GDP Documentation Guide

Updated October 8, 2021

GDP Keywords 4
$this 4
Keys Object 4

GDP Functions 5
clone(parameter) 5
createTimer(parameter1, parameter2, parameter3) 5
getMouseX() 5
getMouseY() 5
getPointerPos() 6
isKeyPressed(parameter) 6
Math.round(parameter) 6
moveX(parameter1, parameter2) 6
moveY(parameter1, parameter2) 7
parseInt(parameter) 7
random(parameter1, parameter2) 7
spin(parameter1, parameter2) 7

Object Methods 8
object.animation(parameter) 8
object.cleanupTimers() 8
object.clone() 8
object.draggable() 8
object.draggable(parameter) 8
object.fill(parameter) 9
object.findDistance(parameter) 9
object.findRoles(parameter) 9
object.findName(parameter) 9
object.flipDirection() 10
object.frameIndex(parameter) 10
object.height() 10
object.incrementAnimation() 10
object.isTouching(parameter) 10
object.moveForwardByRotation(); 10
object.moveTo(parameter) object.moveTo(parameter1, parameter2) 11

object.moveToObject(parameter) 11
object.pointToCursor() 11
object.pointToObject(parameter) 11
object.remove() 11
object.rotation() 11
object.rotation(parameter) 12
object.scaleX() object.scaleY() 12
object.scaleX(parameter) object.scaleY(parameter) 12
object.speedX() object.speedY() 12
object.speedX(parameter) object.speedY(parameter) 12
object.text(parameter) 13
object.toggleDraggable() 13
object.toggleVisible() 13
object.visible(parameter) 13
object.width() 13
object.x() object.y() 13
object.x(parameter) object.y(parameter) 14

Other Computing Concepts 15
Arrays 15
Array Methods 16
Boolean Operators and Logic 17
Comparison Operators 18
Conditional Statements 19
For Loops 20
Functions 21
Hexadecimal Numbers and Colors 22
Mathematical Operators 23
Switch Statements 24

GDP Keywords

$this

The $this keyword refers to the current object that is running the code. This allows for referencing objects without using specific names. It is often used to help write events for the scene and cloned objects.

```
$this.speedY(100);
```

Keys Object

The GDP's Keys object is used with the isKeyPressed function to check to see if a specific key is being pressed by the user.

All of the letter keys can be checked by using the format Keys.a, Keys.b, and so on.

The number keys can be checked by using the format Keys.one, Keys.two, and so on.

Other keys you can check are:

Keys.upArrow	Keys.downArrow
Keys.leftArrow	Keys.rightArrow
Keys.enter	Keys.space

GDP Functions

clone(parameter)

parameter: GDP object

returns: GDP object

The clone function clones the specified object and places it in the scene. The cloned objects' names are in the form of originalName_X where X starts at 1 and increases for every clone.

```
var newShuriken = clone(shuriken);
```

createTimer(parameter1, parameter2, parameter3)

parameter1: number

parameter2: function

parameter3: Boolean

The createTimer function will wait the amount of time specified in parameter1 in milliseconds and run the function provided in parameter2. If parameter3 is true, the function will repeat continually based on parameter1 until the cleanupTimers function is called. If parameter3 is false, the function will only run once after the delay.

```
createTimer(1500, spawnShuriken, true);
```

getMouseX()

returns: number

The getMouseX function returns the mouse pointer's current x position.

```
$this.x(getMouseX());
```

getMouseY()

returns: number

The getMouseY function returns the mouse pointer's current y position.

```
var y = getMouseY();
```

getPointerPos()

returns: { x, y } object

This function will return the x and y values of the mouse pointer. This is generally stored into a variable for later manipulation.

```
var pos = getPointerPos();
```

isKeyPressed(parameter)

parameter: GDP key object

returns: Boolean

The isKeyPressed function returns if the specified key is pressed on any given frame. Reference the GDP key object table in the GDP Keywords section to see available keys.

```
var isSpacePressed = isKeyPressed(Keys.space);
```

Math.round(parameter)

parameter: number

returns: number

Part of the Javascript Math library, the round function rounds the value passed in as the parameter.

```
var roundedPosX = Math.round(getPointerPos().x);
```

moveX(parameter1, parameter2)

parameter1: GDP object

parameter2: number (optional)

The move function moves the object specified in parameter 1 at the speed given in parameter 2 in the x direction. If parameter 2 is left blank, then the object moves at the x speed defined in the object's properties tab. This function should be used in the Update Every Frame event.

```
moveX($this, 200);
```

moveY(parameter1, parameter2)

parameter1: GDP object

parameter2: number (optional)

The move function moves the object specified in parameter 1 at the speed given in parameter 2 in the y direction. If parameter 2 is left blank, then the object moves at the y speed defined in the object's properties tab. This function should be used in the Update Every Frame event.

```
moveY(ninja);
```

parseInt(parameter)

parameter: string

returns: number

The parseInt function converts the string passed as a parameter and returns its value as a number.

```
var score = parseInt(scoreLabel.text());
```

random(parameter1, parameter2)

parameter1: number

parameter2: number (optional)

returns: number

The random function returns a random whole number between parameter1 and parameter2. The order of the parameters does not matter. If only one parameter is used, a random whole number between 0 and the parameter. The ranges are inclusive.

```
var diceRoll = random(1, 6);
```

spin(parameter1, parameter2)

parameter1: GDP object

parameter2: number

This function targets an object specified in parameter 1 and rotates it at the speed specified in parameter 2. Positive speed values rotate clockwise, and negative speed values rotate counter-clockwise. Adjusting the offset x and y values is key to making objects rotate from their center instead of from their top-left anchor, which is the default behavior of this function.

```
spin(ninja, 90);
```

Object Methods

object.animation(parameter)

parameter: string

The animation method changes the sprite object's active animation to the name specified in the parameter. The animation names and frames are defined in the object's Animations tab.

```
ninja.animation("run");
```

object.cleanupTimers()

The scene's cleanupTimers method will stop all running timers that were made with the createTimers function.

```
$this.scene.cleanupTimers();
```

object.clone()

returns: GDP object

The clone method clones the specified object and places it in the scene. The cloned objects' names are in the form of originalName_X where X starts at 1 and increases for every clone.

```
var newShuriken = shuriken.clone();
```

object.draggable()

returns: Boolean

When used without a parameter, the draggable method returns true if dragging the object is enabled or false if it is disabled. The draggable property can be set on the object's Properties tab by clicking the lock image.

```
var canMoveNinja = ninja.draggable();
```

object.draggable(parameter)

parameter: Boolean

The draggable method will set the object's draggable property to true or false based on the provided parameter.

```
$this.draggable(false);
```

object.fill(parameter)

parameter: hex color string

The fill method will change the color of certain GDP objects to the hexadecimal color string passed in as a parameter. See the Hexadecimal Color section for more information.

```
rectangle.fill("#FFA71A");
```

object.findDistance(parameter)

parameter: GDP object

returns: number

The findDistance method returns the distance in pixels between the original object and the object passed in as a parameter.

```
var dist = ninja.findDistance(shuriken);
```

object.findRoles(parameter)

parameter: string

The findRoles method will find all GDP objects with the role passed in as the parameter. Object roles can be set on the Properties tab. This method is often used on the scene.

```
$this.scene.findRoles("coin");
```

object.findName(parameter)

parameter: string

returns: GDP Object

The findName method searches the object's children to find a different object with a name matching the parameter. If an exact match is not found, code that tries to use the result will cause errors. This method is often used on the scene object to obtain a reference to an object.

```
var theNinja = $this.scene.findName("ninjaSprite");
```

object.flipDirection()

The flipDirection method changes an object's direction of moment by multiplying both its speedX and speedY values by -1.

```
shuriken.flipDirection();
```

object.frameIndex(parameter)

parameter: number

The frameIndex method will set a sprite's active frame to the number provided as a parameter.

```
ninja.frameIndex(3);
```

object.height()

returns: number

The height method returns the value of the object's height as specified in the properties tab.

```
var radius = $this.height() / 2;
```

object.incrementAnimation()

When used on a sprite with animation frames, the incrementAnimation method will advance the sprite's active frame by one. This method is often used in the Update Every Frame event to continually animate a sprite.

```
$this.incrementAnimation();
```

object.isTouching(parameter)

parameter: GDP object

returns: Boolean

The isTouching method compares the original object and the object passed in the parameter to see if they are overlapping or touching. The method will return true if any parts of the objects are touching and false if they are not.

```
shuriken.isTouching(ninja);
```

object.moveForwardByRotation();

This method uses the object's rotation value and moves it in that specific direction using its speedX and speedY values.

```
shuriken.moveForwardByRotation();
```

object.moveTo(parameter)
object.moveTo(parameter1, parameter2)

parameter: GDP object

parameter1, parameter2: number

If used with one parameter, the object will move to an object passed as the parameter. If used with two parameters, the object will move to the specified x and y coordinates.

```
enemy.moveTo(goal);
ninja.moveTo(50, 50);
```

object.moveToObject(parameter)

parameter: GDP object

The moveToObject method will make the original object move to the object passed in as a parameter.

```
shuriken.moveToObject(ninja);
```

object.pointToCursor()

The pointToCursor method will make the object rotate to face the mouse cursor.

```
ninja.pointToCursor();
```

object.pointToObject(parameter)

parameter: GDP object

The pointToObject method rotates the original object to face the center of the object passed in as a parameter.

```
arrow.pointToObject(goal);
```

object.remove()

The remove method completely removes the object from the scene.

```
shuriken.remove();
```

object.rotation()

returns: number

The rotation method when used without a parameter will return the value of the object's rotation in degrees from 0 to 359.

```
var rot = ninja.rotation();
```

object.rotation(parameter)

parameter: number

The rotation method rotates the object to the specified value in degrees.

```
shuriken.rotation(90);
```

object.scaleX()
object.scaleY()

returns: number

When used without a parameter, the scaleX and scaleY methods return the value of the object's x or y scale, respectively. The scale of the object determines how stretched or squished it is.

```
var currentScale = $this.scaleY();
```

object.scaleX(parameter)
object.scaleY(parameter)

parameter: number

When used with a parameter, the scaleX and scaleY methods set the value of the object's x or y scale, respectively. The scale of the object determines how stretched or squished it is.

```
ninja.scaleX(3);
ninja.scaleY(0.5);
```

object.speedX()
object.speedY()

returns: number

When called without parameters, the speedX and speedY methods return the value of the object's x or y speed, respectively.

```
var mySpeed = $this.speedY();
```

object.speedX(parameter)
object.speedY(parameter)

parameter: number

When called with a parameter, the speedX and speedY methods set the value of the object's x or y speed, respectively.

```
ninja.speedX(150);
```

object.text(parameter)

parameter: string or number

When used on a label object, the text method will set the label's text property and update it in the scene.

```
label.text("Hello world!");
```

object.toggleDraggable()

When used without a parameter, the toggleDraggable method changes the object's draggable property of the object from true to false or false to true. The draggable property can be set on the object's Properties tab by clicking the lock image.

```
tile.toggleDraggable();
```

object.toggleVisible()

When used without a parameter, the toggleVisible method changes the visibility of the object from true to false or false to true. The visible property can be set on the object's Properties tab by clicking on the eye image.

```
target.toggleVisible();
```

object.visible(parameter)

parameter: Boolean

The visible method will set the object's visible property based on the passed in Boolean. The visible property can be set on the object's Properties tab by clicking on the eye image.

```
ninja.visible(true);
```

object.width()

returns: number

The width method returns the value of the object's width as specified in the properties tab.

```
var hitboxSize = ninja.width();
```

object.x()
object.y()

returns: number

When called without parameters, the x and y methods return the value of the object's x or y coordinate, respectively.

```
var ninjaXPos = ninja.x();
```

object.x(parameter)
object.y(parameter)

parameter: number

When called with parameters, the x and y method will set the x and y positions of the object, respectively.

```
ninja.y(72);
```

Other Computing Concepts

Arrays

An array is a set of items inside square brackets and separated by commas. They are used to store data in a specific order. In some programming languages all the data in an array needs to be similar, but JavaScript lets you put any data in any array in any order!

Each item in an array has an index, or the number that represents its position in the array. All arrays start at index 0 and go up by one for each element.

```
var shoppingList = ["eggs", "orange juice", "granola", "bread"];
```

The element in the 0th index is "eggs".

The element in the 1st index is "orange juice".

The element in the 2nd index is "granola".

The element in the 3rd index is “bread".

You can get the value of an element in the array by using the arrayName[index] format.

```
var myFavorite = shoppingList[2]; // returns "granoloa"
```

You can also set the value of an array at an index.

```
shoppingList[1] = "milk"; // replaces "orange juice" with "milk"
```

You can get the number of elements inside an array by using the array’s length property. This is not a function or a method, so you do not need parentheses.

```
var myItems = shoppingList.length; // returns 4
```

Array Methods

To add an item to the end of an array, use the push method with the new item as a parameter.

```
var computerParts = ["CPU", "RAM"];
computerParts.push("mouse");
// the computerParts array is now ["CPU", "RAM", "mouse"]
```

To remove an item or items from anywhere in the array, use the slice method with two parameters. The first parameter is the index you want to remove, and the second parameter is how many elements you want to remove. To remove just a single element the second parameter should be 1.

```
var toDo = ["clean room", "make bed", "wash dishes", "make lunch"];toDo.slice(2, 1); // remove one
 element at index 2
// the toDo array is now ["clean room", "make bed", "make lunch"];
```

Boolean Operators and Logic

A Boolean variable has two possible values, true or false. Boolean logic uses Boolean operators to combine Boolean statements into one true or false value.

The AND operator is represented in JavaScript with &&. A statement using the AND operator will return true only if both the left and the right hand sides are true.

true	&&	true	Evaluates to	true
true	&&	false	Evaluates to	false
false	&&	true	Evaluates to	false
false	&&	false	Evaluates to	false

The OR operator is represented in JavaScript with ||. A statement using the OR operator will return true if at least one of the left or right hand sides is true.

true	||	true	Evaluates to	true
true	||	false	Evaluates to	true
false	||	true	Evaluates to	true
false	||	false	Evaluates to	false

The NOT operator is represented in JavaScript with !. When used before a Boolean variable it will return the opposite of the original value.

!	true	Evaluates to	false
!	false	Evaluates to	true

Comparison Operators

Comparison operators are used to compare two values. All comparison operators will return either true or false.

a	<	b	Returns true when	a is less than but not equal to b
a	<=	b	Returns true when	a is less than or equal to b
a	>	b	Returns true when	a is greater than but not equal to b
a	>=	b	Returns true when	a is greater than or equal to b
a	===	b	Returns true when	a is exactly equal to b
a	!==	b	Returns true when	a is not exactly equal to b

Conditional Statements

A conditional statement lets you run code if a condition is met. A conditional statement begins with the if keyword followed by a set of parentheses. Inside the parentheses is a conditional statement that needs to evaluate to either true or false.

```
var spawnChance = random(0, 100);
if (spawnChance > 25) {
  shuriken.clone();
}
```

An if statement will only run specific code if the condition evaluates to true. You can use an else statement to run code when the condition evaluates to false.

```
if (ship.speedY() !== 0) {
  ship.animation("thrust");
} else {
  ship.animation("idle");
}
```

Sometimes you want to check two different conditions. You can combine multiple if statements by using an else if statement.

```
if (isKeyPressed(Keys.leftArrow)) {
  ninja.speedX(-100);
} else if (isKeyPressed(Keys.rightArrow)) {
  ninja.speedX(-100);
} else {
  ninja.speedX(0);
}
```

For Loops

All for loops tart with the for keyword followed by a set of parentheses. Inside the parentheses are the loop's counter variable, run condition, and increment instruction. For loops are used to repeat code a specific number of times or to perform actions on every item in an array.

Before running the code, the for loop checks the run condition. If it evaluates to true, the code inside will run. If it evaluates to false, the loop will stop repeating.

```
var hiddenTreasures = [chestSprite, gemSprite, ringSprite];

for (var t = 0; t < hiddenTreasures.length; t++) {
  var treasure = hiddenTreasures[t];
  treasure.visible(false);
}
```

Functions

A function is code that is given a name and can be reused. A function is declared by using the function keyword followed by the function's name. After the name, there open and closed parentheses with any parameters inside. The code of the function is placed inside a set of curly brackets. You can use the return keyword to provide a value when the function is called.

```
function double(x) {
  return x * 2;
}
```

```
function sayHello(person) {
  var greeting = "Hi, " + person + "!";
  return greeting;
}
```

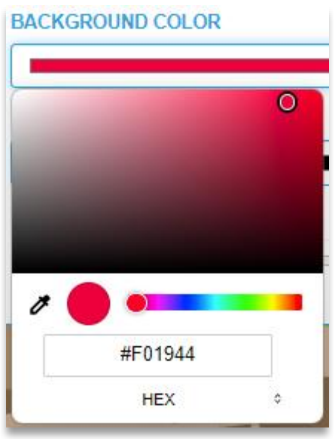
Hexadecimal Numbers and Colors

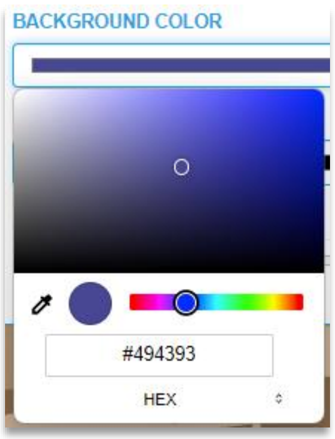
Our normal number system uses 0 through 9, but a hexadecimal system also uses letters A through F. You don't need to know how to convert between decimal and hexadecimal, but you should be familiar with what a hexadecimal number is and what it looks like.

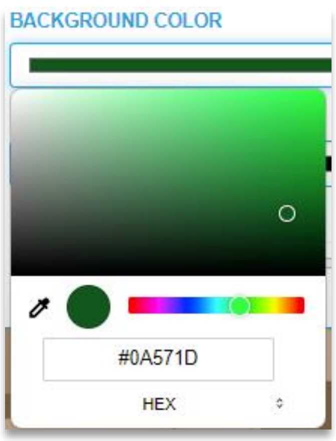
Decimal	is equal to	Hexadecimal	Decimal	is equal to	Hexadecimal
00	=	00	11	=	0B
01	=	01	12	=	0C
02	=	02	13	=	0D
03	=	03	14	=	0E
04	=	04	15	=	0F
05	=	05	16	=	10
06	=	06	17	=	11
07	=	07	18	=	12
08	=	08	19	=	13
09	=	09	20	=	14
10	=	0A	21	=	15
30	=	1E	70	=	46
40	=	28	80	=	50
50	=	32	90	=	5A
60	=	3C	100	=	64

Computers create colors by combining red, blue, and green to create over 16 million unique colors. Websites and many programming languages represent colors by a 6 digit hexadecimal string starting with a #. The first two hexadecimal numbers represent the amount of red in the final color, the second pair represents the amount of green, and the final two numbers represent the amount of blue.

The values of red, green, and blue in a hexadecimal color can each be anywhere between 00 and FF (converted to decimal, this is the same as 0 to 255). Some GDP objects have a color picker in the Properties tab.







Mathematical Operators

A mathematical operator performs a specific operation on one or more values. Computer science uses the same addition, subtraction, multiplication, and division as regular math, but it also includes a few new operators like increment and decrement.

a	+	b	Add a and b
a	-	b	Subtract a and b
a	*	b	Multiply a and b
a	/	b	Divide a by b
a	++		Increase a by 1
a	--		Decrease a by 1
a	+=	b	Increase a by b
a	-=	b	Decrease a by b

Switch Statements

A switch statement looks at a single variable and runs code based on its value. A switch statement starts with the switch keyword followed by the variable you want to check in parentheses.

Inside the body of the switch statement is a series of case statements. Each case statement starts with the case keyword followed by the value you are checking and a colon. The body of a case statement is the code you want to run with a break keyword at the very end. The break keyword tells the switch statement that the case is over.

After all the case statements, you can use the default keyword followed by a colon and the code you want to run if no cases match.

```
switch(powerup) {
  case "speedUp":
    ninja.speedX(150);
    break;
  case "slowDown":
    ninja.speedX(50);
    break;
  default:
    ninja.speedX(100);
    break;
}
```