**مقدمة في مفهوم المعاملات & مقدمة في أنواع البيانات Data Types**

1. **مقدمة في مفهوم المعاملات**

**المعاملات الحسابيّة Arithmetic Operators**

يمكنك تنفيذ العمليات الحسابيّة المختلفة باستخدام الصيغة التالية: left op right حيث يمثل الرمز op نوع العملية الرياضية المُراد استخدامها، ويمثل كل من left و right القيمتين (أو المتغيرين أو الثابتين) اللذين سيتم تنفيذ العملية عليهما. سنتحدث في هذا الجزء عن مجموعة العمليات الحسابية التي توفرها لغة JavaScript.

**معامل الجمع**

يقوم المعامل + بجمع الأعداد كما هو موضح في المثال التالي:

let result = 2 + 5;

في هذا المثال قمنا بتنفيذ عملية الجمع باستخدام المعامل + ، وسيتم تخزين ناتج العملية في المتغيّر result.

**معامل الطرح**

يقوم - بطرح الأعداد كما هو موضح في المثال التالي:

let result = 4 - 10;

في هذا المثال قمنا بتنفيذ عملية الطرح باستخدام المعامل - ، وسيتم تخزين ناتج العملية في المتغيّر result.

**معامل الضرب**

يقوم المعامل \* بضرب الأعداد كما هو موضح في المثال التالي:

let result = 2 \* 2;

في المثال أعلاه قمنا بتنفيذ عملية الضرب باستخدام المعامل \* وسيتم تخزين ناتج العملية في المتغيّر result.

**معامل القسمة**

يقوم المعامل / بقسمة الأعداد كما هو موضح في المثال التالي:

let result = 2 / 6;

في هذا المثال قمنا بتنفيذ عملية القسمة باستخدام المعامل / وسيتم تخزين ناتج العملية في المتغيّر result.

عند القسمة على صفر سوف يكون الناتج Infinity.

**معامل باقي القسمة**

يستخدم المعامل ٪ لحساب المتبقي من عملية القسمة، ولتوضيح الفكرة نلاحظ المثال التالي:

let result = 10 % 3;

console.log(result);

نتج عن هذه العملية طباعة العدد ١ وذلك لأن باقي قسمة العدد ١٠ على ٣ يساوي ١.

تعرفنا في هذا الدرس على العمليات الرياضية والحسابية من جمع وطرح وضرب وقسمة كما تعرفنا على عملية حساب باقي القسمة.

--------------------------------------------------------------------------------------------------------------------

**معاملات المقارنة Comparison Operators**

يمكنك تنفيذ عمليات المقارنة المختلفة باستخدام الصيغة التالية: left op right حيث يمثل الرمز op نوع عملية المقارنة المُراد استخدامها ويمثل كل من left و right القيمتين (أو المتغيرين أو الثابتين) اللذان ستتم تنفيذ العملية op عليهما. وستكون نتيجة عمليات المقارنة قيمة من نوع boolean، أي أن ناتج المقارنة سيكون إما true أو false، ويمكنك استخدام عمليات المقارنة في سياقات برمجية مختلفة، على سبيل المثال كشرط مع جملة if كما سنرى لاحقًا، وسنتحدث في هذا الجزء عن مجموعة عمليات المقارنة التي توفرها لغة JavaScript.

**معامل التساوي**

يقوم المعامل == بمقارنة قيمتين، إذا كانت القيم متساوية فإنها تعود بالقيمة true، وفي حال كانت غير متساوية ستعود بالقيمة falseكما هو موضح في المثال التالي:

let result = 6 == 3;

console.log(result);

**المخرجات:**

false

نتج عن هذه العملية طباعة false وذلك لأن 6 لاتساوي 3.

**معامل عدم التساوي**

يقوم المعامل =! بمقارنة قيمتين، إذا كانت القيم غير متساوية فإنها تعود بالقيمة trueوفي حال كانت متساوية فإنها ستعود بالقيمة falseكما هو موضح في المثال التالي:

let result = 8 != 7;

console.log(result);

**المخرجات:**

true

نتج عن هذه العملية طباعة true وذلك لأن 8 لاتساوي 7.

**معامل أكبر من**

يقوم المعامل < بمقارنة قيمتين إذا كانت القيمة الأولى أكبر من القيمة الثانية فإنها تعود بالقيمة trueوفي حال لم تكن أكبر من القيمة الثانية فإنها تعود بالقيمة false، كما هو موضح في المثال التالي:

let result = 2 < 3;

console.log(result);

**المخرجات:**

true

نتج عن هذه العملية طباعة true وذلك لأن 3 أكبر من 2.

**معامل أكبر من أو يساوي**

يقوم المعامل <= بمقارنة قيمتين، إذا كانت القيمة الأولى أكبر من أو تساوي القيمة الثانية فإنها تعود بالقيمة trueوفي حال لم تكن أكبر من أو تساوي القيمة الثانية فإنها ستعود بالقيمة false، كما هو موضح في المثال التالي:

let result = 3 => 3;

console.log(result);

**المخرجات:**

true

نتج عن هذه العملية طباعة true وذلك لأن القيمتان متساويتان، ولن تكون النتيجة false إلا في حال كانت قيمة العدد الأول أصغر من قيمة العدد الثاني.

**معامل أصغر من**

يقوم المعامل > بمقارنة قيمتين، إذا كانت القيمة الأولى أصغر من القيمة الثانية فإنها تعود بالقيمة trueوفي حال لم تكن القيمة الأولى أصغر من القيمة الثانية فإنها ستعود بقيمة false، كما هو موضح في المثال التالي:

let result = 1 > 4;

console.log(result);

**المخرجات:**

false

نتج عن هذه العملية طباعة false وذلك لأن 4 ليست أصغر من 1.

**معامل أصغر من أو يساوي**

يقوم المعامل >= بمقارنة قيمتين، إذا كانت القيمة الأولى أصغر من أو تساوي القيمة الثانية فإنها ستعود بقيمة trueوفي حال لم تكن أصغر من أو تساوي القيمة الثانية فإنها ستعود بقيمة false نلاحظ المثال التالي:

let result = 7 =< 3;

console.log(result);

**المخرجات:**

false

نتج عن هذهِ العملية طباعة false وذلك لأن ٧ ليست أصغر من ولا تساوي ٣.

**معامل تساوي القيمة والنوع**

يقوم المعامل === بمقارنة قيمتين، إذا كانت متساوية من حيث القيمة و نوع البيانات فإنها تعود بقيمة trueفي حال غير ذلك ستعود بقيمة false نلاحظ المثال التالي:

let result = 1 === 1;

console.log(result);

**المخرجات:**

true

نتج عن هذهِ العملية طباعة true وذلك لأن القيمتين متساويتين من حيث القيمة والنوع.

**معامل عدم تساوي القيمة والنوع**

يقوم المعامل ==! بمقارنة قيمتين، إذا كانتا غير متساويتين من حيث القيمة ونوع البيانات فإنها ستعود بقيمة true وستعود بقيمة false في حال تساوت القيمتان نلاحظ المثال التالي:

let result = 1 !== "1";

console.log(result);

**المخرجات:**

true

نتج عن هذه العملية طباعة true وذلك لأن القيمتين غير متساويتين من حيث نوع البيانات. تعرفنا في هذا الدرس معاملات المقارنة، وطرق استخدامها والفرق بينها.

-------------------------------------------------------------------------------------------------------------------

**نظرة على Increment و Decrement**

من العمليات التي يتكرر استخدامها أثناء البرمجة، عملية زيادة واحد على قيمة المتغير الحالية أو إنقاص واحد منها، وتسمى عملية الزيادة في هذهِ الحالة Increment وتُسمى عملية الإنقاص Decrement، وسنتحدث في هذا الجزء عن طريقة استخدامهما وسنتطرق لشرح مفهوم prefix و postfix والفرق بينهما.

**مقدمة في Increment و Decrement**

كما ذكرنا سابقا يستخدم Increment لزيادة واحد على قيمة المتغير ويستخدم Decrement لخصم واحد من قيمة المتغيّر، لتوضيح الفكرة العامة نلاحظ الأسطر التالية:

let number = 5;

number = number + 1;

number = number - 1;

في السطر الثاني قمنا بزيادة واحد على قيمة number، لتصبح القيمة 6 وفي السطر الثالث قمنا بخصم واحد من قيمة number لتصبح 5. توفر لغة JavaScript طريقة مُختصرة لتنفيذ كلتا العمليتين السابقتين، وذلك من خلال استخدام معامل الزيادة ++ لزيادة واحد على قيمة المتغير وهذا هو المقصود بمفهوم Increment، ومعامل الخصم -- لخصم واحد من قيمة المتغير وهذا هو المقصود بمفهوم Decrement، لتوضيح الفكرة، دعنا نقوم بإعادة كتابة المثال السابق بالطريقة المُختصرة في المثال التالي:

let number = 5;

number++; // increment

number--; // decrement

كما نلاحظ في المثال السابق قمنا باستخدام الطريقة المختصره لكل من عملية الزيادة والخصم وهذا باستخدام Increment و Decrement.

لايمكن استخدام Increment و Decrement إلا على المتغيرات وعند محاولة استخدامها على الثوابت أو على القيم مباشرة - على سبيل المثال 3++ - فإن البرنامج سيظهر لنا رسالة خطأ.

**مفهوم prefix و postfix**

يمكنك وضع علامة (Increment أو Decrement) قبل المتغيّر أو بعده و هذا سوف يؤثر على توقيت تنفيذ العملية، فمثلًا عند وضع العلامة بعد المتغيّر، فإنها تُسمى postfix، وعند وضعها قبل المتغيّر تُسمى prefix وحتى نفهم الفرق بين prefix وpostfix، نلاحظ المثال التالي:

let number = 2;

let result = number++ + 4; // postfix

console.log(result);

console.log(number);

في المثال السابق استخدمنا أسلوب postfix لأن ++ أتت بعد number، وهذا يعني أن قيمة number ستزداد بعد تنفيذ العملية الحالية، أي سيتم أخذ القيمة الحالية للمتغير number وهي 2 ثم سيتم جمعها مع القيمة 4 ويتم تخزين النتيجة في result، وبعد الانتهاء من تنفيذ هذه العملية سيتم زيادة واحد على number أي أن postfix تعني قم بالزيادة بعد تنفيذ العملية ليصبح الناتج على النحو التالي:

6

3

أما بالنسبة لطريقة prefix، فهي عكس العملية السابقة، وهذا يعني أن قيمة المتغير ستزداد قبل تنفيذ العملية الحالية ولتوضيح الفكرة دعنا نقوم بإعادة المثال السابق ولكن بطريقة prefix وعلى النحو الموضح في الأسطر التالية:

let number = 2;

let result = ++number + 4; // prefix

console.log(result);

console.log(number);

في هذه الحالة ستتم زيادة قيمة number لتصبح 3 وبعد ذلك يتم تنفيذ العملية عليه بجمعه مع 4 لتصبح قيمة result هي 7 وستكون المخرجات على النحو التالي:

7

3

بالنسبة لعملية الخصم -- فينطبق عليها ما ذكرناه عن ++ ، أي أنها تكون على هيئة prefix و postfix، وتختلف في أن العملية هنا عملية خصم،postfix تخصم واحد من المتغير بعد تنفيذ العملية عليه و prefix تخصم واحد من المتغير قبل تنفيذ العملية عليه.

تعرفنا في هذا الدرس على عملية Increment لزيادة واحد على قيمة المتغير و Decrement لخصم واحد من قيمة المتغير كما تعرفنا على مفهوم prefix و postfix و الفرق بينهما.

--------------------------------------------------------------------------------------------------------------------

**المعاملات المنطقية Logical Operators**

هناك ثلاث عمليات منطقية، اثنتان منهما تكتب بالصيغة التالية: left op right أما الثالثة فتكتب بالصيغة التالية: op right، حيث يمثل الرمز op نوع العملية المنطقية المُراد استخدامها ويمثل كل من left و right القيمتين (أو المتغيرين أو الثابتين) اللذين سيتم تنفيذ العملية op عليهما وستكون نتيجة العمليات المنطقية قيمة من نوع boolean، أي أنها ستكون إما true أو false ويمكنك استخدامها في سياقات برمجية مختلفة على سبيل المثال كشرط مع جملة if كما سنرى لاحقًا، سنتحدث في هذا الجزء عن المعاملات المنطقية التي توفرها لغة JavaScript.

**المعامل AND**

المعامل AND يستخدم علامة && والتي تعود بالقيمة true فقط إذا كانت كلتا القيمتين صحيحة، وتعود بالقيمة false في باقي الحالات، لفهم الفكرة نلاحظ المثال التالي:

let notRaining = true;

let notFoggy = true;

let isSunny = notRaining && notFoggy;

console.log(isSunny);

**المخرجات:**

true

في المثال السابق تمت طباعة true لأن كلتا القيمتين صحيحتين، ولو كانت إحداهما أو كلتاهما غير صحيحة فسيكون ناتج الطباعة false.

**المعامل OR**

عملية OR تستخدم علامة || و التي تعود بـالقيمة true إذا كانت إحدى القيمتين صحيحة، وتعود بالقيمة false فقط إذا كانت كلتا القيمتين غير صحيحة ولفهم الفكرة نلاحظ المثال التالي:

let isRaining = true;

let isFoggy = false;

let notSunny = isRaining || isFoggy;

console.log(notSunny);

**المخرجات:**

true

في المثال السابق تمت طباعة true لأن إحدى القيمتين صحيحة، لو كانت كلتا القيمتين غير صحيحة سوف تكون نتيجة الطباعة false.

**المعامل NOT**

عملية NOT تستخدم علامة ! و التي تقوم بعكس قيمة Boolean. مما يعني أنها إن كانت القيمة true سوف تصبح false بعد استخدام معامل NOT. لفهم الفكرة نلاحظ المثال التالي:

let isStudent = true;

console.log(!isStudent);

**المخرجات:**

false

نتيجة الطباعة false لأن المعامل NOT قام بعكس القيمة الأصلية للمتغيّر isStudent.

**الفرق بين And و Or و Not**

الفرق بين العمليات المنطقية الثلاث:

**:And** تكون النتيجة true في حالة واحدة فقط، وهي إذا كانت القيمتين left و right تساويان true.

**:Or** تكون النتيجة false في حالة واحدة فقط، وهي إذا كانت القيمتين left و right تساويان false.

**Not :** يقوم بعكس قيمة right في حال كانت right تساوي true فستكون النتيجة هي false، والعكس صحيح.

يوضح المثال التالي استخدام العمليات السابقة واختلاف النتائج:

let first = true, second = false;

let andResult = first && second; // false

let orResult = first || seocnd; // true

let notResult = !first // false

console.log(andResult);

console.log(orResult);

console.log(notResult);

**المخرجات:**

false

true

false

تعرفنا في هذا الدرس على المعاملات المنطقيّة، وطرق استخدامها والفرق بينها وكيفية توظيفها في البرنامج والاستفادة منها.

1. **مقدمة في أنواع البيانات Data Types**

في هذا الدرس سنسلط الضوء على أنواع البيانات Data Types وعلى كيفية استخدامها وتوظيفها برمجيًا، كما سنتعرف على أنواع البيانات الرئيسية مثل النصوص والأعداد وغيرها من أنواع البيانات التي سوف تحتاج للتعامل معها في أي مشروع برمجي.

**أنواع**

يقصد بأنواع البيانات أنواع القيم التي تُحفظ وتخزن في المتغيرات والثوابت، تدعم لغة JavaScript عددًا من أنواع البيانات، ويوضح الجدول التالي بعض من هذه الأنواع:

* **النوع String:** يمثل هذا النوع البيانات النصيّة مثل characters والنصوص strings.
* **النوع Number:** يمثل هذا النوع البيانات الرقميّة مثل الأعداد الصحيحة integers والأعداد ذات النقطة العشرية double و float.
* **النوع Boolean:** يمثل هذا النوع إحدى القيمتين إما true أو false.
* **النوع null:** يدل هذا النوع على عدم وجود قيمة.
* **النوع undefined:** يدل هذا النوع على أن القيمة غير معرفة.

سيتم فيما يلي مناقشة كل نوع من هذه الأنواع بشكل مفصل.

**نظرة على النوع String**

النص string هو نوع من أنواع البيانات يتكوّن من مجموعة أحرف ورموز، ويكتب بين علامات التنصيص كما هو موضح في المثال التالي:

let name = "Ahmed"

في المثال أعلاه مثل القيمة Ahmed البيانات من النوع String، نلاحظ أن البيانات من نوع String تُوضع بين علامتي تنصيص.

لإنشاء String في JavaScript يمكن استخدام علامات التنصيص المزدوجة " " أو علامات التنصيص المفرد ( ‘ ) أو حتى علامة **backtick** ( `) ويسمى النص المحاط backtick (علامات التنصيص المائلة) بـ **Template literals** ويستخدم الـ Template literals لكتابة النص متعدد الاسطر او لكتابة كود JavaScript داخل النص او في Expressions كما في المثال التالي:

Let language = " JavaScript ";

Let message = `Welcome

to

${ language }` ;

Console.log(message)

**نظرة على النوع Number**

النوع number وهو نوع من أنواع البيانات يُمثل القيم العدديّة وقد تكون قيمته موجبة، أو سالبة، أو صفر، نلاحظ المثال التالي:

let age = 20

في المثال السابق قمنا بإنشاء متغير وأسندنا إليه القيمة 20 وهي قيمة عددية من نوع number.

**نظرة على النوع Boolean**

يتميز النوع boolean عن غيره من أنواع البيانات أنه يحتوي على قيمتين فقط، إما أن تكون قيمته true أو false، لتوضيح الفكرة نلاحظ المثال التالي:

let isStudent = true

let isEmployee = false

في المثال السابق قمنا بإنشاء متغيرين من نوع boolean الأول باسم isStudent ويحتفظ بقيمة true، والثاني باسم isEmployee ويحتفظ بقيمة false

**نظرة على النوع Null**

يشير النوع null إلى أن المتغير يعتبر فارغ أو يحتوي على قيمة غير معروفة، لتوضيح الفكرة نلاحظ المثال التالي:

let homeAddress = null

في المثال السابق قمنا بإنشاء متغير باسم homeAddress وتم إسناد القيمة null إليه لتوضيح أن القيمة غير معروفة.

**نظرة على النوع undefined**

عند تعريف متغير ما دون إسناد قيمة إليه، سيكون نوع قيمته undefined، نلاحظ المثال التالي:

let phoneNumber

console.log(phoneNumber)

**المخرجات:**

undefined

عند طباعة قيمة المتغير phoneNumber كانت نتيجة الطباعة undefined وذلك لأنه لم يتم إسناد أي قيمة له.

**القيمة Nan (Not a number)**

القمية تظهر في حال كانت العميلة الحسابية غير ممكنه مثل :

let value = 2;

console.log(value \* ‘JavaScript’)

**استخدام typeof لمعرفة نوع البيانات**

لمعرفة نوع القيمة المسندة للمتغيّر نستخدم typeof متبوعة باسم المتغيّر نلاحظ المثال التالي:

let name = "Khalid"

let age = 18;

let isStudent = true

console.log(typeof name)

console.log(typeof age)

console.log(typeof isStudent)

في المثال السابق قمنا باستخدام typeof على ثلاث متغيرات مختلفه:

* string
* number
* boolean

نلاحظ أنها تمت طباعة string وذلك لأن قيمة المتغير name كانت نص Khalid، وتمت طباعة number لأن قيمة المتغير age كانت العدد 18 وأخيرًا تمت طباعة boolean لأن قيمة المتغير isStudent كانت true.

تعرفنا في هذا الدرس على مفهوم أنواع البيانات والذي يشير إلى أنواع القيم التي تُحفظ في المتغيرات والثوابت، ثم تعرفنا على كيفية استخدامها وتوظيفها برمجيًا وطريقة استخدام typeof لمعرفة نوع بيانات المتغير أو الثابت.