Динамични изрази. Оператори *Manipulate u Animate* . Управление на обекти. Структура на динамичните изрази. Динамични модули

Програмирането, което дискутирахме дотук беше статично, защото изходите са статични., т.е. ако се смени стойността на някакъв параметър , това не променя стойността на предварително изчислен израз.
Но ние можем да накараме *Mathematica автоматично* да променя символи и това да се отразява съответно и в изходите в бележника.
Това става с *динамичен език*. Подобно на графиките, този език има *основни обекти (примитиви) и динамични функции,* които генерират основни елементи.

Ще започнем с използването на основните функции Manipulate и Animate. След това ще разгледаме и основните обекти Dynamic и DynamicModule, за да разберем по-добре динамичния език и как можем да създаваме свои собствени интерактивни и динамични интерфейси.

Основни динамични манипулации

Основната идея за Manipulate е да се създаде панел, наречен демонстрация така, че да се движат управляемите плъзгачи, като се променя и изхода, независимо дали е числен, символичен или графичен. Възловата точка е да има такива изходи, които се сменят бързо така, че да може да се сравняват. Ако сравняването отнема много време, тогава Manipulate не е най-добрата идея да се използва, защото има вграден стопиращ елемент относно времето.

Динамичен изход на изрази

Да припомним, че вградената функция **Table** изчислява аргумента върху областта, зададена с нейния итератор, оформен като списък и дава *статичен* списък от стойности

```
In[12]:= Table[i^2, {i, 1, 100, 2}]

Out[12]= {1, 9, 25, 49, 81, 121, 169, 225, 289, 361, 441, 529, 625, 729, 841,
  961, 1089, 1225, 1369, 1521, 1681, 1849, 2025, 2209, 2401, 2601, 2809,
  3025, 3249, 3481, 3721, 3969, 4225, 4489, 4761, 5041, 5329, 5625,
  5929, 6241, 6561, 6889, 7225, 7569, 7921, 8281, 8649, 9025, 9409, 9801}
```

Ако искаме да изведем *динамичен* списък от стойности, то използвам функцията Manipulate, която има подобна структура.

Дискретна динамична структура:

Manipulate[израз на x, { x , начална ст., крайна ст., стъпка}]

Да започнем с класическия пример, при който итератора е дискретен и се променя от 1 до 100 със стъпка 2.

```
In[18]:= Manipulate[i^2, {i, 1, 100, 2}]
```

Manipulate автоматично създава потребителски интерфейс, който показва изхода заедно с управлението чрез плъзгач. Този плъзгач дава възможност динамично да се сменя стойността на параметъра. Когато се мести плъзгача с мишката, се променя стойността на променливата I и се появява стойността на израза i^2.

Out[18]=

Пример. Да се намерят всички делители на полинома $x^n - 1$ където n се променя от 1 до 100 със стъпка 1.

Да припомним отначало оператора ***Factor***, който разлага полином на множители

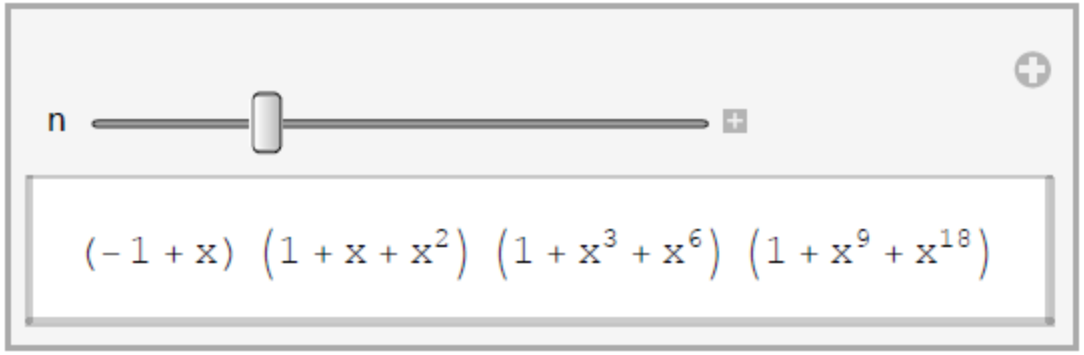
```
In[21]:= Clear[x]

In[22]:= Factor[x^10 - 1]
```

Out[22]= $(-1+x)\,(1+x)\left(1-x+x^2-x^3+x^4\right)\left(1+x+x^2+x^3+x^4\right)$

Ако искаме да оформим динамичен интерфейс, при който да се появяват делителите при различни стойности на n, то поставяме Factor като аргумент на Manipulate

```
In[23]:= Manipulate[Factor[x^n - 1], {n, 1, 100, 1}]
```

Out[23]=



Генерира се интерфейс с плъзгач, който като се мести с мишката се променят стойностите на n, а съответно се появат и делителите на полинома.

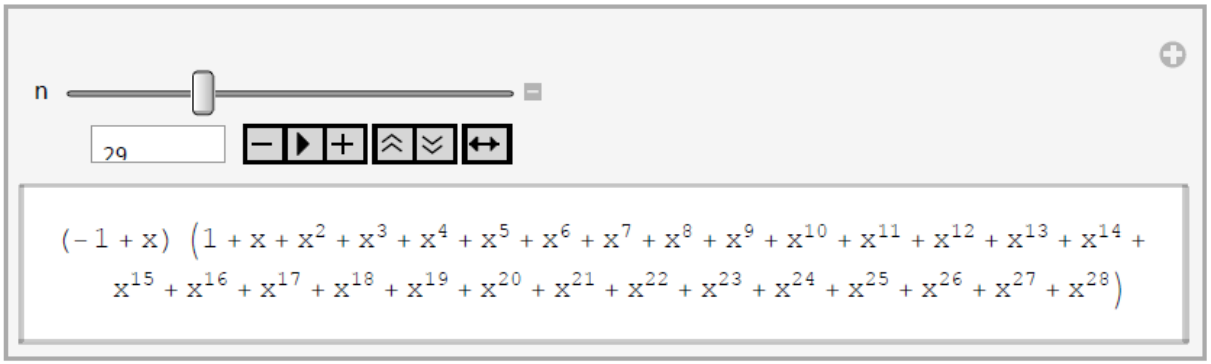
Ако искаме да започнем демонстрацията от средата на областта, от 10, то

```
In[25]:= Manipulate[Factor[x^n - 1], {n, 10, 100, 1}]
```

Ако искаме да добавим възможността да се появи и управляемата лента, на която се вижда степента на полинома. то

```
In[27]:= Manipulate[Factor[x^n - 1],
          {{n, "stepen    na  x^n-1"}, 1, 100, 1, Appearance -> "Open"}]
```

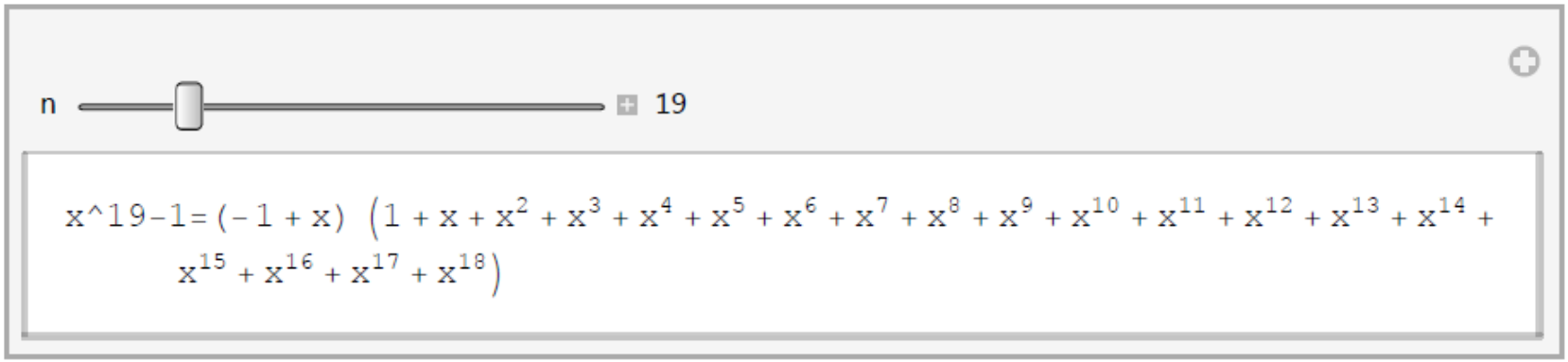
Out[27]=



Може да организираме изхода така, че да се появява и полинома и неговото разлагане

```
In[34]:= Manipulate[StringForm["x^``-1=``", n, Factor[x^n - 1]],
          {n, 1, 100, 1, Appearance -> "Labeled"}]
```

Out[34]=



Непрекъсната динамична структура

При оператора Manipulate, за разлика например от Do, ако липсва стъпката, тя не се подразбира 1, а се подразбира, че се изменя върху всичките реални числа, т.е. структурата

Manipulate[израз на x, { x , начална ст., крайна ст.}]

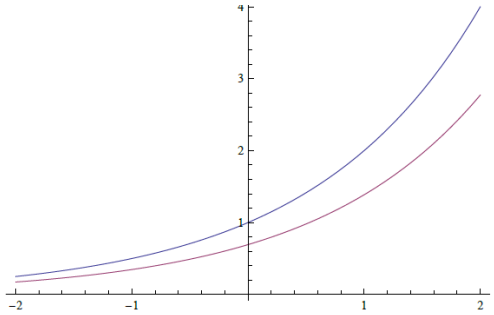
генерира интерфейс с плъзгач който се променя върху всички реални числа от ***начална ст.*** до ***крайна ст.***

Пример: да начертаем графиката на функцията $f(x)= b^x$ *и нейните производни при различни стойности на b.*

Предварително, да напишем командата, която чертае графиката при конкретна стойност на b, въведена чрез оператор за присвояване

```
In[35]:= b = 2; Plot[Evaluate[{b^x, D[b^x, x]}], {x, -2, 2}]
```

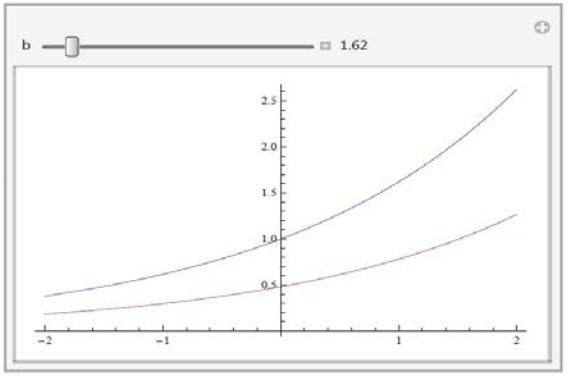
Out[35]=



След това поставяме оператора Plot в Manipulate, като премахваме оператора за присвояване

```
In[36]:= Manipulate[Plot[Evaluate[{b^x, D[b^x, x]}], {x, -2, 2}],
          {b, 1, 10, Appearance → "Labeled"}]
```

Out[36]=

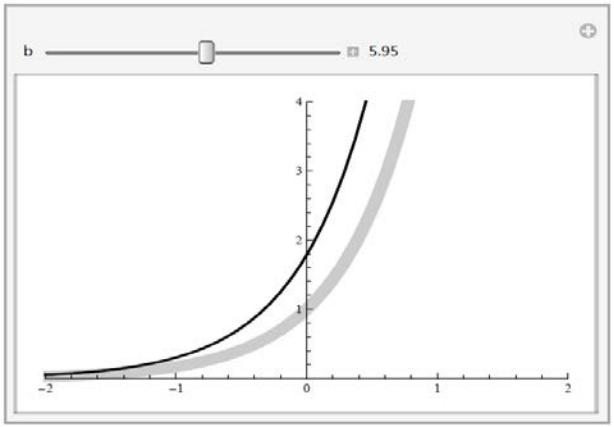


Да забележим, че има една особена стойност , при която двете графики съвпадат, това е b=e=2.71

или с използване на графичните опции

```
In[37]:= Manipulate[Plot[Evaluate[{b^x, D[b^x, x]}], {x, -2, 2},
           PlotStyle → {{GrayLevel[0.8], Thickness[0.025]}, {Black, Thick}},
           PlotRange → {{-2, 2}, {0, 4}}], {b, 1, 10, Appearance → "Labeled"}]
```

Out[37]=



Динамичен интерфейс с Animate

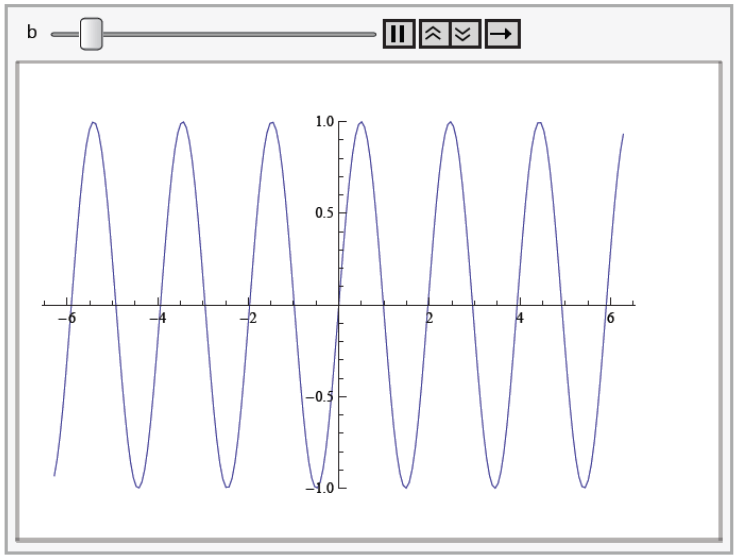
Animate[израз на x, { x , начална ст., крайна ст., стъпка}]

Този оператор автоматично създава подобен интерфейс както Manipulate, но осигурява по-малък контрол върху детайлите, защото анимира и сменя автоматично стойностите на параметрите, а не чрез местене на плъзгача с мишката .

Например, да анимираме функцията sin(b x) за стойности на параметъра b от 1 до 5

```
In[14]:= Animate[Plot[Sin[b x], {x, -2 Pi, 2 Pi}, PlotRange → 1], {b, 1, 5}]
```

Out[14]=



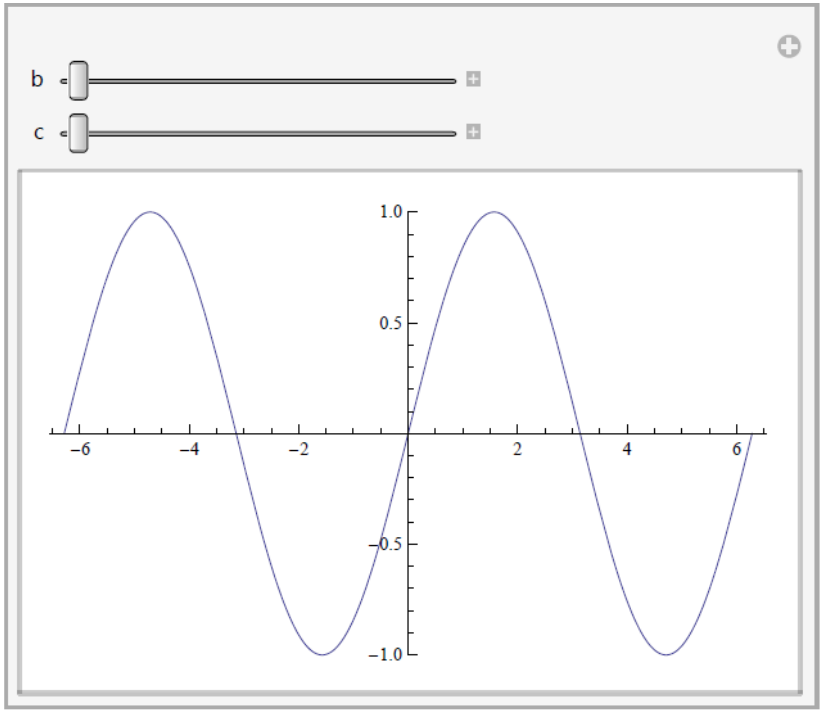
Да сравним с използването на Manipulate

```
In[15]:= Manipulate[Plot[Sin[b x], {x, -2 Pi, 2 Pi}, PlotRange → 1], {b, 1, 5}]
```

В действителност двете функции създават един и същ дисплей, и се организират по един и същи начин, но Manipulate дава възможност за повече раздвиженост и управление на повече променливи. Например, да разгледаме sin(b x +c), като разглеждаме параметър b(дава периода) и параметър c(дава изместването по оста x).

```
In[16]:= Manipulate[Plot[Sin[b x + c], {x, -2 Pi, 2 Pi}, PlotRange → 1],
```

Out[16]=



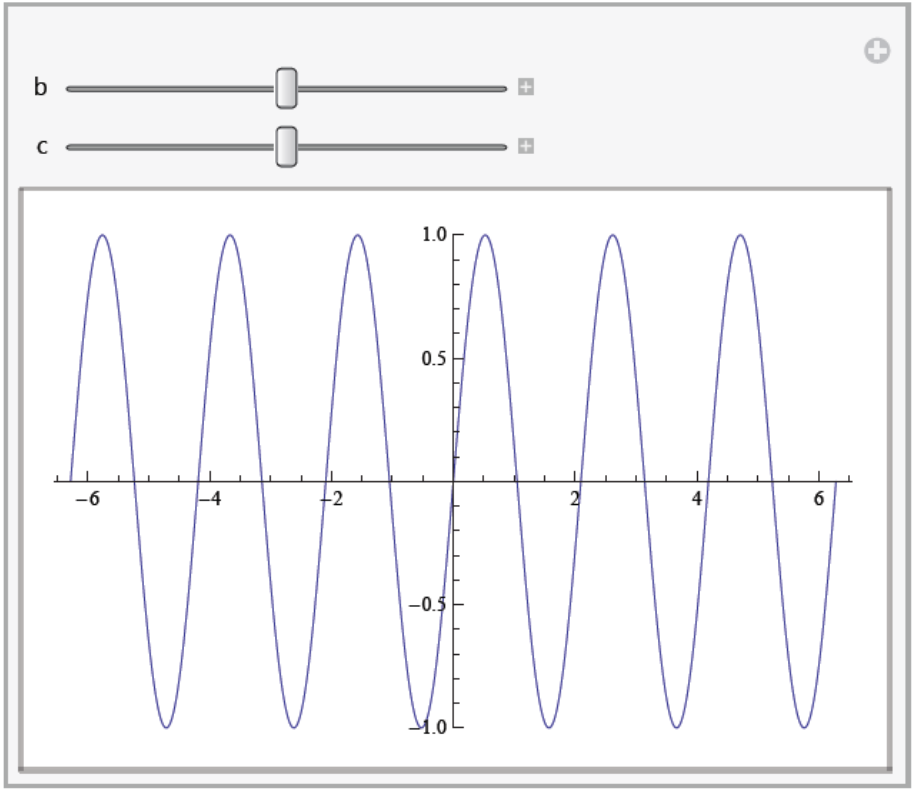
Началната стойност на параметъра b по подразбиране е 1(виж списъка)
а на параметъра c е -2π.

Можем да променим началната стойност на параметъра b, да започва например, от 3, и на параметъра c – да започва от 0. За целта оформяме списъците {b,3} и {c,0}.

```
In[17]:= Manipulate[Plot[Sin[b x + c], {x, -2 Pi, 2 Pi}, PlotRange → 1],
          {{b, 3}, 1, 5}, {{c, 0}, -2 Pi, 2 Pi}]
```

Оформянето на списъка {b,3} дава възможност демонстрацията да започне от 3, но плъзгача е от 1 до 5.

Out[17]=



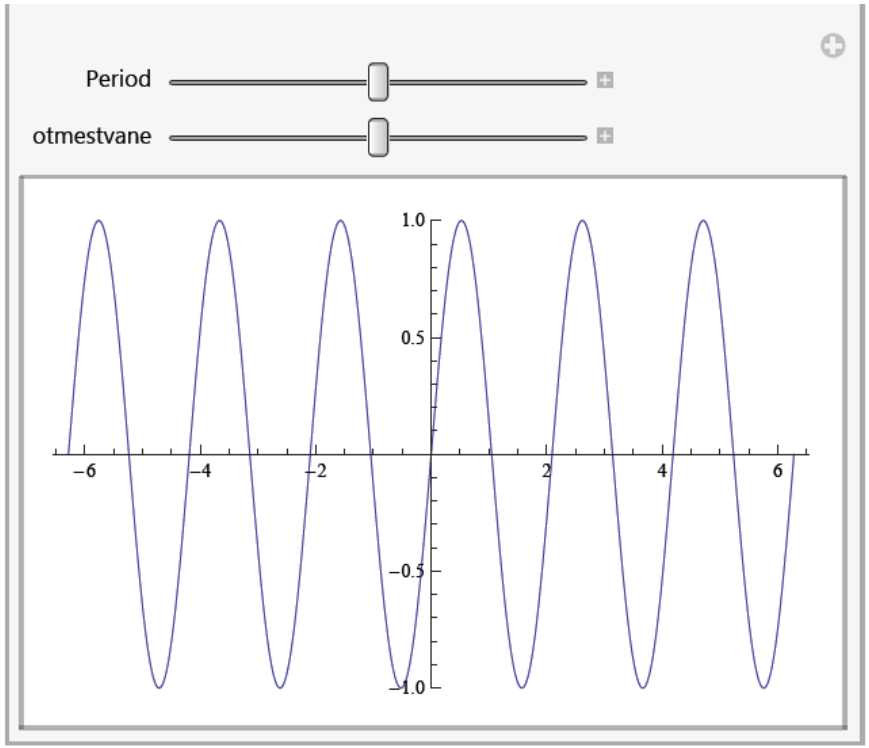
По-удобна за използване вариация с наименуване на управлението за всеки параметър е :

{{param, init, label}, min, max}

където *label* обикновено е списък, но може да бъде всеки израз.

```
In[19]:= Manipulate[Plot[Sin[b x + c], {x, -2 Pi, 2 Pi}, PlotRange → 1],
  {{b, 3, "Period"}, 1, 5}, {{c, 0, "otmestvane"}, -2 Pi, 2 Pi}]
```

Out[19]=



Управление на обекти

В предишните примери, параметрите се управляваха с плъзгача. Преместването на плъзгача променя стойността на параметъра и на всеки израз, зависещ от него вътре в Manipulate. Но понякога се налага да се избират стойности от списък с дискретни стойности. Тогава е удобно да се постави специален управляващ обект, като се използва различен синтаксис за списъка от параметри:

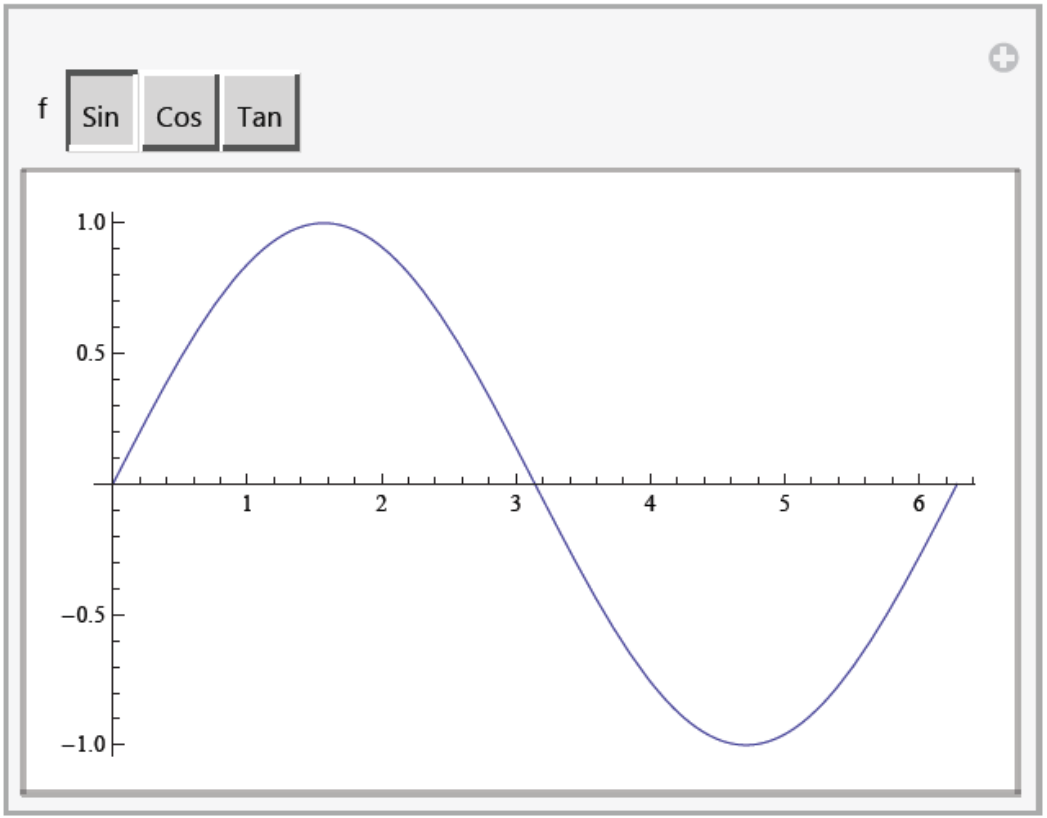
{ парам , {cm1, cm2, cm3,…. }}

който кара Manipulate автоматично да избира съответните стойности вместо да се движи плъзгача.

Например, може да оформим динамичен обект, който да ни показва графиките на три графики, като си избираме чрез кликване коя от тях да ни се показва.

In[20]:=
```
Manipulate[Plot[f[x], {x, 0, 2 Pi}], {f, {Sin, Cos, Tan}}]
```

Out[20]=



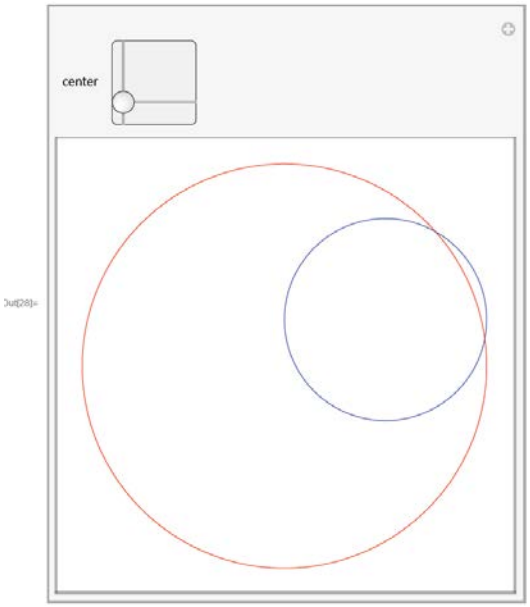
Двумерен плъзгач.

Когато е необходимо два различни параметъра да се управляват едновременно, се оформя двумерен плъзгач. Синтаксисът на управляващия обект е

{ парам,{ min x, min y}, {max x, max y }}

Например, да начертаем фиксиран син кръг с радиус 1 и център (0,0) и друг червен с радиус 2 и център който се променя от (-1,-1) до (1,1), като параметъра е с име center

```
In[28]:= Manipulate[Graphics[{{Red, Circle[center, 2]}, {Blue, Circle[{0, 0}, 1]}}],
  {center, {-1, -1}, {2, 2}}]
```



Пример. Създайте динамичен интерфейс с графика на x=sin(nt), y=cos(mt), z=sin(5t) за дискретни стойности на n ,m от 1 до 10, където аргументът t се изменя непрекъснато в интервала [-π, π].

Отначало да си припомним чертаенето на графика на параметрично зададени функции с **ParametricPlot3D**

```
n = 3; m = 4; ParametricPlot3D[
 {Sin[n t], Cos[m t], Sin[5 t]}, {t, -Pi, Pi}, PlotStyle → Tube[0.05]]
```

Поставяме този оператор без операторите за присвояване като аргумент на Manipulate и добавяме итераторите

```
Manipulate[ParametricPlot3D[{Sin[n t], Cos[m t], Sin[5 t]},
  {t, -Pi, Pi}, PlotStyle → Tube[0.05]], {n, 1, 10, 1}, {m, 1, 10, 1}]
```

Структура на динамичните изрази

Както при графиките, така и тук, за създаването на интерактивен интерфейс има основни елементи, които се използват при конструирането на високо ниво функции като Manipulate. Тези основни елементи са Dynamic и DynamicModule. Аналогично, както при графиките, може да се създава динамичен интерфейс директно с използването на тези основни елементи.

Какво значи динамичен обект? Как да се разбира?

Когато се прави присвояване, се фиксира стойност на променлива/символ в момента, в който се дефинира. Например, на символа t се дава стойност 10 чрез присвояване t=10. когато се използва t, статично тя се замества с нейната стойност, например в 4 t -1 се получава 39. Ако сменим стойността на t с оператор за присвояване, то навсякъде в следващите изрази ще се използва новата стойност, като правилото t=10 отива в паметта на ядрото ядрото. Ако затворим ядрото, тази стойност вече не се помни.

Но, може да се използва ДИНАМИЧЕН изход, който автоматично се променя, като това рефлектира и на настоящата стойност на символа. Използва се оператор
Dynamic[израз] като изхода е израза, който обаче вътрешно е динамичен обет.

Създаване на интерактивен израз, чрез Dynamic

Първо трябва да се използва обекта-плъзгач, **Slider**[], който по подразбиране се мести за стойности от 0 до 1. Плъзгача може да се мести с мишката но ако Slider няма аргумент, то той е свързан с нищо.

```
In[1]:= Slider[]
```

Out[1]= 

Нека да използваме динамична променлива z, и да се появи z надясно от плъзгача (ако липсва z, то се появява само плъзгача, но се създава и динамичната променлива z, което не се вижда).

```
In[2]:= {Slider[Dynamic[z]], z}
```

Out[2]= 

За да се появи стойността на z надясно от плъзгача, използваме отново Dynamic[z]

```
In[3]:= {Slider[Dynamic[z]], Dynamic[z]}
```

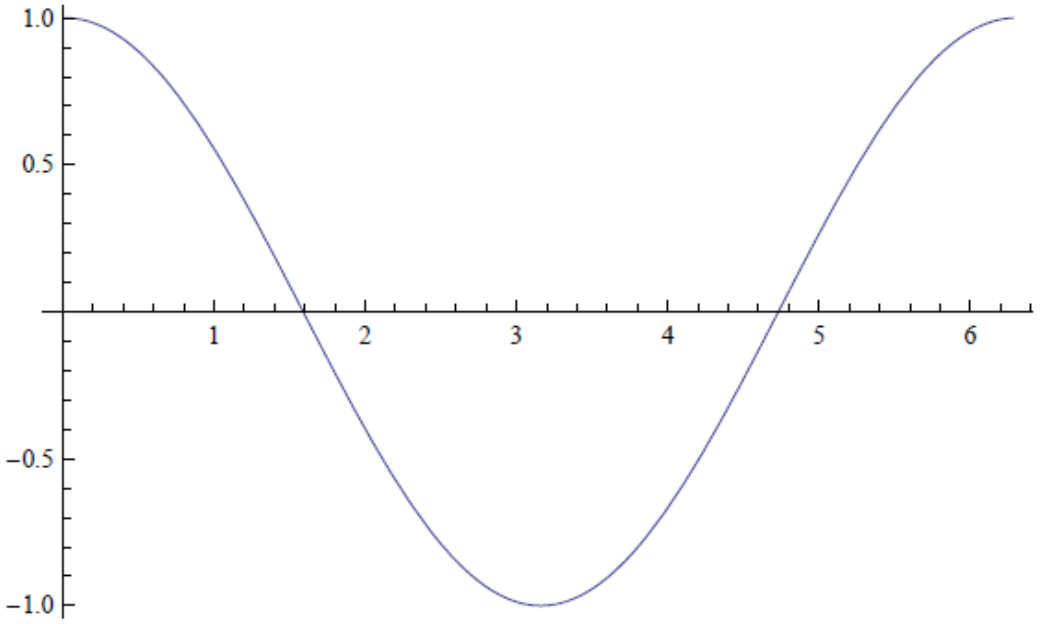
Out[3]= 

Ако местите плъзгача, стойността на z се променя- появява се надясно от плъзгача.

Забележка. Ако се мести последния плъзгач, то се мести и предишния. Това е защото предишния също има като аргумент динамичната променлива z. Освен това, ако се мести плъзгача и следната графика автоматично ще се променя, понеже се сменя стойността на z.

```
In[5]:= Dynamic[Plot[Sin[x + 4 z], {x, 0, 2 Pi}]]
```

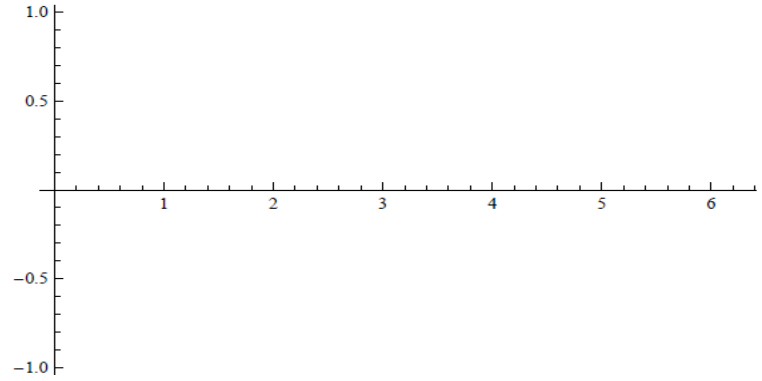
Out[5]=



Когато правите динамична графика, внимавайте да не сте поставили променливата z в Dynamic, защото това няма да създаде динамична графика.

```
In[6]:= Plot[Sin[x + 4 Dynamic[z]], {x, 0, 2 Pi}]
```

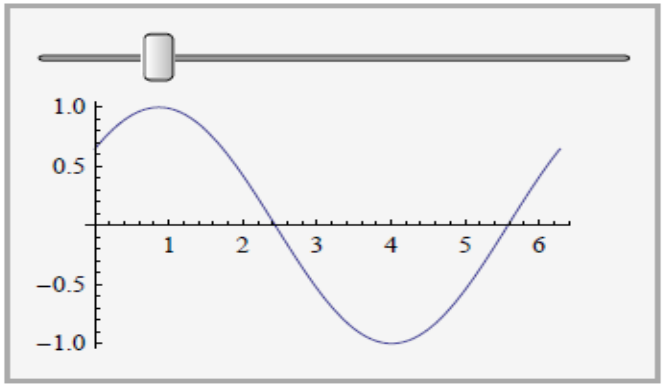
Out[6]=



В действителност, не се появява графика, защото Plot се нуждае от специфични стойности за да изчертае крива. За да създаде динамична графика, Plot се поставя в Dynamic. Освен това е важно заедно с плъзгача да се оформи и динамичния екран(интерфейса), както при Manipulate.

```
In[7]:= Panel[Column[{Slider[Dynamic[z]], Dynamic[Plot[Sin[x + 4 z], {x, 0, 2 Pi}]]}]]
```

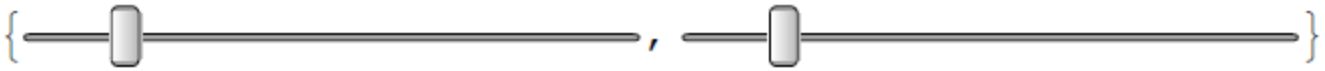
Out[7]=



Динамични модули

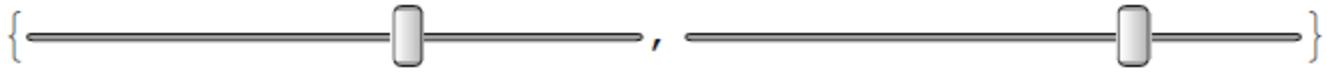
Ако има повече от една динамична променлива, тогава се оформя динамичен модул. Например, по долу, като се използва двукратно dynamic, то на екрана се появяват два плъзгача, но като местим единия, се мести и другия, защото и двата се отнасят за една и съща променлива x.

```
In[8]:= {Slider[Dynamic[x]], Slider[Dynamic[x]]}
```

Out[8]= 

Ако искаме двете променливи да са независими, като почват с различни начални стойности, то използваме DynamicModule

```
In[11]:= {DynamicModule[{x = 0.25}, Slider[Dynamic[x]]],
          DynamicModule[{x = 0.75}, Slider[Dynamic[x]]]}
```

Out[11]= 

Полезни съвети

Тъй като динамичните изрази се променят бързо, то трябва да се внимава с изчисленията които се извършват.

Първо, да се внимава с използването на Dynamic. Той загражда променливите, които ви трябват, но може да не се получи ефекта, който искате.

Например, като местите плъзгача се виждат всички стойности, които се променят, но не и стойността на целия интеграл.

```
In[12]:= DynamicModule[{a},
          {Slider[Dynamic[a], {0, 1}], Integrate[Exp[Dynamic[a] v], {v, 0, 1}]}]
```

Out[12]= 

То е защото искаме и интегрирането да е динамично, затова го слагаме в Dynamic.

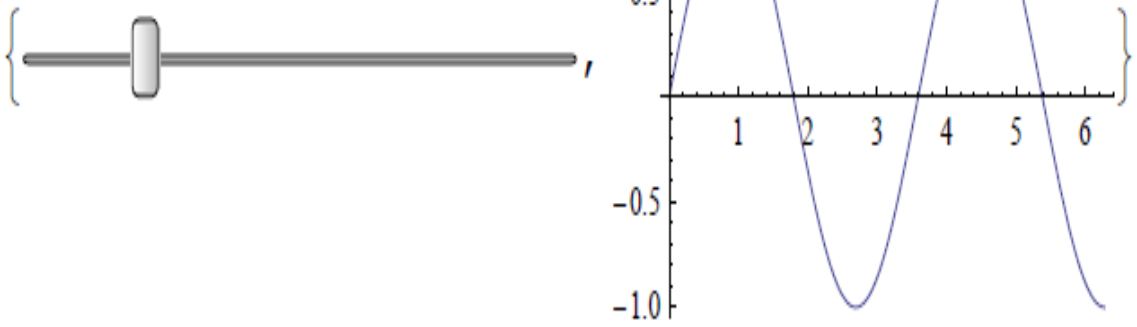
```
In[15]:= DynamicModule[{a},
          {Slider[Dynamic[a], {0, 1}], Dynamic[Integrate[Exp[a v], {v, 0, 1}]]}]
```

Out[15]= 

Същото се отнася и за графиките

```
In[17]:= DynamicModule[{a}, {Slider[Dynamic[a], {1, 5}],
   Dynamic[Plot[Sin[a v], {v, 0, 2 Pi}, PlotLabel → StringForm["period=`1`", a]]]}]
```

Out[17]=



КРАЙ