green', 'blue'];

colors.shift();

colors.shift();

console.log(colors);

**المخرجات:**

['blue']

نلاحظ أنه قام بحذف العنصر الأول مرتين، في المره الأولى تم حذف العنصر red وفي المره الثانية كان العنصر green يمثل أول عنصر في المصفوفة وتم حذفه بناءً على ذلك.

لأن shift تقوم بعمل return لقيمة العنصر الذي يتم حذفه، فيمكننا بهذه الحالة الاحتفاظ بقيمته، نلاحظ المثال التالي:

let colors = ['red', 'green', 'blue'];

const redColor = colors.shift();

console.log(colors);

console.log(redColor);

**المخرجات:**

['green', 'blue']

red

نلاحظ أنه تم حذف red من المصفوفة واستقبال قيمته في المتغير redColor.

**استخدام isArray**

تستخدم isArray للتحقق من أن بيانات محددة هي من نوع array، وتكون قيمتها إما true أو false

**طريقة استخدام isArray**

سنقوم بتعريف متغيرين أحدهما يحتوي على قيمة واحدة والآخر عبارة عن مصفوفة، ثم سنقوم بالتحقق من أنواع البيانات باستخدام isArray في أمر طباعة، ويتم ذلك على النحو التالي:

let color = 'black';

let colors = ['red', 'green', 'blue'];

console.log(Array.isArray(color))

console.log(Array.isArray(colors))

ستكون المخرجات على النحو التالي:

false

true

كانت نتيجة أمر الطباعة الأول false لأن المتغير color عبارة عن متغير بينما كانت نتيجة أمر الطباعة الثاني true لأن المتغير colors عبارة عن مصفوفة.