**Mise en application (plusieurs pages)**

Variables et portée

1)

const entree = 8; const valeurControle = entree / 2 + 3; function multiplicateur(nombre, phase) { const valeur = nombre \* valeurControle + phase; console.log(valeur);

};

Quelles sont les variables qui possèdent uniquement une portée dans la fonction "multiplicateur" ?

1. valeur, nombre, controlVal, phase
2. entree, valeurControle
3. nombre, phase, valeur

const entree = 8;

const valeurControle = entree / 2 + 3;

function multiplicateur(nombre, phase) { 🡪 nombre et phase ne sont fait que pour garder des valeurs au sein de ma fonction et pas ailleurs

const valeur = nombre \* valeurControle +phase; console.log(valeur);

}; 🡪 en surligné c’est mon bloc multiplicateur

2)

let bonjour = 'Salut'; const auRevoir = 'Bye'; function direPolitesse() { bonjour = 'Ne vous approchez pas de mes enfants!'; console.log(bonjour); console.log(auRevoir);

}; direPolitesse();

Qu'est ce qui sera affiché dans la console ?

1. Salut

Bye

1. Ne vous approchez pas de mes enfants !

Bye

1. Ne vous approchez pas de mes enfants ! ReferenceError: variable not defined.

let bonjour = 'Salut';

const auRevoir = 'Bye';

function direPolitesse() {

bonjour = 'Ne vous approchez pas de mes enfants!';

console.log(bonjour);

console.log(auRevoir);

}; direPolitesse();

3)

function roadTrip() { const destination = 'Crater Lake, Oregon'; const snacks = 'trail mix'; const supplies = 'sleeping bags';

}; console.log(`Next stop: ${destination}`);

Qu'est ce qui sera affiché dans la console ?

1. Next stop: destination
2. Next stop: Crater Lake, Oregon
3. Uncaught ReferenceError: destination is not defined.

function roadTrip() {

const destination = 'Crater Lake, Oregon'; const snacks = 'trail mix';

const supplies = 'sleeping bags';

};

console.log(`Next stop: ${destination}`);

c’est le C car mes variables existe seulement à l’intérieur de la fonction roadtrip. Le console.log est en dehors de la fonction donc je n’ai pas de variable définie

4)

Laquelle de ces définitions définit au mieux une variable de portée de bloc ?

1. Une variable qui est accessible dans tout le programme.
2. Une variable qui est accessible à l'extérieur d'un bloc.
3. Une variable qui est accessible à l'intérieur d'une fonction.
4. Une variable qui est définie dans un bloc et seulement accessible à l'intérieur de ce bloc.

5)

const entree = 8; const valeurControle = entree / 2 + 3; const multiplicateur = (nombre, phase) => { const valeur = nombre \* valeurControle + phase; console.log(valeur);

};

Quelles sont les variables qui ont une portée globale ?

1. ~~Il y en a trois: entree, valeurControle et multiplicateur.~~
2. Il y en a quatre: entree, valeurControle, valeur et multiplicateur.
3. Il y en a deux: entree, valeurControle.
4. Il y en a trois: entree, valeurControle, valeur.

const entree = 8;

const valeurControle = entree / 2 + 3;

const multiplicateur = (nombre, phase) => {

const valeur = nombre \* valeurControle +phase;

console.log(valeur);

};

En surligné le bloc de la fonction. Les variables qui ont une portée GLOBALES sont donc celles qui ne sont PAS dans le bloc. Donc pas la réponse a mais bien la réponse c

6)

Laquelle de ces définitions définit au mieux une variable de portée globale ?

1. Une variable qui est aussi un paramètre.
2. Une variable qui est accessible depuis n'importe où dans le programme.
3. Une variable qui est accessible dans un bloc, et seulement dans ce bloc.
4. Une variable définie dans une fonction.

7)

Quelle est la meilleure option : définir les variables de façon globale ou définir les variables avec une portée de bloc et pourquoi ?

En définissant avec une portée bloc, on peut d’une part réutiliser le nom de la variable à plusieurs reprises sans que cela soit la même variable, et d’autre part on évite d’en modifier la valeur par accident ou de la réutiliser en gardant une valeur mais en croyant l’avoir modifiée

1. Définir les variables avec une portée de bloc car elles peuvent être utilisées de façon globale également.
2. Définir les variables avec une portée de bloc car les variables définies dans l'espace global peuvent causer un comportement inattendu dans notre code.
3. Définir les variables dans l'espace global et les définir avec une portée de bloc est équivalent.
4. Définir les variables dans l'espace global car les variables définies avec une portée de bloc sont restrictives et causent des conflits avec les variables définies dans l'espace global.