Rapport

Javascript 1 - Inlämningsuppgift 3

Fullständigt namn: Per Zackrisson
Personnummer: 950722-0094

Uppgift 1.

Först skapas en konstruktor som jag kallar *Person*. Denna funktion innehåller flertal attribut. Dessa attribut är beståndsdelar som kommer att innehålla String -värden.
I konstruktorn för *Person* skapas i sin tur objektet *Adress* (som har en fristående konstruktor) som då både är ett attribut till objektet *Person* samt ett eget objekt med egna attribut. Objektet *Adress* får sina attribut från konstruktorn till *Person* då jag valde att sammanfoga dessa objekt på det sättet.
Metoden *print* förekommer som en knuten metod till objektet Person och returnerar en sträng som innehåller de väsentliga delarna av *Person* på samma sätt som uppgifts-exemplet visar.
I denna metod kan man även se att när man anropar ett objekt inom ett objekt så måste det punkt-noteras för att förtydliga sambandet och dimensionen där objektet existerar.

Exempel från kod: `this.Adress.gNamn`

I nästkommande konstruktor *(kronologisk ordning i koden)* skapas objektet *Adress*. Denna anropas i nuläget enbart från konstruktorn till *Person* men kan i framtiden användas för att skapa fristående *Adress* -objekt som då kan kopplas till 0..∞ förhållanden med andra *Person* -objekt till skillnaden från i nuläget då *Person* och *Adress* har ett 1..1 förhållande.

I funktionen *Constructor* finns all användardata som behövs för att utföra uppgiften. Här skapas objektet *Person* med hjälp utav *Person* -konstruktorn som också kräver den nödvändiga datan för att skapa objektet *Adress*, eftersom att *Adress* blir ett attribut till *Person*.
Print(person) åkallas sedan för att datan ska hamna i rätt funktion.

Eftersom koden skrivs ut på samma sätt varje gång den exekveras så får programmet ett statiskt händelseförlopp som också deklareras i *Constructor* -funktionen.

Sist i koden så görs ett anrop till *Constructor* för att starta exekveringen av koden. Efter att koden körts igenom en gång så avslutas programmet och returnerar printen.

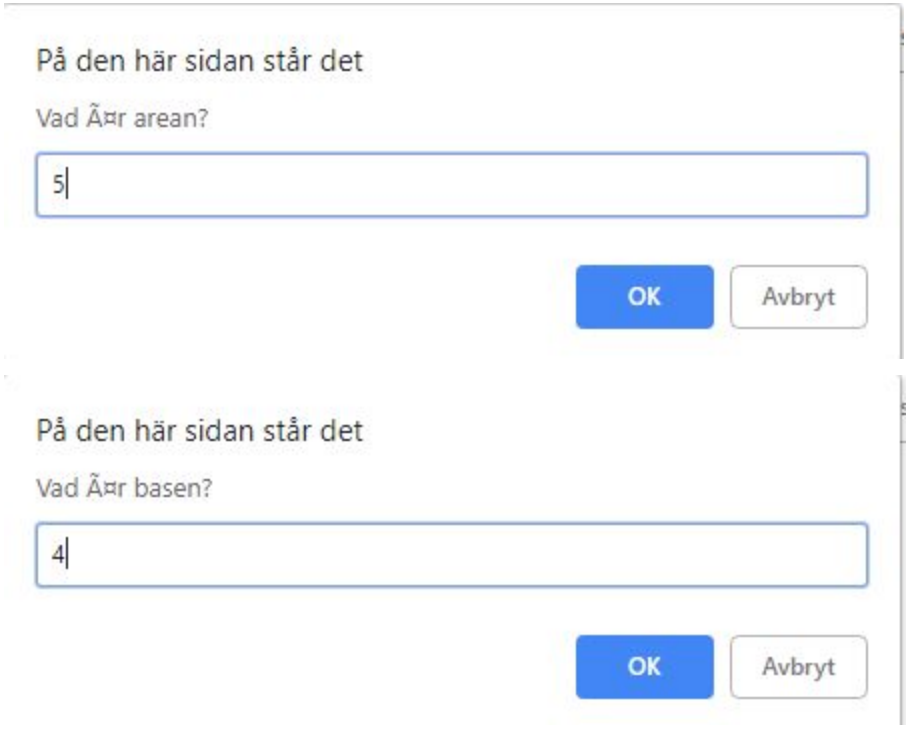
Uppgift 2.

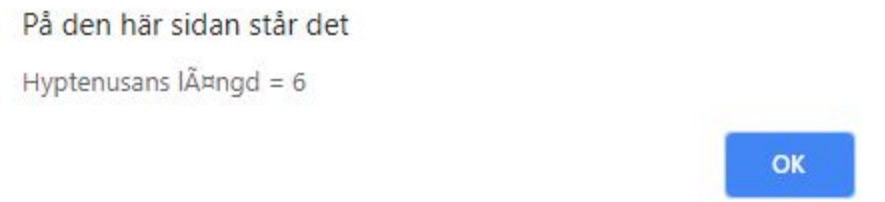
För att skapa en mer dynamisk känsla i koden så används två prompt´s som frågar efter area och basen på triangeln så får jag ett grundvärde till variablerna : *a* samt *b*.

Sen deklareras funktion *calc* som utför beräkningen i två steg:

1. Deklarerar en variabel, *x*, som representerar summan av produkterna från variablerna a i kvadrat samt b i kvadrat.
2. Sedan så retuneras ett avrunda tal av roten ur *x*.

Detta skrivs sedan ut i en alert där funktionen *calc(a,b)* åkallas på hemsidan:





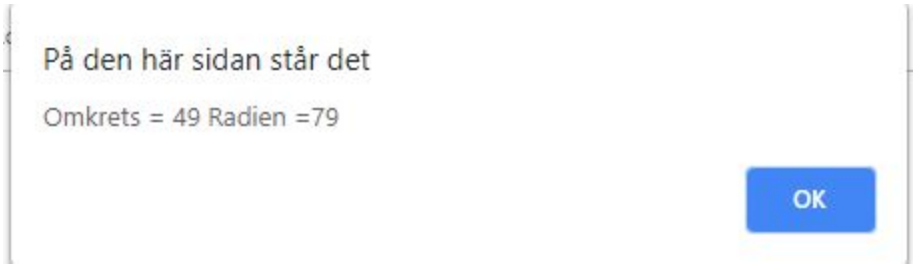
Uppgift 3.

Det som är annorlunda med uppgift 3 förhållandevis till uppgift 2 är att i uppgift 3 så skapas det två funktioner som båda använder värdet ur variabel r och gör två olika uträkningar.

Funktioner

1. Calc räknar ut basen enligt formeln pi i kvadrat multiplicerat med radien: `Math.pow(Math.PI,2) * r;`
2. Calc2 räknar ut arean av cirkeln med hjälp av formeln pi multiplicerat med radien i kvadrat: `Math.PI * Math.pow(r,2);`

Dessa funktioner åkallas i *alert* som då ger utskriften.



Uppgift 4

Denna uträkning är kodmässigt väldigt lik föregående uppgifter 2 & 3. Eftersom den tar ett värde och bearbetar värdet genom två funktioner. Det är bara den matematiska uträkningen som är svårare att skriva så förklarar hur jag tänkt den.

I den första funktionen som räknar ut volymen så räknar *Math.pow* metoden ut PI gånger sig själv 4 gånger för att sedan ta produkten av radien gånger 3. Detta divideras sedan med 3.

Uppgift 5

Till att börja med har jag skapat sex stycken variabler och en tom array. Dessa 6 int -variabler är initialt 0 eftersom jag vill behålla dem tomma för tillfället.

Första funktionen *randomDice()* innehåller en iteration som skapar ett nytt element varje gång i(index) är under 1000. Eftersom iterationen är anvisad till en array skapas ett nytt element i arrayen varje gång loopen körs. Detta görs 1000 gånger innan iterationen tar slut.
// Se mer detaljerad förklaring av iteration i nästkommande uppgift

Math.floor används till att Math.random inte ska gå över värdet 6.

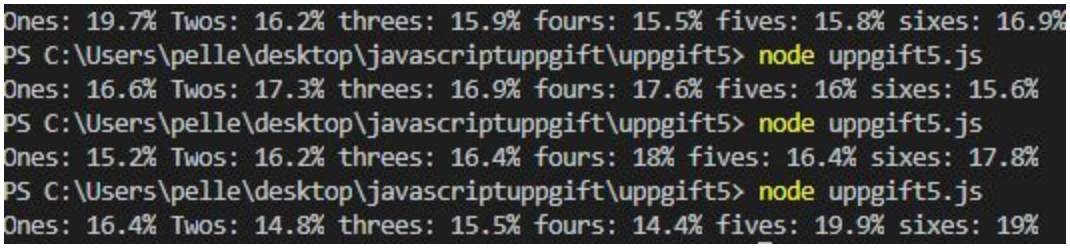
För att Math.random ska hålla sig inom en viss numerisk ram så används Math.random * 6 så randomiseringen stannar inom värden 1-6. Efter detta adderas +1 eftersom jag vill att Math.random ska välja tal mellan *1-6* och inte *0-5*. +1 eliminerar förekomsten av *0*:or.

Jag har sedan skapat en ny funktion som kallas *print();*
Även här måste iterationer ske då jag vill att loopen ska gå igenom samtliga positioner i arrayen *arr*[i].
I loopen har det skapats if och else “statements” som läser av förekomsten utav antal 1:or, 2:or osv.

Här läggs alltså 1 till på variablerna (*one,two, osv*) om Integern i arrayen med positionen av index == det som deklareras i if “statementet”.

```
if (arr[i] == 1) {
    one += 1;
} else if (arr[i] == 2) {
    two += 1;
}
```

Detta skrivs sedan ut med en clg som dividerar värden i variablerna med 10 för att få ut procentsatsen (1000/10 = 100 vilket är representerar procentsatsen). Detta ger då frekvensen på förekomsten av de nummer som genererats.



Uppgift 6

Först så sker en användarinteraktion för att definiera det heltal som programmet ska utgå ifrån (*inputNumber*). Genom att använda *ParseInt* i prompt så omvandlas det initiala strängen till ett integer-värde. Prompt i metoden frågar efter ett nummer mellan 1-1000 för att fortsättningsvis beräkna fakultet på detta värde.

För att kunna beräkna fakultet så används en funktionen vid namn *fakultet* som läser av värdet av variabel "*inputNumber*" som deklarerades i första steget.
Först så definieras en ny variabel vid namn *x* som kommer att representera summan som sedan ska returneras.

Till att börja med deklarerar vi x = 1; eftersom fakulteten ska börja på 1 eftersom faktoren 0 aldrig kommer att ske per definition. *x* kommer sedan att stiga i värde vilket man ser längre ner i funktionen.
Loopen som kommer efter detta definieras som följande:

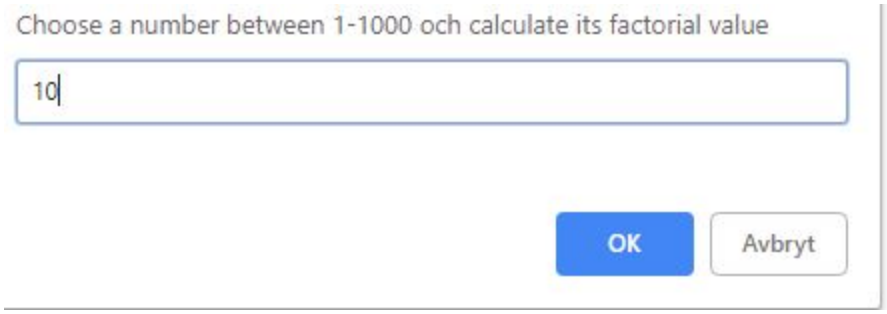
Statement 1 : *i=1*; *i* tillägnas värde 1 då indexeringen av antalet iterationer kommer användas i den matematiska uträkningen och därför måste börja på 1 (`i = 1`).
Statement 2 : *i < inputNumber+1;* eftersom loopen initieras med värde 1 så minskas antalet iterationer. Därav adderar vi antalet iterationer med 1 (`i < inputNumber+1`).
Statement 3: Efter varje iteration så ökar vi index[*i*] med 1 (`i++`).

Inuti for-loopen tar vi den aktuella summan multiplicerat med det aktuella index-värdet.

> *För att förtydliga användningen av aktuellt index-värde så sker fakulationen i strikt talföljd 1 till det aktuella talet.*

När for-loopen har fullföljt sina kriterier så returneras x.

För att sedan skriva ut detta på hemsidan används
`document.getElementById("ID").innerHTML`


Choose a number between 1-1000 och calculate its factorial value
10
OK
Avbryt


Arkiv | file:///C:/Users/pelle/Desktop/Javascriptuppgift/uppgift6/Uppgift6.html
Appar
How to Optimize You
Lars Andersen: a new
Is there an awesome
Exploring Ukr
The factorial of 10= 3628800