

AS Basics }

AS Basics

- { Anatomy of ActionScript
- { Vocabulary
- { Read the ActionScript Diagram



Anatomy of ActionScript

Anatomy

- { code blocks }
- { dot syntax . }
- { semi colons ; }
- { parenthesis () }

Code Blocks

```
function onClick() {  
    mc.gotoAndPlay("in");  
    navLoc= "main";  
}
```

Code Blocks

```
function onClick() {  
    mc.gotoAndPlay("in");  
    navLoc= "main";  
}
```

Code Blocks

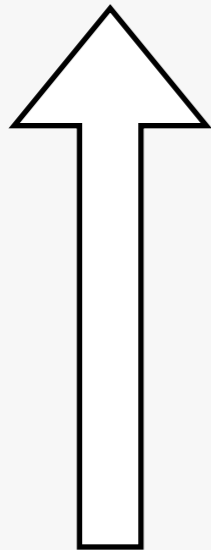
```
function onClick() {  
    mc.gotoAndPlay("in");  
    navLoc= "main";  
}
```

Code Blocks

```
function onClick() {  
    mc.gotoAndPlay("in");  
    navLoc= "main";  
}
```

Dot Syntax

```
mc.gotoAndPlay("in");
```

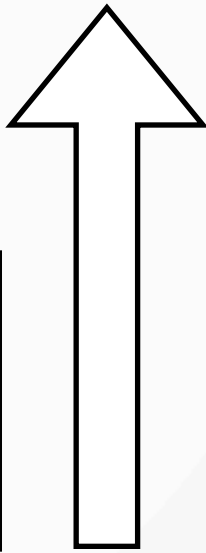


separates an object
and a method
or used as targeting

Semi Colons

```
mc.gotoAndPlay("in");
```

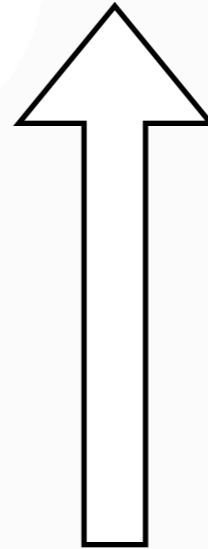
End of a line
of code



Parenthesis

```
mc.gotoAndPlay("in");
```

An Argument
that specifies
what to do





Vocabulary

Vocabulary

- { Variables
- { Data Typing
- { Tracing
- { Functions and returning values
- { Scope
- { Event Handlers
- { Objects and Classes
- { Methods
- { Properties
- { Comments
- { Conditionals

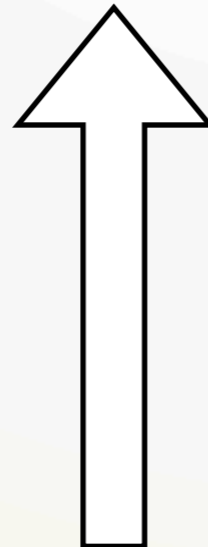


Variables

Variables }

Storage container

```
var myName:String = "Josh";
```



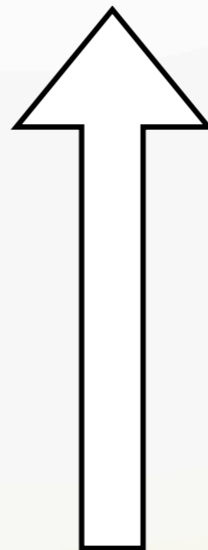
DataTyping



Data Typing}

Associates a variable with a type of value

```
var myName:String = "Josh";
```





Tracing

Tracing

```
trace("hello") //outputs the string hello  
var name:String = "Josh";  
trace(name) //outputs the string Josh
```

Tracing

```
trace("hello") //outputs the string hello  
var name:String = "Josh";  
trace(name) //outputs the string Josh
```

**Outputs to a
window only
the developer
can see.**

Functions



Functions }

Groups of reusable code

Declares the Function

```
function onClick(){  
    play();  
}
```

Calls the Function

```
onClick();
```

Functions }

Groups of reusable code

Declares the Function

```
function onClick(){  
    play();  
}
```

Calls the Function

```
onClick();
```

Functions }

Groups of reusable code

Declares the Function

```
function onClick(){  
    play();  
}
```

Calls the Function

```
onClick();
```

Returning Values

Returning Values }

Receipt for your function

Declares the Function

```
function onClick():Number{  
    return 2 + 2;  
}
```

Calls the Function

```
trace(onClick());  
  
//returns 4
```

Arguments



Arguments

```
mc.gotoAndPlay("in");
```

Arguments

```
mc.gotoAndPlay("in");
```

**Declares option
for the function.**



Scope

Scope

*Accessibility of functions, values,
and objects through code*

Scope}

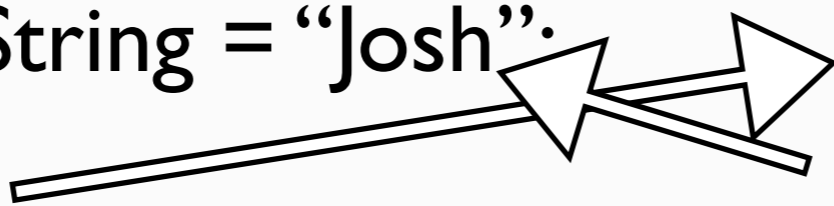
```
function getMyName(){  
    var myName:String = "Josh";  
    trace(myAge);  
}
```

```
function getMyAge(){  
    var myAge:Number = 29  
    trace(myName);  
}
```

Scope}

```
function getMyName(){  
  var myName:String = "Josh".  
  trace(myAge);  
}
```

```
function getMyAge(){  
  var myAge:Number = 29  
  trace(myName);  
}
```



Scope

myName Scope myAge Scope

```
function getMyName(){  
  var myName:String = "Josh".  
  trace(myAge);  
}
```

```
function getMyAge(){  
  var myAge:Number = 29  
  trace(myName);  
}
```

Neither value is is accessible

Scope

```
var myName:String = "Josh";
```

```
var myAge:Number = 29
```

```
function getMyName(){  
    trace(myAge);  
}
```

```
function getMyAge(){  
    trace(myName);  
}
```

Now the Values Are OK

Scope}

```
var myName:String = "Josh";  
function getMyName(){  
    trace(myAge);  
}
```

```
var myAge:Number = 29  
function getMyAge(){  
    trace(myName);  
}
```

The New Scope

Event Handlers



Event Handlers

```
mc.addEventListener(MouseEvent.CLICK, onClick);
```

This Registers all CLICKs on mc to fire off the onClick function



Objects and Classes

Objects and Classes

- { Objects are **instances** manipulated Programmatically
- { **Classes** are **blueprints** / templates that can be reused over and over with minor adjustments



Methods

Methods

- { A **function** or command of an object
- { play() is a method of MovieClip



Properties

Properties

- { An attribute that can be read and/or set.
For example height, width, or alpha of an object.

Comments



Comments

```
// used for making notes in the code  
/* They can be life savors  
especially in larger projects  
or when working with  
multiple developers */
```

Conditionals



Conditionals

- { if
- { else
- { else if
- { switch



Read the AS Diagram

Reading the AS Diagram

Stage

align: String

focus: InteractiveObject

frameRate: Number

quality: String

scaleMode: String

showDefaultContextMenu: Boolean

stageFocusRect: Boolean

stageHeight: int

stageWidth: int

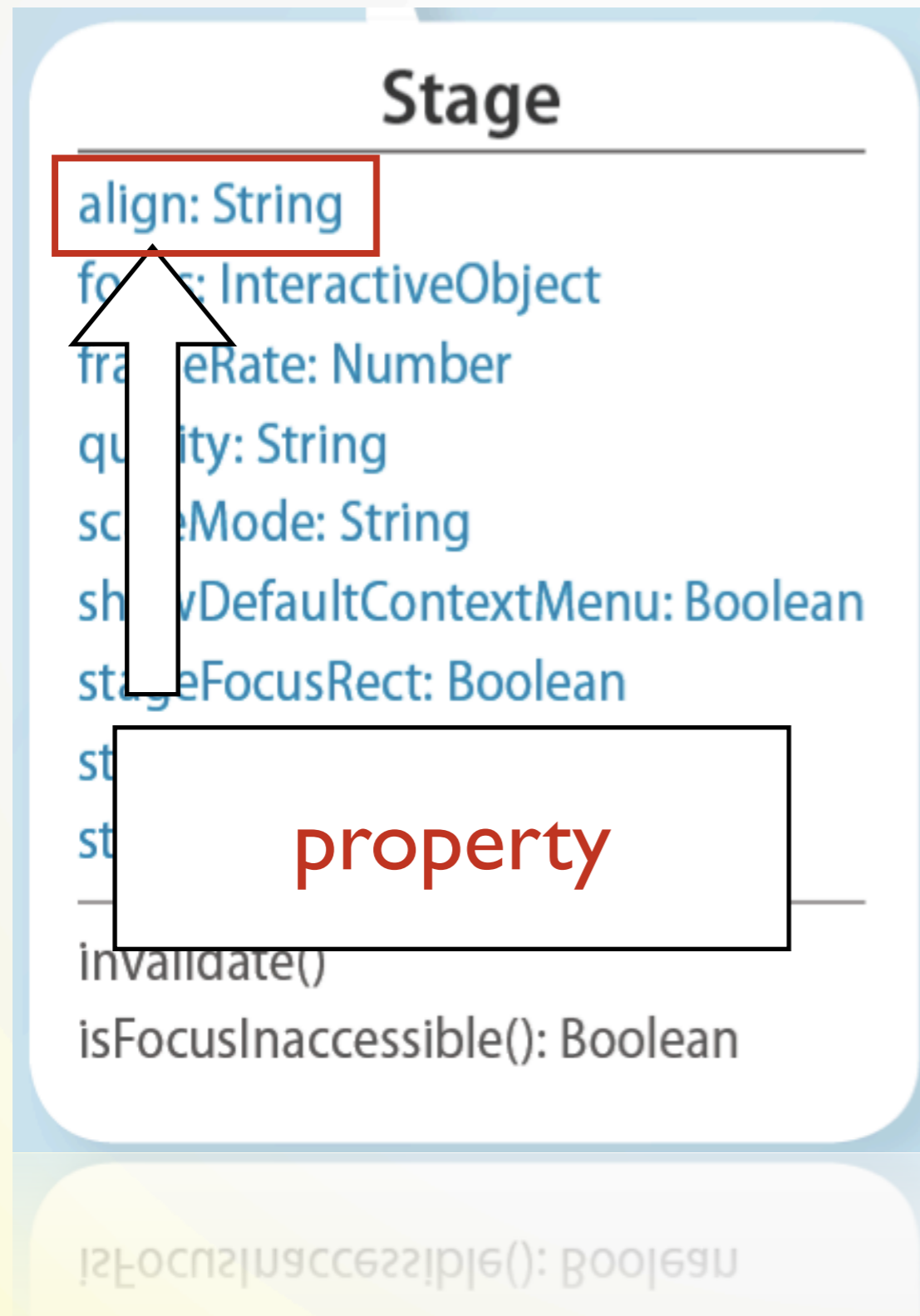
invalidate()

isFocusInaccessible(): Boolean

isFocusInaccessible(): Boolean

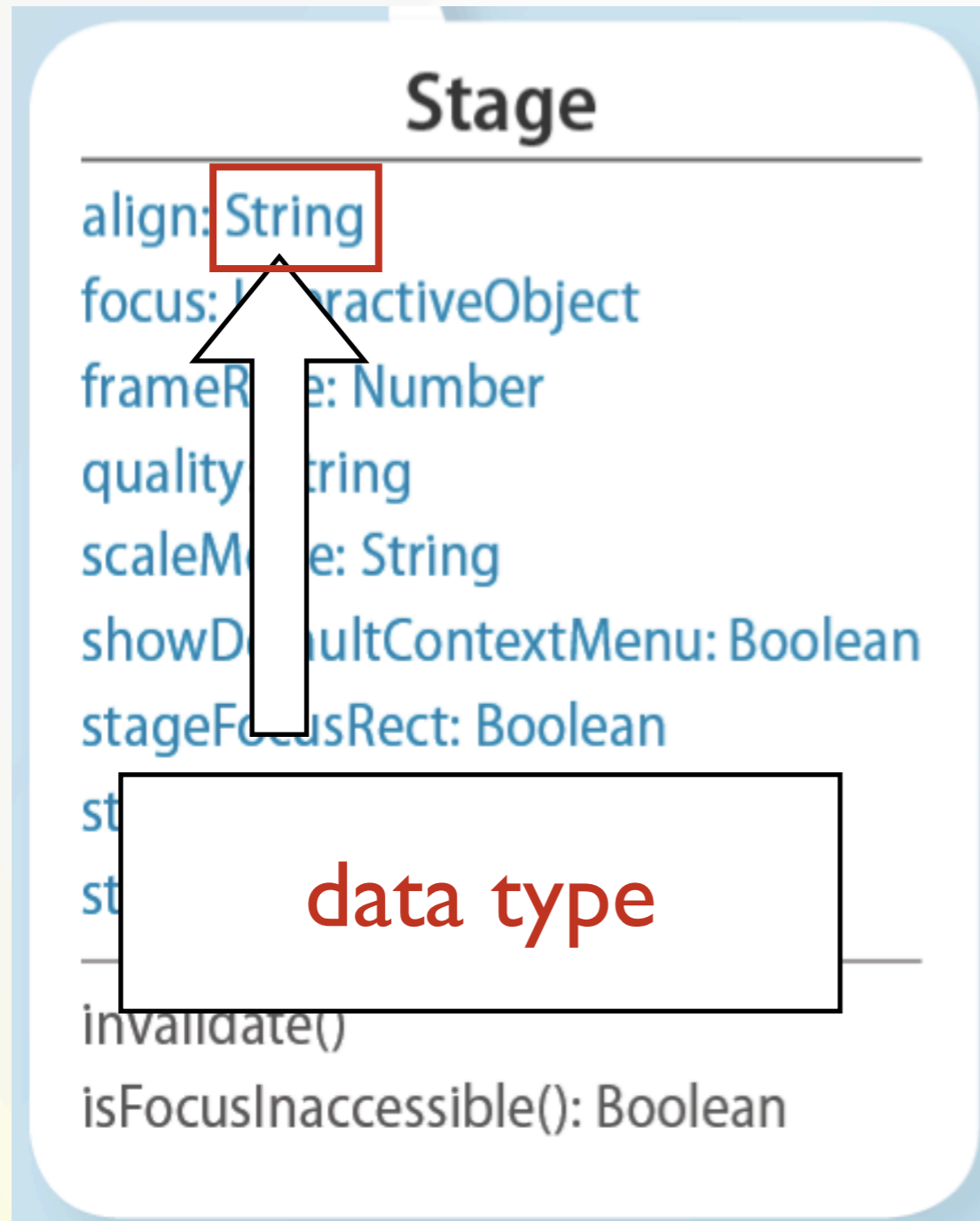
invalidate()

Reading the AS Diagram



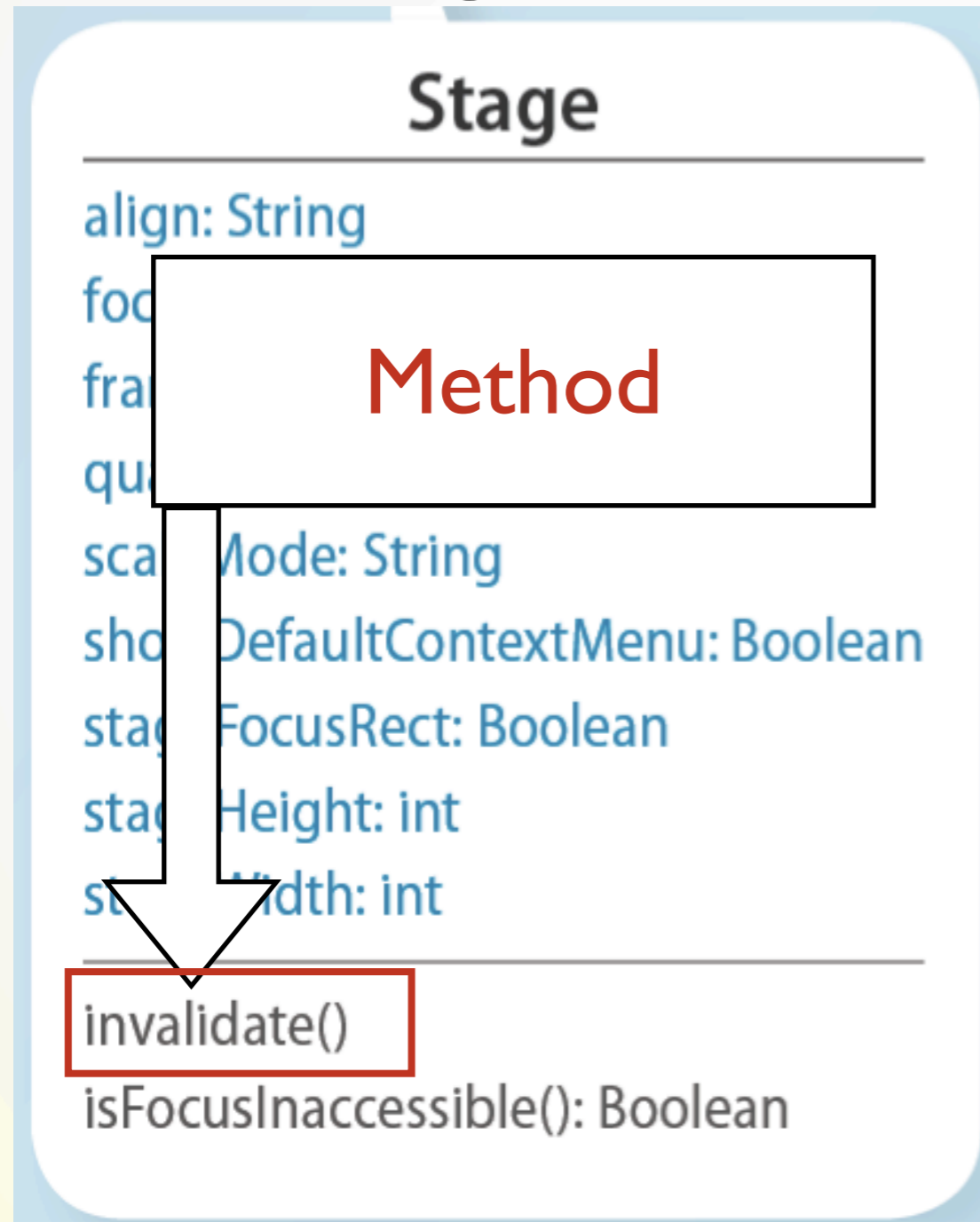
Note the colon.
That's an indication
of a property

Reading the AS Diagram



The data type describes the value required by the property

Reading the AS Diagram



Note the parenthesis. That's an indication of a method

Pop Quiz



Name a Method

Name a Method

Stage

align: String

focus: InteractiveObject

frameRate: Number

quality: String

scaleMode: String

showDefaultContextMenu: Boolean

stageFocusRect: Boolean

stageHeight: int

stageWidth: int

invalidate()

isFocusInaccessible(): Boolean

isFocusInaccessible(): Boolean

invalidate()

Name a Property

Name a Property

Stage

align: String

focus: InteractiveObject

frameRate: Number

quality: String

scaleMode: String

showDefaultContextMenu: Boolean

stageFocusRect: Boolean

stageHeight: int

stageWidth: int

invalidate()

isFocusInaccessible(): Boolean

isFocusInaccessible(): Boolean

invalidate()

Anatomy of ActionScript

- { Anatomy of ActionScript
- { Vocabulary
- { Read the ActionScript Diagram