Zadanie 2

Zalety JavaScript:

* Uniwersalność (działa po stronie klienta i serwera).
* Łatwość nauki.
* Duża społeczność i wsparcie.
* Szybkość wykonania w przeglądarkach.
* Obsługiwany przez główne przeglądarki.
* Asynchroniczność.

Wady JavaScript:

* Różnice w przeglądarkach.
* Brak typizacji.
* Zagrożenia bezpieczeństwa.
* Potencjalne problemy z wydajnością.
* Skomplikowane zarządzanie stanem w większych aplikacjach.
* Brak dostępu do niektórych zasobów systemowych.

Zadanie 1

Komentowanie kodu:

**Python**:

1. Komentarze jednoliniowe rozpoczynają się znakiem #.
2. Komentarze wieloliniowe można utworzyć za pomocą potrójnych cudzysłowów (''' lub """).

**PHP**:

1. Komentarze jednoliniowe rozpoczynają się znakiem // lub #.
2. Komentarze wieloliniowe można utworzyć za pomocą /\* ... \*/.

**JavaScript**:

1. Komentarze jednoliniowe rozpoczynają się znakiem //.
2. Komentarze wieloliniowe można utworzyć za pomocą /\* ... \*/.

**Instrukcje warunkowe:**

**Python**:

Instrukcje warunkowe definiuje się za pomocą słów kluczowych if, elif i else.

**PHP**:

Instrukcje warunkowe definiuje się za pomocą słów kluczowych if, elseif i else.

**JavaScript**:

Instrukcje warunkowe definiuje się za pomocą słów kluczowych if, else if i else.

**Wypisywanie komunikatu "hello world":**

**Python**:

print("Hello, World!")

**PHP**:

echo "Hello, World!";

**JavaScript**:

console.log("Hello, World!");

**zadanie 3**

**Tag <script> w sekcji <head> lub <body>:**

Jednym z najczęstszych sposobów dodawania JavaScript do dokumentów HTML jest użycie tagu <script>. Możesz umieścić ten tag w sekcji <head> lub na końcu sekcji <body>:

<!DOCTYPE html>

<html>

<head>

    <title>Strona z JavaScript</title>

    <script>

        function sayHello() {

            alert("Hello, World!");

        }

    </script>

</head>

<body>

    <button onclick="sayHello()">Kliknij mnie</button>

</body>

</html>

**Link do zewnętrznego pliku JavaScript:**

Możesz również umieścić kod JavaScript w zewnętrznym pliku i odwołać się do niego w dokumencie HTML za pomocą tagu <script>.

<!DOCTYPE html>

<html>

<head>

    <title>Strona z JavaScript</title>

    <script src="moj-skrypt.js"></script>

</head>

<body>

    <button onclick="sayHello()">Kliknij mnie</button>

</body>

</html>

Zadanie 4

1. alert():
   1. alert() służy do wyświetlania komunikatu w oknie dialogowym przeglądarki.
2. console.log():
   1. console.log() służy do wypisywania komunikatów lub danych w konsoli przeglądarki.
3. confirm():
   1. confirm() wyświetla okno dialogowe z przyciskami OK i Anuluj. Można go używać do potwierdzania działań użytkownika.
4. prompt():
   1. prompt() wyświetla okno dialogowe z polem tekstowym, które pozwala użytkownikowi wprowadzić dane.

Zadanie 5

1. window.alert():
2. Ta metoda wyświetla okno dialogowe z określonym komunikatem i przyciskiem "OK". Nie zwraca żadnej wartości.
3. window.prompt():
4. Ta metoda wyświetla okno dialogowe z polem tekstowym, komunikatem i przyciskami "OK" oraz "Anuluj". Zwraca wartość wprowadzoną przez użytkownika lub null, jeśli kliknie "Anuluj".
5. window.confirm():
6. Ta metoda wyświetla okno dialogowe z komunikatem oraz przyciskami "OK" i "Anuluj". Zwraca true, jeśli użytkownik kliknie "OK", lub false, jeśli kliknie "Anuluj"

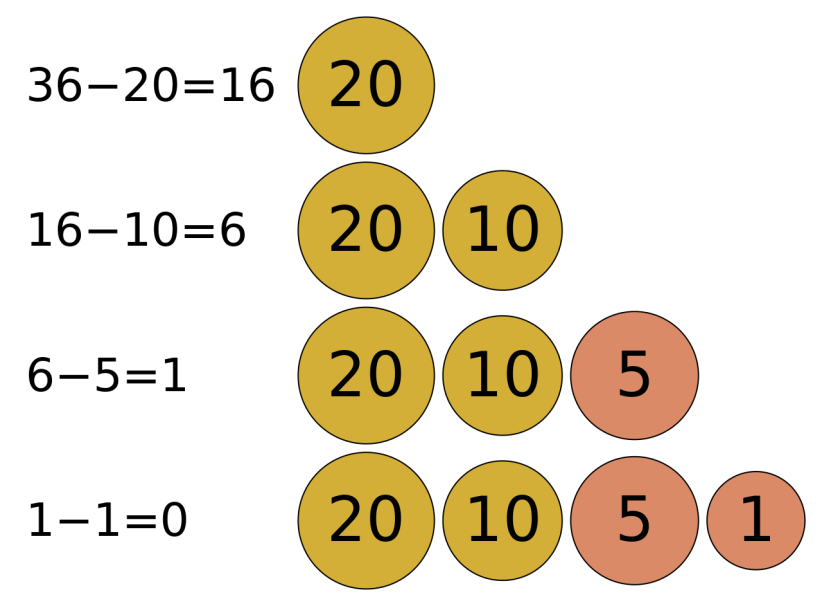
Zadanie 6

JavaScript jest językiem programowania, który jest wrażliwy na wielkość liter, co oznacza, że rozróżnia wielkie litery od małych liter. Oto dlaczego wielkość liter ma znaczenie w JS:

1. Rozróżnianie zmiennych i identyfikatorów: W JavaScript, nazwy zmiennych, funkcji i innych identyfikatorów muszą być zapisywane dokładnie w tej samej formie, w jakiej zostały zadeklarowane. Oznacza to, że zmienna o nazwie "myVariable" jest różna od zmiennej o nazwie "myvariable".

2. Zwracane wartości funkcji: Jeśli funkcja zostanie zadeklarowana z wielką literą, musi być wywoływana z dokładnie tą samą wielkością liter. Jeśli nie zostanie spełniony ten warunek, może to prowadzić do błędów lub nieoczekiwanych wyników.

3. Operacje na ciągach znaków: W operacjach na ciągach znaków wielkość liter jest uwzględniana. Na przykład, "Hello" i "hello" są traktowane jako różne ciągi znaków.

Zad 7

Algorytm zachłanny (greedy approach) to strategia rozwiązywania problemów w informatyce i matematyce, która polega na podejmowaniu lokalnie najlepszych decyzji w nadziei na osiągnięcie optymalnego rozwiązania globalnego. W każdym kroku algorytmu zachłannego wybierana jest opcja, która w danym momencie wydaje się być najlepsza, niezależnie od tego, jakie konsekwencje mogą wynikać z wyboru tej opcji w przyszłości.

Główne cechy algorytmu zachłannego to:

1. Lokalna optymalizacja: Algorytm zachłanny dokonuje wyborów na podstawie kryterium optymalizacji w danym punkcie, bez analizy globalnych skutków tych wyborów.
2. Brak cofania się: Po dokonaniu wyboru, algorytm niecofa się i nie rozważa alternatywnych ścieżek. Kontynuuje on postępowanie na podstawie swoich lokalnych wyborów.
3. Szybkość działania: Algorytmy zachłanne są często szybkie i wydajne, ponieważ nie wymagają analizy wszystkich możliwych opcji.

Przykłady problemów, które mogą być rozwiązywane za pomocą algorytmu zachłannego, to m.in.:

1. Problem plecakowy (knapsack problem): Wybieranie przedmiotów o określonych wagach i wartościach w taki sposób, aby zmieścić je w plecaku o określonej pojemności i maksymalizować łączną wartość.
2. Algorytmy kompresji danych: Wybieranie najczęściej występujących symboli lub ciągów symboli, aby skompresować dane.
3. Minimalne drzewo rozpinające (minimum spanning tree): Wybieranie krawędzi o minimalnych wagach, aby połączyć wszystkie wierzchołki w grafie.
4. Algorytmy wyboru tras w sieciach komunikacyjnych: Wybieranie najkrótszej lub najszybszej trasy pomiędzy dwoma punktami.

Zadanie 8

Jeśli (warunek jest prawdziwy):

**Wykonaj to działanie**

W przeciwnym razie:

**Wykonaj inne działanie**

If (a=true){

Console.log(„Tak”);

} else{

Console.log(„Nie”);}