

Programy wykonywalne po stronie klienta – skrypty (JavaScript, VBScript), aplety Javy, obiekty ActiveX

Maciej Piasecki
Politechnika Bialostocka
Wydział Informatyki
ul. Wiejska 45a
macpias@poczta.fm

STRESZCZENIE

Artykuł zawiera informacje na temat czterech technologii internetowych, które są przydatne przy tworzeniu stron. Opisane są podstawowe konstrukcje językowe, zalety i wady każdego z rozwiązań. Artykuł zawiera informacje na temat semantyki języków. Przedstawione są po krótko najważniejsze funkcje języka. Omówione są także możliwości każdej z technologii. Opisane jest także dołączanie ich do stron internetowych. Dla każdej technologii jest opisane zagadnienie bezpieczeństwa.

WSTEP

Coraz częściej korzystając z Internetu natykamy się na strony coraz ciekawiej zrobione. Jednym z przykładów takiej strony jest strona 2advancedstudio.com. Wygląda bardzo ciekawie. Zawiera wszystkie elementy multimedialności. Do tworzenia stron interaktywnych wykorzystuje się różnego rodzaju języki skryptowe i podobne mechanizmy, o których powiem. Jest jeszcze mechanizm Flash, który pozwala na tworzenie różnego rodzaju animacji i filmów oraz interakcji między użytkownikiem a stroną. Te dodatki czy inaczej mówiąc programy wykonywalne po stronie klienta pozwalają uatrakcyjnić stronę oraz na kontakt z użytkownikiem. Takie strony zatrzymują użytkownika. Oczywiście jeszcze sporo zależy od samego projektu strony. Jeżeli zdoła się zatrzymać użytkownika na dłużej i nie opuści zdegustowany szybko naszej strony to już jest sukces. Dlatego trzeba uważać, żeby nie umieścić za dużo zbędnych dodatków, które mogłyby spowodować, że strona będzie się ładować wolno i użytkownicy z wolnym łączem zrezygnują z jej obejrzenia. Ważnym aspektem jest także wybranie odpowiedniej technologii dla swojej strony. Dlatego też w swoim referacie spróbuję przedstawić najważniejsze z nich i opisać ich wady oraz zalety.

JAVASCRIPT

Język JavaScript jest między platformowym, zorientowanym obiektowo językiem skryptowym. Jądro JavaScript zawiera podstawowy zbiór obiektów takich jak tablice, daty, funkcje matematyczne oraz podstawowy zbiór elementów języków programowania takich jak operatory, funkcje i instrukcje warunkowe. Język JavaScript po stronie klienta został rozszerzony przez dodanie obiektów kontrolujących przeglądarkę i Document Object Model. Na przykład skrypty

pozwalają aplikacjom na dodanie elementów do formularzy w HTMLu i reagowanie na zdarzenia wywołane przez użytkownika takie jak kliknięcia myszka, wprowadzanie danych do formularza i nawigacje po stronie.

JavaScript pozwala na tworzenie aplikacji, które działają przez Internet. Aplikacja klienta uruchamiana jest w przeglądarce, a aplikacja serwera na serwerze. Używając JavaScript można stworzyć dynamiczne strony HTML, które przetwarzają dane wprowadzone przez użytkownika i przetwarzają dane używając specjalnych obiektów, plików i relacyjnych baz danych.

JavaScript's Live Connect pozwala na komunikację między Java a JavaScript.. Z JavaScript można utworzyć obiekt Javy i korzystać z jego publicznych metod i pól. Z Javy można otrzymać dostęp do obiektów JavaScript, właściwości i metod.

JavaScript po stronie klienta i po stronie serwera zawiera następujące elementy:

- słowa kluczowe
- składnię instrukcji i semantykę
- zasady dla wyrażen, zmiennych i literalów
- underlying object model
- predefiniowane obiekty i funkcje takie jak tablice, data i funkcje matematyczne

Przeglądarki mogą interpretować JavaScript dołączony do kodu strony. Kiedy przeglądarka załaduje taką stronę, serwer przesyła cały dokument HTML wraz z JavaScript'em przez sieć do klienta. Przeglądarka czyta cały dokument wyświetlając zinterpretowaną stronę i uruchamiając procedury JavaScript'u jak je napotka. Zostało to pokazane na rysunku 1.2.

Funkcje JavaScript po stronie klienta dołączone do strony mogą odpowiadać na zdarzenia związane z interakcją przeglądarki z użytkownikiem takie jak kliknięcia myszka, wprowadzenie tekstu, nawigacja po stronie. Na przykład można napisać funkcję w JavaScript, która będzie sprawdzała czy użytkownik poprawnie wprowadził numer telefonu lub kod pocztowy. Bez jakiegokolwiek transmisji w sieci dołączony skrypt do strony może sprawdzić poprawność daty i wyświetlić okno dialogowe z informacją, że data jest niepoprawna.

JavaScript i Java są podobne do siebie w niektórych przypadkach, ale fundamentalne różnice są w innych. Język JavaScript jest podobny do Javy ale nie ma statycznego wprowadzania i rygorystycznego sprawdzania składni. JavaScript wspiera większość składni wyrażen z Javy i podstawową kontrolę przebiegu wykonania.

Program w Javie musi zostać skompilowany natomiast program w JavaScript jest wykonywany podczas przeglądania strony wtedy, gdy jest wywołany. Model obiektowy w JavaScript bazuje na prototypach w przeciwieństwie do bardziej ogólnego opierającego się na klasach. Prototypowy model pozwala na dynamiczne dziedziczenie – to znaczy, że to co jest dziedziczone pozwala zmienić indywidualne obiekty. JavaScript pozwala na deklaracje funkcji bez żadnych specjalnych wymagań. Funkcje mogą być właściwościami obiektów, wykonywane jako swobodne metody.

JavaScript jest bardzo swobodnym językiem w porównaniu do Java. Nie trzeba deklarować wszystkich zmiennych, klas i metod. Nie trzeba się przejmować czy metody w klasie są publiczne, prywatne czy też chronione i nie trzeba implementować interfejsów. Zmienne, parametry i typy zwracane przez funkcje nie są dokładnie deklarowane.

Java jest językiem programowania opartym na klasach w celu szybszego wykonywania i bezpieczeństwa implementacji. Bezpieczeństwo oznacza, że nie można rzutować wartości całkowitej na referencje do obiektu lub dostępu do chronionej pamięci co może zniszczyć kod bajtowy Javy. Model obiektowy oparty na klasach oznacza, że programy zawierają klasy i ich metody. Dziedziczenie klas w Javie i ścisła kontrola kodu programu wymagają zwartej, powiązanej ze sobą hierarchii obiektów. Te wymagania Javy sprawiają, że jest on bardzo złożony.

W porównaniu do Javy, JavaScript jest mniejszym, dynamicznym językiem. Język skryptowy jest dostępny dla szerszej grupy osób, ponieważ jego składnia jest prostsza, jest funkcjonalny, i ma minimalne wymagania związane z tworzeniem obiektów.

Następujące wartości są rozpoznawane przez język JavaScript:

- liczby
- wartości logiczne
- stringi
- null
- niezdefiniowane

JavaScript jest dynamicznym językiem. Oznacza to, że nie trzeba podawać typu zmiennej podczas jej deklaracji, typy są automatycznie konwertowane podczas wykonywania skryptu wtedy, gdy tylko jest to potrzebne.

Zmienne mogą być używane jako symboliczne nazwy dla wartości w aplikacji. Identyfikator lub nazwa musi się zaczynać od litery lub podkreślenia. JavaScript odróżnia wielkość liter. Zmienna może być zadeklarowana na dwa sposoby przez przypisanie zmiennej wartości lub przez podanie przed zmienną słówka *var*. Tablica lub zmienna, której nie przypisano wartości ma wartość nieokreśloną.

Literale służą do reprezentowania zmiennych w JavaScript. Są one związane z wartościami i nie są zmiennymi. W JavaScript występują następujące literale:

- literale tablicowe
- literale logiczne
- literale zmiennie-przecinkowe
- całkowite
- literale obiektowe
- literale stringowe

Wyrażenie jest poprawnym zbiorem zmiennych, literalów, operatorów i wyrażen, z których po obliczeniu powstaje pojedyncza wartość. Wartość może być zmienna logiczna, string albo liczba.

Są dwa typy wyrażen: te, które przypisują wartość do zmiennej i te, które po prostu mają wartość.

W JavaScript można wyróżnić następujące typy operatorów:

- przypisania
- porównania
- arytmetyczne

- bitowe
- logiczne
- znakowe
- specjalne

Do specjalnych operatorów należa:

- warunkowy
- przecinka
- delete
- new
- this
- typeof
- void

Wyrażenia regularne w JavaScript tworzy się na dwa sposoby: używając inicjalizatora obiektu lub przez wywołania konstruktora obiektu RegExp.

W JavaScript są dostępne różnego rodzaju instrukcje:

1) Warunkowe: if ... else ... oraz switch

```

if(wyrażenie) {
    instrukcja-1;
}
[else {
    instrukcja-2;
}]

switch(zmienna) {
    case wartosc-1:
        instrukcja-1;
        break;
    case wartosc-2:
        instrukcja-2;
        break;
    ...
    default: instrukcja-3;
}

```

2) Pętle: for, while, do while, label, break i continue

```

for(wartosc_pocatkowa;warunek;zmiana_zmiennej) {
    instrukcja;
}

do {
    instrukcje;
}
while (warunek)

while (warunek)
{

```

```
        instrukcje;
    }

etykieta:
    instrukcje;

break

break [etykieta]

continue

continue [etykieta]
```

3) Petle dzialajace na obiektach: for ... in

```
for(zmienna in obiekcie)
{
    instrukcje;
}

with(obiekt) {
    instrukcje;
}
```

4) Komentarze

```
// To jest komentarz jednolinijkowy
/* To jest
   komentarz dwulinijkowy */
```

Funkcje sluza do budowy bloków. Aby zdefiniowac funkcje nalezy za slowem kluczowym podac jej nazwe, a potem argumenty podzielone przecinkami. Nastepnie wpisuje sie instrukcje ograniczone nawiasami klamrowymi. Przykladowa definicja funkcji:

```
function potega(liczba) {
    return liczba * liczba;
}
```

Funkcje wywoluje sie przez podanie nazwy funkcji wraz z parametrami aktualnymi. Przyklad wywołania funkcji:

```
a=potega(5) // a=25
```

Argumenty funkcji sa przechowywane w tablicy. Dzieki temu mozna sie odwoływac do nich za pomoca indeksów. Indeksowanie zaczyna sie od zera.

```
arguments[i]
nazwafunkcji.arguments[i]
```

Nie trzeba w deklaracji funkcji podawać wszystkich parametrów, gdyż do nich będzie się można odwołać za pomocą tablicy.

Funkcje predefiniowane:

- eval
- isFinite
- isNaN
- parseInt i parseFloat
- Numer i String
- escape i unescape

Obiekty w JavaScript składają się z właściwości, którymi mogą być zmienne i obiekty. Obiekty posiadają także metody. Do właściwości obiektu można się odwoływać używając notacji kropkowej podając nazwę obiektu a po kropce pole do którego chce się mieć dostęp.

Obiekty tworzy się przez podanie nazwy obiektu i jako wartość inicjującą ciąg nazw i wartości w nawiasach klamrowych. Obiekt można także stworzyć przez użycie konstruktora. Przykładowe tworzenie obiektów:

```
nazwaobektu={nazwa1: wartosc, nazwa2: wartosc, ...}
```

```
function nazwa(zmienna) {  
    this.parametr=zmienna  
    ...  
}
```

Do właściwości obiektu można odwoływać się przez dodanie do nazwy obiektu indeksu np. nazwa[3].

W JavaScript można zdefiniować właściwości dla typu obiektu. Oznacza to że można zainicjować właściwości obiektu. Dzięki temu każdy nowo utworzony obiekt będzie miał zainicjalizowane dane pole.

```
typobjektu.property.pole=null
```

Metody danego obiektu definiuje się przez podanie nazwy obiektu a po kropce nazwę metody i przypisując do niej nazwę funkcji.

```
obiekt.nazwametody=nazwafunkcji
```

Słowo kluczowe `this` służy do odwoływania się do aktualnego obiektu.

Predefiniowane obiekty:

- Array metody: concat, join, pop, reverse, shift, slice, splice, sort, unshift
- Boolean
- Date metody: set, get, to, parse
- Function object
- Math metody: abs, sin, cos, tan, asin, acos, atan, exp, log, ceil, floor, min, max, pow, round, sqrt
- Number object

- RegExp object metody: anchor, link, concat, indexof ...

Skrypt można dołączyć do dokumentu HTML przez dodanie tagu <SCRIPT>, przez podanie nazwy pliku z kodem źródłowym, przez podanie wyrażenia dla wartości atrybutu w HTMLu lub jako funkcje przechwytyjące zdarzenia.

Za pomocą JavaScript można przechwytywać zdarzenia. W tym celu należy zdefiniować funkcje, która się wykona podczas zaistnienia zdarzenia. Następnie wywołać ją podczas zaistnienia zdarzenia. W tym celu dodaje się wywołanie funkcji w znaczniku INPUT. Przykładowe wywołanie:

```
<INPUT TYPE="button"
VALUE="Calculate"onClick="compute(this.Form)">
```

Tab. 1 Zdarzenia

Zdarzenie	Dotyczy	Wywołane zostaje	Funkcja przechwytyjąca
Abort	images	Użytkownik przerywa ładowanie obrazka.	onAbort
Blur	windows and all form elements	Użytkownik zmienia aktywne okno lub pole formularza	onBlur
Change	text fields, textareas, select lists	Użytkownik zmienia wartość elementu	onChange
Click	buttons, radio buttons, checkboxes, submit buttons, reset buttons, links	Użytkownik klika formularz lub link.	onClick
DragDrop	windows	Użytkownik przenosi i upuszcza obiekt w oknie przeglądarki	onDragDrop
Error	images, windows	Ładowanie obrazu lub dokumentu powoduje błąd	onError
Focus	windows and all form elements	Użytkownik aktywuje okno lub pole w formularzu	onFocus
KeyDown	documents, images, links, text areas	Użytkownik naciska klawisz	onKeyDown
KeyPress	documents, images, links, text areas	Użytkownik nacisnął przycisk	onKeyPress

KeyUp	documents, images, links, text areas	Uzytkownik puszcza klawisz	onKeyUp
Load	document body	Uzytkownik laduje strone	onLoad
MouseDown	documents, buttons, links	Uzytkownik nacisnal klawisz myszy	onMouseDown
MouseMove	nothing by default	Uzytkownik przesuwa kursor myszy	onMouseMove
MouseOut	areas, links	Uzytkownik przesunal kursor myszy poza obszar rysunku	onMouseOut
MouseOver	links	Uzytkownik przesunal kursor nad link	onMouseOver
MouseUp	documents, buttons, links	Uzytkownik puscil klawisz myszy	onMouseUp
Move	windows	Uzytkownik przesuwa lub skrypt przesuwa ekran	onMove
Reset	forms	Uzytkownik wyczyscil formularz	onReset
Resize	windows	Uzytkownik lub skrypt zmienil rozmiar okna	onResize
Select	text fields, textareas	Uzytkownik wybiera element z pola formularza	onSelect
Submit	forms	Uzytkownik potwierdza formularz	onSubmit
Unload	document body	Uzytkownik opuszcza strone	onUnload

Mozna zarejestrowac takze funkcje, która bedzie obslugiwala dane zdarzenie. W tym celu trzeba włączyc przechwytywanie danego zdarzenia przez wywołanie funkcji `window.captureEvents(Event.CLICK)`, zdefiniowac funkcje, a nastepnie zarejestrowac przez przypisanie wartosci `window.onClick=nazwafunkcji`.

Za pomoca Navigator objects mozna poznac dane o dokumencie. Za ich pomoca mozna pobrac dane z formularzy, zmieniac wartosci niektórych elementów np. `document.title="Nowy tytul"`. Kazda strona ma swój obiekt. Kazdy formularz takze ma swój obiekt. Obiekt `location` zawiera dane zwiazane z adresem URL. Obiekt `history` zawiera strony jakie uzytkownik odwiedzil. Obiekt `navigator` pozwala na sprawdzenie czy jest włączona Java i pozwala ustawiac wiele wlasciwosci.

Mozna użyć metody `write` obiektu `document` w celu napisania w HTMLu jakiegoś tekstu, który zostanie wyświetlony.

JavaScript pozwala tworzyć i manipulować oknami i ramkami w celu wyświetlania dokumentów HTML. Metoda `window.open` pozwala na otwarcie nowego okna. Jako jeden z argumentów podaje się nazwę pliku zawierającego dokument HTML. Do zamykania okna służy metoda `close()`, która zamyka albo aktualne okno albo, jeżeli podamy uchwyt do okna to zamyka te okno.

JavaScript pozwala na używanie plików cookies. W tym celu pole `document.cookie` są przypisywane wartości wraz z nazwami.

W związku z bezpieczeństwem stosuje się różnego rodzaju ograniczenia. W tym celu sprawdza się czy dany skrypt pochodzi z danej domeny i jeżeli nie to nie jest wykonywany. Wykonuje się także skrypty z zaufanych serwerów. Skrypty są także podpisywane.

VBSCRIPT

Pełna nazwa VBScript to Microsoft Visual Basic Scripting Edition. Jest uproszczona wersja języka Visual Basic i Visual Basic for Applications. Został zaprojektowany tak, żeby być podobny do języka BASIC.

Język VBScript jest „środowiskiem skryptowym”, które urozmaica strony HTML przez dodanie do nich interakcji w porównaniu do stron statycznych. VBScript został stworzony w celu dodania języka skryptowego do MS Internet Explorer'a. Wykorzystanie VBScript po stronie klienta pozwala na odciążenie serwera, gdyż skrypt wykonywany jest przez przeglądarkę. Dodatkową zaletą jest także to, że dane wpisywane przez użytkownika nie są przysyłane do serwera za każdym razem, kiedy użytkownik błędnie je wpisze sprawdzane są tylko na komputerze klienta.

Język VBScript używa do komunikowania się z aplikacjami Windows Script. Przy pomocy Windows Script przeglądarki i inne aplikacje nie potrzebują dodatkowych własności dla każdego komponentu skryptującego. Windows Script zajmuje się kompilowaniem skryptów, otrzymuje i wywołuje punkty styku i zarządza zasobami dostępnymi dla developerów. Obecnie jest opracowywany standard dla tego języka.

Język ten został zaprojektowany jako dodatek do strony HTML. Przeglądarka otrzymuje skrypt razem z całą stroną. Wykonanie skryptu spoczywa na przeglądarce. Skrypty dołączane są tak samo jak w języku JavaScript za pomocą tagu `<SCRIPT>`. Jedną ze zmiennych tego znacznika jest `LANGUAGE`, który informuje przeglądarkę, jaki to jest język skryptowy.

Nie wszystkie przeglądarki wspierają język VBScript, dlatego należy skrypt potraktować jako komentarz w języku HTML. W innym przypadku przeglądarka mogłaby po prostu wyświetlić skrypt jako zwykły tekst.

W języku VBScript zostały zdefiniowane grupy stałych:

- kolory – definiują osiem podstawowych kolorów
- czas i data
- formatowanie daty
- ogólne
- okna dialogowego

- string
- tristate
- VarType

Własne stałe można deklarować za pomocą słowa kluczowego Const i podając nazwę zmiennej i jej wartość.

Zmienne są nazwanym obszarem pamięci służącym do przechowywania danych podczas wykonywania skryptu. Można je użyć do przechowywania danych pobranych od użytkownika za pomocą strony WWW, zapisania wyniku zwracanego przez funkcję lub zapisania wyniku obliczeń. Są dwie metody deklarowania zmiennych: dokładna i niedokładna. Pierwszy sposób polega na deklaracji zmiennej z użyciem słowa kluczowego Dim. Można w ten sposób zadeklarować wiele zmiennych w jednej linii. Drugi sposób polega po prostu na użyciu zmiennej w skrypcie, co w praktyce nie jest polecane, gdyż program staje się mniej czytelny i trudniejsze staje się do wychwycenia błędów. Można wymusić, aby wszystkie zmienne w VBScript były deklarowane dokładnie, dodając instrukcję Option Explicit na początku każdego skryptu. Każda zmienna zadeklarowana niedokładnie spowoduje błąd. Przykładowa deklaracja zmiennej:

```
Dim nazwa_zmiennej;
```

Nazywając zmienne należy zwracać uwagę na następujące zasady:

- każda zmienna powinna się zaczynać od litery;
- nie mogą zawierać spacji;
- muszą być unikalne;
- nie mogą być większe niż 255 znaków.

VBScript posiada typ danych nazywany variant. Variants pozwalają przechowywać dane różnego typu. Typy danych, jakie mogą być przechowywane przez variant to podtypy. Tabela poniżej przedstawia podtypy wspierane przez VBScript

Tab. 2 Podtypy

Podtyp	Opis użycia podtypów
Byte	Liczby całkowite od 0 do 255
Boolean	<i>True</i> i <i>False</i>
Currency	Wartosci monet
Date	Data i czas
Double	Bardzo duze liczby zmiennoprzecinkowe
Empty	Wartosc uzywana gdy nie ma zainicjowanej wartosci
Error	Numer bledu
Integer	Liczby całkowite od -32,768 do 32,767
Long	Bardzo duze liczby całkowite (-2,147,483,648 i 2,147,483,647)
Object	Obiekty
Null	Zadna poprawna wartosc
Single	Duze liczby zmiennoprzecinkowe
String	Znaki

Zmienne posiadaja zasieg. Jezeli sa zadeklarowane wewnatrz jakiejś procedury, wtedy odwoływac sie do nich mozna na poziomie danej procedury. Jezeli natomiast zmienna została zadeklarowana poza jakakolwiek procedura wtedy mozna sie do niej odwoływac z poziomu skryptu.

Tablice w VBScript deklaruje sie tak samo jak zmienna tyle, ze po nazwie dodatkowo podaje sie w nawiasach okraglych numer indeksu ostatniego elementu. W VBScript indeksy tablicy zaczynaja sie od zera. Wartosc elementowi tablicy przypisuje sie przez podanie nazwy tablicy i w nawiasach okraglych indeksu elementu. Tablic moga miec wiele wymiarów. VBScript wspiera do 60 wymiarów. Deklarujac tablice n wymiarowa dodaje sie wewnatrz nawiasów okraglych kolejne indeksy po przecinku. Odwołanie do elementów tej tablicy realizuje sie poprzez podanie w nawiasach okraglych indeksów po przecinku. VBScript pozwala takze na zmiane rozmiaru tablicy podczas wykonywania skryptu. Te tablice to tak zwane tablice dynamiczne. Deklaruje sie je bez okreslania wielkosc. Instrukcja ReDim jest uzywana do zmiany wielkosc tablicy. Podaje sie wtedy nowy rozmiar tablicy. Nie ma ograniczenia na to ile razy moze byc wielkosc tablicy zmieniona. Zeby zabezpieczyc zawartosc tablicy nalezy uzyc slowa kluczowego Preserve.

Zeby dodac do formularza funkcje, która bedzie odpowiadala na zdarzenia wywoływane przez uzytkownika, nalezy zmiennej NAME w znaczniku INPUT nadac nazwe funkcji. Jednak sama nazwa funkcji nie sklada sie tylko nazwy, po podkresleniu nalezy dodac nazwe zdarzenia, jakie ma obslugiwac. Aby zdefiniowac funkcje nalezy uzyc slowa kluczowego Sub. Do wartosci pól mozna sie odwoływac poprzez kolejne w hierarchi dokumentu. Przykład odwołania sie do wartosci pola w formularzu.

```
document.frmName.txtName.value
```

Komentarze w VBScript są poprzedzane znakiem apostrofu. Są one ignorowane podczas wykonywania skryptu. Mogą się pojawić na początku linii jak i na końcu.

Podstawowymi operatorami używanymi najczęściej w VBScript są: + dla dodawania, - dla odejmowania, * dla mnożenia i / dla dzielenia.

Do wyświetlania komunikatów służy funkcja MsgBox. Do wyświetlania można także używać kontrolek ActiveX

Obiekty, i w apletach Javy i kontrolkach ActiveX, pozwalają zwiększyć funkcjonalność HTMLa. Używając VBScript można poszerzyć możliwości tych kontrolek, integrując i manipulując nimi z poziomu skryptu. Na dodanie obiektów do stron składają się dwa kroki. Po pierwsze należy dodać obiekt do strony. Po drugie należy napisać procedurę w skrypcie, która będzie odpowiadała na zdarzenia związane z obiektem.

Obiekty, zarówno apletów Javy jak i kontrolek ActiveX są dodawane za pomocą znacznika <OBJECT>. Właściwości lub cechy charakterystyczne obiektu są konfigurowane przez znacznik <PARAM>. Zwykle jest zaimplementowany jeden obiekt wraz z kilkoma znacznikami <PARAM>.

Raz dodając kontrolkę do strony może być konfigurowana, zmieniana i może odpowiadać przez właściwości, metody i zdarzenia. Właściwości są charakterystyczne dla obiektu. Zawierają elementy takie, jak opis, kolor tła i rozmiar czcionki. Metody powodują wykonanie pewnych zadań. Zdarzenia są akcjami rozpoznawanymi przez obiekt. Na przykład przycisk rozpoznaje zdarzenie onclick.

VBScript pozwala na kontrolowanie przepływu danych podczas działania skryptu przez instrukcje warunkowe i pętle. Używając instrukcji warunkowych można wykonywać operacje na danych w zależności od zadanych kryteriów. Pętle pozwalają na wielokrotne wykonanie kawałka kodu skryptu. Dzięki temu można projektować bardziej złożone strony.

Instrukcje warunkowe:

- 1) If ... Then ... Else ... End If

Instrukcja ta jest używana na początku w celu sprawdzenia warunku czy jest prawdziwy czy fałszywy, następnie, w zależności od warunku, wykonywane są instrukcje. Jeżeli mamy kilka instrukcji wtedy dodajemy End If w celu sygnalizacji końca instrukcji związanych z danym warunkiem. Można także użyć instrukcji Else If w celu sprawdzenia dodatkowego warunku.

- 2) Select ... Case

Instrukcja ta jest alternatywa dla instrukcji If, pozwala na dodatkową kontrolę warunków i zwiększa czytelność, kiedy mamy doczynienia ze złożonymi warunkami. Jest ona bardzo przydatna w przypadku, kiedy trzeba sprawdzić wiele warunków. Podobnie jak If Select sprawdza warunki bazując na tym wykonuje instrukcje, jeżeli wynikiem warunku była prawda. Select sprawdza wszystkie warunki dopóki nie napotka tego, którego wynikiem jest prawda. Jeżeli żaden nie będzie prawdziwy wykonuje instrukcje, które znajdują się po Case Else.

VBScript ma cztery pętle, które można podzielić na dwie grupy. Jedną to pętle wykonywane określoną ilość razy. Drugą grupą nie ma zdefiniowanej ilości wykonanych pętli.

- 1) For ... Next

Jest używana do wykonywania pętli z góry zaplanowaną ilość razy. Używa licznika, który jest zwiększany lub zmniejszany w zależności od kierunku pętli z każdym wykonaniem pętli. Standardowa wartość o którą zmieniany jest licznik to jeden. Można określić ją stosując słowo kluczowe Step.

2) For Each ... Next

Jest podobna do zwykłego For tyle, że jest wykonywana dla każdego elementu z podanego zbioru.

3) Do ... Loop

Wykonuje bloki instrukcji dopóki określony warunek nie zostanie spełniony. Zwykle sprawdzanym warunkiem jest wynik operacji przeprowadzanych w strukturze pętli. Są dwie wersje tej pętli:

a) Do ... While

Pętla będzie się wykonywała dopóki sprawdzany warunek nie będzie prawdziwy. Może być on sprawdzany na początku i na końcu pętli. Różnica polega na tym, że dla pierwszego warunku pętla może się nie wykonać ani razu, natomiast dla drugiego wykona się przynajmniej raz.

b) Do ... Until

Pętla będzie się wykonywać dopóki warunek jest fałszywy. Warunek może być sprawdzany na początku albo na końcu.

4) While ... Wend

Wykonuje się dopóki warunek jest prawdziwy. Jeżeli warunek jest prawdziwy pętla działa podobnie do pętli Do ... Loop jednak bez jej elastyczności.

Coraz częściej są używane formularze dlatego jest potrzeba sprawdzania poprawności danych wprowadzanych przez użytkownika zanim zostaną one przesłane do serwera. Jako język skryptowy VBScript bardzo dobrze się nadaje do tego. Jak tylko dane zostały sprawdzone mogą zostać w tym samym skrypcie przesłane do serwera. Proces sprawdzenia poprawności danych składa się ze sprawdzenia czy wszystkie dane są wprowadzone i czy są wprowadzone poprawnie. Skrupulatne sprawdzanie danych za pomocą skryptu może być nudne, ale jest warte ze względu na poprawność danych.

W VBScript są dwa rodzaje procedur: procedura Sub i procedura Function. Procedura Sub jest zbiorem instrukcji. Znajdują się one między instrukcjami Sub i End Sub. Po wywołaniu procedury wykonywane są instrukcje, które znajdują się wewnątrz niej, nie jest jednak zwracana żadna wartość. Natomiast funkcja zaczyna się od słowa kluczowego Function potem występują instrukcje, funkcja kończy się instrukcją End Function. Funkcja jest podobna do procedury z tą różnicą, że funkcja zwraca wartość. Funkcje jak i procedury mogą pobierać argumenty. W funkcjach, jeżeli nie ma żadnych parametrów należy podać listę pustych argumentów. Funkcja zwraca wartość przypisując ją do swojej nazwy. Typem zwracanym jest zawsze **Variant**. Dane do procedury mogą być przekazywane za pomocą argumentów. Kolejne argumenty są oddzielane od siebie za pomocą przecinka. Funkcje wywołuje się zwykle po prawej stronie przypisania. Procedury wywołuje się podając jej nazwę wraz z jej argumentami lub za pomocą funkcji Call.

VBScript posiada szeroki zakres operatorów. Kiedy kilka operacji jest przeprowadzanych w wyrażeniu, każda część jest wyliczana w oparciu o priorytet operatorów. Za pomocą nawiasów można zmieniać priorytety. Obliczenia wykonywane w nawiasach zawsze mają wyższy priorytet od innych. Jeżeli w wyrażeniu są użyte operatory z różnych grup wtedy najpierw wykonywane są operatory arytmetyczne, następnie operatory porównania, a na końcu operatory logiczne.

Operatory porównania wykonywane są od lewej do prawej. W VBScript występują następujące operatory:

- arytmetyczne: ^,-,*,/, \,Mod,+,&
- porównania: =,<,>,<=,>=,Is
- logiczne: Not,And,Or,Xor,Eqv,Imp

Ze względów bezpieczeństwa VBScript nie pozwala na użycie funkcji CreateObject(), gdyż stworzenie obiektu mogłoby być użyte w celu zniszczenia zasobów komputera klienta. Bezpieczeństwo może zostać naruszone na pozwolenie wykonania się kontrolki, które są automatycznie ściągane z sieci i rejestrowane na komputerze klienta. Aby się zabezpieczyć przed tym można wyłączyć je opcjach przeglądarki.

ACTIVEX

W dzisiejszych czasach dane są przechowywane w różny sposób, poczynając od baz danych do dokumentów Worda, wiadomości elektronicznych i innych. ADO ułatwia komunikację między różnymi formatami danych. Producenci oprogramowania nie muszą myśleć o specyficznych wywołaniach funkcji API lub innych niuansów związanych z przechowywaniem danych. Dzięki ADO prawie wszystkie dane są dostępne w standaryzowany sposób. ActiveX jest nazwą zbioru usług i technologii bazujących na COM. Kontrolki ActiveX używają technologii COM w celu komunikacji się z innymi kontrolkami COM i usługami. ActiveX są trzecia wersja kontrolki OLE. Zostały zaprojektowane w celu dodania paru unowocześnień przeważnie do przesyłania komponentów przez sieć i do dodania kontrolki do przeglądarek. Jednym z dodatków jest podpisywanie kodu, żeby użytkownik wiedział, kto jest autorem kontrolki, zanim ją wykona.

Do obiektów ADO należą:

1) Command

Ten obiekt jest używany do potwierdzania i wykonywania zapytań w bazie danych. Zapytanie może przeprowadzić różne rodzaje operacji, takich jak dodawanie, usuwanie, tworzenie, otrzymywanie i modyfikowanie informacji w bazie danych. Jeżeli zapytanie jest używane do uzyskania informacji z bazy danych, dane zostaną zwrócone przez obiekt RecordSet. Oznacza to, że otrzymane dane można zmieniać za pomocą 64 właściwości, metod, kolekcji i zdarzeń, które należą do obiektu RecordSet. Jedną z głównych zalet obiektu Command jest to, że pozwala na użycie zachowanych zapytań i procedur, które akceptują argumenty. To wymaga dostępu do kolekcji Parameter, która jest unikalna dla obiektu Command. Można także użyć metody Execute obiektu Connection lub metody Open obiektu RecordSet w celu wykonania zapytania do bazy danych. Jednak nie posiadają one takiej wszechstronności jak metody, właściwości, kolekcje i zdarzenia znajdujące się w obiekcie Command.

Właściwości:

ActiveConnection.
CommandStream
CommandText
CommandTimeout
CommandType
Dialect
Name.
NamedProperties

Prepared
State

Kolekcje:

Parameters

Property.

Metody:

Cancel.

CreateParameter

Execute

2) Connection

Pozwala na pobranie otwartego polaczenia do danych. Przez to polaczenie mozna uzyskac dostep i zmieniac dane w bazie danych. Zeby wywolac zapytanie do bazy danych nie trzeba jawnie tworzyć obiektu Connection. Polaczenie moze być utworzone przez przekazanie stringu polaczenia do obiektu Command lub RecordSet. Jakolwiek takie polaczenie jest dobre w przypadku specyficznego, pojedynczego zapytania. Przy wielokrotnym dostepie do bazy bardziej efektywne jest nawiązanie polaczenia uzywając obiektu Connection. Mozna wywolac zapytanie uzywając metody Execute obiektu Connection. Tylko obiekty Connection i RecordSet obslugują zdarzenia. Przechwytywac zdarzenia jednak mozna tylko za pomoca Visual Basic, Visual C++, Visual J++.

Parametry:

Attributes

CommandTimeout

ConnectionString

ConnectionTimeout

CursorLocation

DefaultDatabase

IsolationLevel

Mode

Provider

State

Version

Collections:

Errors

Properties

Metody:

BeginTrans

Cancel

Close

CommitTrans

Execute

Open

OpenSchema

RollbackTrans

Zdarzenia:

BeginTransComplete

CommitTransComplete

ConnectComplete

Disconnect

ExecuteComplete
InfoMessage
RollbackTransComplete
WillConnect
WillExecute

3) Error

Zawiera kompletne informacje na temat błędów, jakie wystąpiły podczas dostępu do danych lub ostrzeżeń, jakie zostały wykonane podczas pojedynczej operacji. Kiedy pojawi się błąd oprogramowanie jest odpowiedzialne za przekazanie informacji o błędzie do ADO. Za każdym razem, gdy wystąpi błąd lub ostrzeżenie ADO tworzy nowy obiekt Error, który zawiera dokładne dane na temat błędu. Każdy z obiektów Error jest przechowywany w kolekcji Errors, która jest unikalną kolekcją dla obiektu Connection. Żeby uzyskać dostęp do poszczególnych błędów, należy się odwołać do konkretnego połączenia.

Właściwości:

Description
HelpContext
HelpFile
NativeError
Number
Source
SQLState

4) Field

Zawiera informacje o pojedynczym polu (kolumnie) znajdującym się w obiekcie RecordSet. Obiekt RecordSet składa się z kolekcji zero lub więcej obiektów Field. Ta kolekcja nazywa się Fields Collection. Tylko obiekty Record i RecordSet zawierają tę kolekcję.

Właściwości:

ActualSize
Attributes
DataFormat
DefinedSize
Name
NumericScale
OriginalValue
Precision
Status
Type
UnderlyingValue

Kolekcje:

Properties

Metody:

AppendChunk
GetChunk

5) Parameter

Zawiera dokładne informacje na temat pojedynczego parametru używanego w zapisanych procedurach i zapytaniach. Parametry są używane w celu stworzenia sparametryzowanych komend. Te komendy po tym jak zostały zdefiniowane i

zainicjalizowane, używają parametrów, aby zmienić pewne detale w tekście komendy zanim zostaną wykonane. Za każdym razem jak jest tworzony obiekt Parameter jest on dodawany do kolekcji Parameters i jest wiązany z obiektem Command. Obiekt Command używa kolekcji Parameters do przekazywania parametrów z i do zapisanych procedur lub zapytań. Są cztery główne typy zapytań: input, output, input/output i return.

Właściwości:

Attributes
Direction
Name
NumericScale
Precision
Size
Type
Value

Kolekcje:

Parameters
Properties

Metody:

AppendChunk

6) Property

Zawiera informacje na temat specyficznych właściwości oprogramowania. ADO potrafi łączyć się z bazami danych różnych producentów. Każde oprogramowanie będzie komunikować się z ADO w nieco odmienny sposób. Dlatego ADO potrzebuje informacji na temat oprogramowania. Rozwiązaniem jest podawanie przez oprogramowanie specyficznych informacji, nazywanych dynamicznymi właściwościami, dla ADO. ADO zapisuje te dane w obiekcie Property, a następnie każdy obiekt zachowuje w kolekcji Properties, która jest związana z obiektem ADO. Cztery obiekty ADO mogą mieć kolekcje Properties: Command, Connection, Field i RecordSet. Zdolność do elastycznego dopasowywania się do oprogramowania sprawiają, że ADO jest bardziej użyteczne.

Właściwości:

Attributes
Name
Type
Value

7) Record

Może przechowywać wiersz z obiektu RecordSet, katalog lub plik z dysku. Przed dodaniem obiektu Record, można było za pomocą ADO tylko uzyskać dostęp do strukturalnych baz danych. W strukturalnych bazach danych, każdy wiersz ma tę samą liczbę kolumn i każda kolumna zawiera ten sam typ danych. Innymi słowami struktura każdego wiersza w bazie danych jest taka sama. Obiekt Record zwiększa użyteczność ADO pozwalając na dostęp do zbiorów danych, gdzie liczba kolumn i/lub typ danych mogą się zmieniać w wierszach. Na przykład może przechowywać zbiory danych przechowywane w bazie o strukturze drzewa. Pola związane z obiektem Record mogą być przeglądane za pomocą kolekcji Fields.

Właściwości:

ActiveConnection
Mode
ParentURL

RecordType

Source

State

Kolekcje:

Fields

Properties

Metody:

Cancel

Close

CopyRecord

DeleteRecord

GetChildren

MoveRecord

Open

8) RecordSet

Używany jest do przechowywania danych otrzymanych z bazy danych. Obiekt RecordSet jest zbudowany z rekordów i pól. Obiekt ten jest najważniejszym obiektem ADO. Za pomocą tego obiektu możemy wybrać dane i zmienić je dodając, usuwając lub modyfikując. Równie istotne jest poruszanie się wewnątrz bazy danych. Obiekt RecordSet ma bardzo obszerny zbiór właściwości, metod, kolekcji i zdarzeń, które pozwalają na szeroką manipulację danymi i interpretację środowiska wykonania. Jednak pewne ograniczenia mogą być spowodowane przez oprogramowanie. Na przykład niektóre właściwości mogą być niedostępne w zależności od używanego oprogramowania. Można użyć metody Supports, która sprawdza czy obiekt RecordSet będzie miał specyficzną funkcjonalność.

Właściwości:

AbsolutePage

AbsolutePosition

ActiveCommand

ActiveConnection

BOF

Bookmark

CacheSize

CursorLocation

CursorType

DataMember

DataSource

EditMode

EOF

Filter

Index

LockType

MarshalOptions

MaxRecords

PageCount

PageSize

RecordCount

Sort

Source
State
Status
StayInSync
Kolekcje:
Fields
Properties
Metody:
AddNew
Cancel
CancelBatch
CancelUpdate
Clone
Close
CompareBookmarks
Delete
Find
GetRows
GetString
Move
MoveFirst
MoveLast
MoveNext
MovePrevious
NextRecordset
Open
Requery
Resync
Save
Seek
Supports
Update
UpdateBatch
Zdarzenia:
EndOfRecordset
FetchComplete
FetchProgress
FieldChangeComplete
MoveComplete
RecordChangeComplete
RecordsetChangeComplete
WillChangeField
WillChangeRecordset
WillMove

- 9) Stream
Daje dostęp do strumienia danych binarnych lub tekstu. Dostęp tu oznacza zdolność do pisania, czytania i manipulowania strumieniem. Na przykład można użyć obiektu Record lub Recordset, żeby uzyskać dostęp do plików na serwerze, a potem użyć obiektu Stream,

zeby uzyskac do nich dostep i zmieniac ich zawartosc. Sa trzy podstawowe sposoby pozwalajace na dostep do obiektu Stream: za pomoca URL wskazujacego plik, folder, lub obiekt Record; tworzac obiekt Stream do przechowywania danych; otwierajac domyslny obiekt Stream zwiazany z obiektem Record.

Wlasciwosci:

CharSet
EOS
LineSeparator
Mode
Position
Size
State
Type

Metody:

Cancel
Close
CopyTo
Flush
LoadFromFile
Open
Read
ReadText
SaveToFile
SetEOS
SkipLine
Write
WriteText

Obiekty ActiveX dzieki temu, ze standaryzuja dostep do danych, sa bardzo przydatne. Poza tym obiekty te mozna tworzyc za pomoca róznych jezyków oprogramowania, dzieki czemu sa bardzo wszechstronne. Moga wykonywac rózne dzialania, które daja sie zdefiniowac w danym jezyku. W zwiazku z tym daja szeroki dostep do komputera klienta w odróznieniu do Javy, która jest zamknieta w odrebnyim srodowisku. Dlatego ze wzgledów bezpieczenstwa takie programy powinny byc podpisywane. Nie wszystkie obiekty sa podpisywane, wiec jezeli przegladarka pozwala na uruchamianie wszystkich obiektów ActiveX to moze to spowodowac, ze zostanie takze uruchomiony zlosliwy obiekt. Najczesciej trudno jest okreslic potem, który element ActiveX byl sprawca ataku.

APLETY JAVY

Aplety Javy sa niezalezne od platformy i systemu operacyjnego. Kompilator Javy generuje tak zwany kod bajtowy przeznaczony dla maszyny wirtualnej Javy. Jest ona zaimplementowana dla wiekszosci popularnych platform. Java jest jezykiem zorientowanym obiektowo. Po wydaniu polecenia zaladowania pliku z kodem bajtowym JVM uruchamiana jest na docelowym komputerze, gdzie wczytuje plik i wykonuje okreslone w nim dzialania. Dzieki apletom strony HTML moga stac sie aktywne. Aplety Javy umieszcza sie na stronie za pomoca znacznika <APPLET>. Poniewaz Java dziala na wszystkich popularnych platformach, aplety sa wyswietlane

i działają w pożądanym sposób zawsze wtedy, gdy użytkownik odwiedzający daną stronę Internetu korzysta z jednej z kompatybilnych z Java przeglądarek.

Aplety działają w środowisku graficznym. Żeby utworzyć aplet należy stworzyć klasę dziedziczącą po klasie Applet i przykrywającą jej wszystkie metody, których działanie nie jest zgodne z wymaganiami stawianymi apletowi. Ponieważ aplet wyświetlany jest w oknie przeglądarki. Za każdym razem, kiedy obszar ten zostanie zakryty przez inne okno lub zniknie, staje się on niewidoczny. Po ponownym wyświetleniu aplet musi odświeżyć jego zawartość. Z technicznego punktu widzenia wygląda to tak, że obiekt klasy Applet wywołuje metodę repaint(). Metoda ta musi być wywołana za każdym razem, kiedy konieczne jest wyświetlenie czegoś w obszarze graficznym apletu.

Obszar graficzny utworzony przez przeglądarkę jest reprezentowany przez obiekt Graphics, tak więc działanie metody paint zaczyna się od jego pobrania. Do klasy Graphics należy około czterdziestu metod publicznych, w tym drawLine(), draw3DRect() i fillArc(). Do wyświetlania tekstu służy metoda drawString(). Pobiera ona trzy parametry: łańcuch, który ma zostać wyświetlony oraz współrzędne x i y punktu, w którym rozpoczyna się jego wyświetlanie.

Dla apletów można określić cykl życia. Każda klasa zawiera konstruktor. W konstruktorze umieszcza się kod inicjalizacyjny. Jest on wywoływany raz podczas istnienia apletu. Metoda init() jest wywoływana po konstruktorze. Jest ona także wykonywana raz podczas istnienia apletu. Po załadowaniu i zainicjalizowaniu apletu środowisko wykonywania programów Javy wywołuje metodę start(). Natomiast zmiany strony lub jej zminimalizowanie powoduje wywołanie metody stop(). Powrót do tej samej strony jest związany z ponownym wywołaniem start(). Umieszcza się w nich instrukcje, które powinny być wykonane po wyświetleniu lub opuszczeniu strony. Środowisko wykonywania programów Javy wywołuje metodę paint() za każdym razem, kiedy istnieje podejrzenie, że obszar graficzny apletu został odsłonięty. Dlatego ta metoda jest wywoływana dużo częściej niż można by było przypuszczać. Wykonywanie tej metody zajmuje większość czasu działania programu.

Gdy zaczyna brakować pamięci lub jest zamykana przeglądarka zasoby apletu są zwalniane. Tuż przed zwolnieniem przydzielonej apletowi pamięci wywoływana jest metoda destroy(). Używa się jej do zwolnienia wszystkich zasobów, które zostały wcześniej apletowi przydzielone.

W Javie są dostępne następujące typy proste:

- byte
- short
- int
- long
- float
- double
- char
- boolean

W Javie mamy dostępne następujące operatory:

- przypisania: =, +=, -=, *=, /=
- porównania: ==, <, <=, >, >=
- jednoargumentowe operatory znaku

- arytmetyczne: +,*,/,-
- inkrementacji i dekrementacji
- przesunięcia bitowego
- bitowej negacji logicznej
- bitowe: OR,AND,XOR
- logiczne: &&, !, ||

Tablice w Javie tworzy się za pomocą konstruktora podając jako parametr rozmiar tablicy. Do wartości w tablicy odwołuje się za pomocą podanie indeksu. Przykład:

```
short tab[]=new short[100];
tab[5]=12;
```

W Javie mamy szereg predefiniowanych klas. Jedną z nich jest klasa String pozwalająca na manipulację łańcuchami znaków. Istnieją także klasy odpowiadające prostym typom danych. Można także stworzyć własną klasę, która będzie zawierać metody i atrybuty. Definicja klasy zaczyna się od modyfikatora, który określa czy dana klasa jest publiczna, czyli dostępna dla wszystkich, private, protected i inne. Następnie podaje się słowo kluczowe class i nazwa klasy. Po nazwie może być podana nazwa klasy, z której klasa dziedziczy po słowie extends. W Javie nie ma wielodziedziczenia. W zamian za to są interfejsy. W celu dodania interfejsu należy po słowie kluczowym implements podać nazwę interfejsu. Wewnątrz klasy definiuje się metody. Są to funkcje, które realizują pewne zdarzenia związane z klasą. W klasie znajdują się także dane. Są to informacje specyficzne dla danej klasy.

W Javie nie ma destruktorów. Nie zwalnia się także pamięci. Tym zajmuje się moduł czyszczenia pamięci. Można także samemu wyczyścić w metodzie finalize().

W Javie dostępne są instrukcje rozgałęzienia. Warunek wyliczony musi być true albo false. Jeżeli w klauzuli if ma być wykonanych więcej instrukcji, to trzeba je umieścić w nawiasach klamrowych.

```
if(warunek)
    instrukcja-1;
[else
    instrukcja-2;
]
```

W Javie mamy także instrukcje switch, która jest używana zamiast używania złożonych instrukcji if.

```
switch(warunek)
{
    case wartosc-1:
        instrukcja-1;
    ...
    default:
        instrukcja-2;
}
```

Java umożliwia przekazywanie sterowania z jednego miejsca programu w inne na cztery różne sposoby. Mimo, że nie wprowadzają one rozgałęzień, to jednak są często stosowane z instrukcjami rozgałęzień oraz z instrukcjami pętli. Instrukcje skoku mają następujące nazwy:

- break
- continue
- return
- throw.

W Javie mamy trzy instrukcje iteracyjne. Pierwsza z nich to while, która się wykonuje po sprawdzeniu warunku. Następna to także while tyle, że wykonuje się przynajmniej raz gdyż warunek jest sprawdzany na końcu. Pętla for działa określona z góry liczbę razy.

W Javie istnieje bardzo rozbudowana obsługa wyjątków. Są one obsługiwane za pomocą bloków try, w którym umieszcza się niepewny kod, który może wygenerować wyjątek, następnie mamy blok catch, który przechwytuje wyjątek i znajdują się w niej instrukcje, jakie mają być wykonane w chwili zaistnienia wyjątku. Można dodać jeszcze jeden blok finally, w którym można posprzątać pamięć i wykonać czynności końcowe.

W Javie mamy pakiet, w którym mamy zdefiniowane podstawowe elementy interfejsu użytkownika. Ten pakiet to AWT. Zdefiniowane są w nim następujące klasy:

- Buttons (Button)
- Checkboxes (Checkbox)
- Single-line Text fields (TextField)
- Larger text display and editing area (TextArea)
- Labels (Label)
- Lists (List)
- Pop-up list of choices (Choice)
- Sliders and scrollbars (Scrollbar)
- Drawing area (Canvas)
- Menu (Menu, MenuItem, CheckboxMenuItem)
- Containers (Panel, Window i jego podklasy)

API apletów pozwala na powiadomienie przez przeglądarkę o stanie, w którym się znajduje. Można ładować pliki z danymi podając URL apletu lub strony, na której jest uruchamiany, aby na przykład załadować plik graficzny lub muzyczny. Pozwala także wyświetlać krótki tekst informujący o statusie. Dzięki temu można wyświetlać informacje o tym, co wykonuje aplet. Można także wymusić na przeglądarce wyświetlenie dokumentu. Można także znaleźć inne applety uruchomione na tej samej stronie. Można także odtworzyć plik muzyczny lub pobrać parametry podane przez użytkownika.

Żeby dowiedzieć się czegoś na temat środowiska, w którym jest wykonywany aplet mogą one czytać właściwości systemu. Właściwości systemu są parami składającymi się z nazwy i wartości zawierające dane na przykład o systemie operacyjnym i architekturze. Nie wszystkie właściwości aplet może przeczytać. Aplet może przeczytać następujące właściwości:

Tab. 3 Wlasciwosci systemu

Klucz	Znaczenie
"file.separator"	Separator plików
"java.class.version"	Numer wersji klas Javy
"java.vendor"	Nazwa dostawcy Javy
"java.vendor.url"	URL dostawcy Javy
"java.version"	Numer wersji Javy
"line.separator"	Separator linii
"os.arch"	Architektura systemu operacyjnego
"os.name"	Nazwa systemu operacyjnego
"path.separator"	Separator sciezek

Aby odczytac te wlasciwosci uzywa sie metody klasy System getProperty.

Kazdy aplet moze tworzyć wiele watków. Metody rysujące są zawsze wołane z watku AWT rysującego i przechwytyjacego zdarzenia. Watki wywołujące metody: init(), start(), stop() i destroy() są uruchamiane w zależności od aplikacji, w której działa aplet. Żadna aplikacja nie wywołuje tych metod z watku AWT.

Wiele przeglądarek tworzy nowy watek dla każdego apletu znajdującego się na stronie, który jest używany do wywołania metod stanów apletu. Niektóre przeglądarki tworzą grupę dla każdego uruchamianego apletu, dzięki temu można łatwo zakończyć watki należące do apletu. Każdy watek tworzony w którejkolwiek metodzie stanu należy do tej samej grupy watków. Tworzenie watków jest potrzebne przy wykonywaniu czasochłonnych operacji. Watki tworzy się używając klasy Thread.

Aplety mogą działać w architekturze klient-serwer. W tym celu należy uruchomić serwer na komputerze, z którego aplet pochodzi. Trzeba także określić numer portu, na którym nasłuchuje. Potem aplet może się już komunikować z serwerem za pomocą protokołu HTTP. Klient porozumiewa się z serwerem za pomocą gniazd. Pakiety do przesłania tworzy się przy pomocy klasy DatagramSocket. Klasa InetAddress służy do przechowywania adresu internetowego.

Ze względów bezpieczeństwa przeglądarki implementują prawa dostępu. Aplety nie mogą ładować bibliotek i wykonywać metod. Nie mogą też czytać z pliku i zapisywać do pliku na komputerze, na którym są wykonywane. Nie mogą się tworzyć połączeń z wyjątkiem hosta, z którego pochodzą. Nie mogą też wykonywać programów na komputerze, na którym są

wykonywane. Nie mogą czytać właściwości systemu. Ich okna wyglądają inaczej niż zwykle okno aplikacji. Z drugiej strony aplety mogą wyświetlać zawartość stron HTML. Mogą także wywoływać metody publiczne innych apletów znajdujących się na tej samej stronie. Aplety uruchomione z systemu plików nie mają żadnych z powyższych ograniczeń. Aplety przestają działać jak się opuści ich stronę.

PODSUMOWANIE

Przedstawione przeze mnie technologie mają swoje wady i zalety. Zalety języków skryptowych to prostota implementacji i przydatność. Natomiast wada są ograniczenia związane z implementowaniem złożonych programów, które wykorzystują zasoby komputera. Zaletą ActiveX jest ustandaryzowany dostęp do baz danych. Zasadniczo aplety Javy pozwalają na tworzenie złożonych programów nie naruszając przy tym bezpieczeństwa komputera. W zależności od tego, co będzie trzeba zrobić należy wybrać odpowiednią technologię, która będzie miał wystarczające możliwości do rozwiązania danego problemu, a nie będzie zawierał niepotrzebnych dodatków spowalniających jego wykonanie.

LITERATURA

Client-Side JavaScript Guide 1.3

<http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm>

<http://developer.netscape.com/docs/manuals/js/client/jsref/index.htm>

<http://developer.netscape.com/docs/javascript/e262-pdf.pdf>

<http://www.republika.pl/dszczepanik/javascript/indeks.html>

http://www.devguru.com/Technologies/vbscript/quickref/vbscript_intro.html

<http://www.oreilly.com/catalog/vbscript/excerpt/ch01.html>

http://www.intranetjournal.com/corner/wrox/progref/vbt/ch21_17.shtml

<http://www.byte.com/art/9709/sec5/sec5.htm>

<http://www.takempis.com/adointro.asp>

<http://tech.irt.org/articles/js178/>

http://www.devguru.com/Technologies/ado/quickref/ado_intro.html

Poznaj język Java 1.2 – Michael Morgan, Warszawa, styczeń 2001

<http://java.sun.com/docs/books/tutorial/applet/TOC.html#appletonly>

<http://java.sun.com/docs/books/tutorial/>

<http://java.sun.com/docs/>