



macromedia
FLASH
MX

Програмиране с ActionScript MX

/факултативен курс/

преподавател: Мая Стойкова

ЛЕКЦИЯ № 4

**Създаване на потребителски обекти.
Динамични данни в ActionScript MX
и тяхното приложение.**

СЪДЪРЖАНИЕ

I. Създаване на потребителски обекти.

- Същност на механизма на обектите. Сравнение между създаване на инстанция на обект и класове от обекти.
- Обектите като контейнери. Връзка наследник-родител.
- Дефиниране на потребителски клас и създаване на негова инстанция.
- Програмиране на обект чрез приспособяване на неговия прототип. Свойства на ниво инстанция.
- Създаване на подкласове, потребителски методи за потребителски класове.
- Наблюдения на свойства и разширяване на методите на съществуващи обекти. Дефиниране на потребителски методи за вече създадени обекти.
- Регистриране на класове.

СЪДЪРЖАНИЕ

II. Динамични данни в ActionScript MX и тяхното приложение.

- Създаване на променливи, масиви, динамични текстови полета и извличане на информация от тях.
- Извличане на данни.

Създаване на потребителски обекти

- До сега разгледах само как се използват готови обекти. В тази лекция ще се спрем на създаване на потребителски обекти. Важното тук е да разберем, че класът описва само как е направен и как работи даден обект, тоест се явява своеобразен проект на обекта, а инстанциите са това, което създаваме от този обект. Но често инстанциите са познати също като обекти. В повечето случаи инстанциите и обектите са едно и също нещо.
- Понеже концепцията за потребителското създаване на обектите е доста абстрактна, то следващите ни упражнения ще бъдат малко по-неграфични от досегашните... Ще разгледаме какво точно става вътре в ActionScript при изпълнението на даден скрипт.

Създаване на потребителски обекти

- Основните средства, които ни помагат за това са: функцията `trace` и командата `List Variables` (от `Debug -> List Variables`, в средата на самото тестване). Така ще видим данните, които имаме в нашия филм, но по принцип остават скрити за нас по време на неговото възпроизвеждане.

Пример: `trace(_root._totalframes);`

Поставим ли това действие в кадър първи на клип, примерно съдържащ 50 кадъра, веднага след стартирането на филма ще се отвори прозорецът `Output` и ще визуализира в него 50.

- Горното просто действие може много да ни помогне за изучаването на начина на работа на обектите.

Обектите като контейнери. Връзка наследник-родител

- Обектите в ActionScript са и контейнери, макар и по-особени. Те не само съхраняват полезни неща като данни (свойства, атрибути) и дори други обекти, но изпълняват и други задачи, свързани с тяхното съдържание. Например обектът `String`, който използваме за съхранение на низова стойност има методи, които ни позволяват да управляваме и работим с неговото съдържание. Като под съдържание разбирате съответния низ, който се съхранява. Като това ще го илюстрирам с пример.
- В следващия пример в обектът `Object` няма да влагаме конкретно приложение, а ще го разгледаме като универсален обект.
- Пример...

Обектите като контейнери. Връзка наследник-родител

- Когато поставяме една променлива във времедиаграма, всъщност задаваме ново свойство на инстанция на съответния обект `MovieClip`. От своя страна поставяйки една времедиаграма (респективно инстанция на обекта `MovieClip`) в друга, както правим за клипчета, то създаваме между двете връзка родител-наследник.
- В примера, който създадохме в преди малко (`person1.head.memories`) обектът `head` е обект-наследник на `person1`, а `memories` на `head`, съответно `person` е родител на `head` и т.н. Важното е да знаем, че това, което се случва с родителя е става и с неговите наследници.
- Следва пример...

Дефиниране на потребителски клас и създаване на негова инстанция

- До тук използвахме инстанция на универсалния обект `Object`, за да покажем как може да работи един обект (в случая `person`). В следващите слайдове ще разгледам как може сами да си създадем потребителски клас (`Person`). Той ще ни послужи като оригинал за създаването на инстанции на хора. Всъщност ще видим, че тази стратегия е по-добра и ефикасна от предната с дефиниране на `person1`.
- В `ActionScript` е прието имената на класовете да се пишат с главна буква, а на инстанциите/обекти с малка. Така кодът при по-късно използване е лесен за разбиране и четимост.
- Когато създаваме инстанция на клас, `ActionScript` си генерира един временен обект `Activation`, който в началото е празен.

Дефиниране на потребителски клас и създаване на негова инстанция

- Този обект се предава на дефиницията на класа и постепенно придобива вид приемайки стъпка по стъпка различните характеристики на класа. По време на този процес временният обект се означава с термина `this`. Тоест, когато имаме следния скрипт: `this.name = "Mihaela"`, то това означава, че на свойството `name` трябва да бъде зададена стойност "Mihaela". Когато Activation завърши конструирането, му се дава съответното име и той става достъпен за използване.
- Така зададен дотук класа създава идентични инстанции;(, в следващия пример ще го променим така, че да създаваме различни негови представители...
- За да създаваме лесно инстанции на клас от произволни времедиаграми, без въвеждането на целеви път, трябва да направим глобална дефиницията на нашия клас.

Дефиниране на потребителски клас и създаване на негова инстанция

- Този обект се предава на дефиницията на класа и постепенно придобива вид приемайки стъпка по стъпка различните характеристики на класа. По време на този процес временният обект се означава с термина `this`. Тоест, когато имаме следния скрипт: `this.name = "Mihaela"`, то това означава, че на свойството `name` трябва да бъде зададена стойност "Mihaela". Когато Activation завърши конструирането, му се дава съответното име и той става достъпен за използване.
- Така зададен дотук класа създава идентични инстанции;(, в следващия пример ще го променим така, че да създаваме различни негови представители...
- За да създаваме лесно инстанции на клас от произволни времедиаграми, без въвеждането на целеви път, трябва да направим глобална дефиницията на нашия клас. Пример....

Дефиниране на потребителски клас и създаване на негова инстанция

- Този обект се предава на дефиницията на класа и постепенно придобива вид приемайки стъпка по стъпка различните характеристики на класа. По време на този процес временният обект се означава с термина `this`. Тоест, когато имаме следния скрипт: `this.name = "Mihaela"`, то това означава, че на свойството `name` трябва да бъде зададена стойност "Mihaela". Когато Activation завърши конструирането, му се дава съответното име и той става достъпен за използване.
- Така зададен дотук класа създава идентични инстанции;(, в следващия пример ще го променим така, че да създаваме различни негови представители...
- За да създаваме лесно инстанции на клас от произволни времедиаграми, без въвеждането на целеви път, трябва да направим глобална дефиницията на нашия клас.

Дефиниране на потребителски клас и създаване на негова инстанция

- Пример: `_global.имеклас = function () { }`
Така направо можете да създадете инстанция на клас, където и когато поискате само с: `new Person()` ;
Тоест без да указваме целеви път 😊...

Програмиране на обект чрез приспособяване на неговия прототип.

Свойства на ниво инстанция

- За всички инстанции на класове в ActionScript част от характеристиките са универсални, но има и други, които варират, въпросът е как можем да спестим код като не пишем всеки път общите свойства.
- Това става с така нареченият *обект-прототип*. Той съдържа свойства и стойности, които са универсални за класа. Можем да го възприемаме като обект, стоящ точно след дефиницията на класа, специално прикрепен към нея.
- Всички класове обекти, включително вградените в езика, имат асоцииран обект-прототип. В него не се съдържат никакви стойности докато не се добави скрипт към тях. Като свойствата на обекта-прототип се *наследяват* от всички свойства по-нататък

Програмиране на обект чрез приспособяване на неговия прототип.

Свойства на ниво инстанция

- ActionScript винаги проверява при търсене на стойност за дадена променлива първо на ниво инстанция! Ако открие, присъединява нея, ако не търси в обекта-прототип.
- С обекта-прототип можете да правите усъвършенствания по всички инстанции, което в много случаи е полезно...
- Добавянето, изтриването или обновяването на свойства в обекта-прототип на клас предизвиква незабавно отразяване на промените във всички инстанции на класа. Има само едно изключение, което ще разгледаме след малко;)
- Свойството на ниво инстанция е свойство, зададено в дефиницията на класа или при създаването на инстанцията, или е стойност, зададена чрез директно адресиране на инстанцията.

Програмиране на обект чрез приспособяване на неговия прототип.

Свойства на ниво инстанция

➤ Пример:

```
person1.age = 16; //ниво прототип  
person1.legs = 2; // ниво инстанция  
person1.name = "Justin"; // ниво инстанция  
person1.head = new Object(); //ниво прототип  
person1.head.eyes = 2; //ниво прототип  
person1.head.memories = new Array(); //ниво прототип
```

➤ Важно се отбележи, че свойството на ниво инстанция има предимство пред това на ниво прототип.

➤ Можем да изтрием свойство на ниво инстанция с:

```
delete име_инстанция.свойство;
```

➤ Предимство на свойства ниво инстанция е, че можем да зададем допълнителни характеристики, освен тези в дефиницията на класа и прототипа...

Програмиране на обект чрез приспособяване на неговия прототип.

Свойства на ниво инстанция

- Ако добавим свойство към дефиницията на класа, то ще е валидно само за новосъздадените инстанции, тъй като функцията конструктор присъединява характеристики само в момента на тяхното създаване.

Създаване на подкласове, потребителски методи за потребителски класове

- Подкласовете са класове, притежаващи общите характеристиките на класа, но всеки от тях си има и уникални свойства. Например подклас на `Person`, може да е `Informatitsi`.
- Можем да укажем, че един клас е подклас на друг като:
`ImePodKlas.prototype = new ImeSuperKlas;`