



# Multimedia Programming 210

## Week 2

# Homework

SLACK

# Review

- `<html>...</html>` .....Start and end HTML
- `<head>...</head>` .....Head of page, not actual content
- `<title>...</title>` .....Title of page
- `<body>...</body>` .....Body of page, where the content goes
- `<div>...</div>` .....Content section - [Block level](#)
- `<span>...</span>` .....Grouping - [Inline](#)
- `<p>...</p>` .....Paragraph
- `<b>...</b>` .....Bold
- `<br />` .....Line break (you'll notice that this tag doesn't have any content and therefore is both an begin and end tag, with the slash)
- `<H1></H1>` ..... (Also H2, H3, H4, etc..)
- `<!-- ... -->` .....Comments
- `<blink>...</blink>` .....Make your text blink
- `<a href="http://...">...</a>` Link to another page. The "href=""" portion is an attribute. Many tags have optional attributes.

# READING & WRITING JAVASCRIPT

# Hours Example

# STATEMENTS

The green code  
are statements

The pink { }  
mark the beginning and end  
of a code block

The purple code  
determines which code should run

```
var today = new Date();  
var hourNow = today.getHours();  
var greeting;  
  
/*Check what time it is and pick a greeting*/  
if (hourNow > 18) {  
    greeting = 'Good Evening!';  
} else if (hourNow > 12) {  
    greeting = 'Good Afternoon!';  
} else if (hourNow > 0) {  
    greeting = 'Good Morning!';  
} else {  
    greeting = 'Welcome';  
}  
document.write ('<h3>' + greeting + '</h3>');
```

*!!!! JavaScript is case sensitive (hourNow is not the same as HourNow)!!!!*

## VARIABLES

A script will have to temporarily store the bits of information it needs to do its job. It uses variables to store this data.

Once you leave the page, the browser will forget any information stored in the variables

A variable can change each time a script is run.

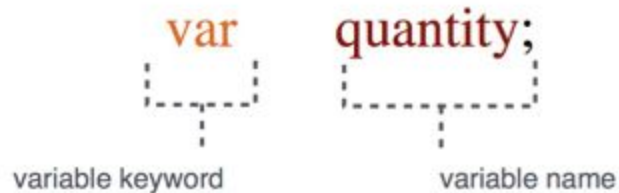
```
var today = new Date();  
var hourNow = today.getHours();  
var greeting;
```



## DECLARING A VARIABLE & ASSIGNING A VALUE

A new variable is **declared** with the **keyword “var”** and is assigned a **name**.

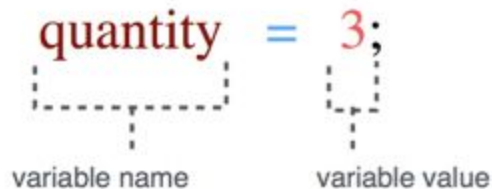
`var quantity;`



The diagram shows the code `var quantity;` with annotations. The word `var` is highlighted in orange, and `quantity` is highlighted in red. Below `var` is a dashed box with a vertical line pointing to it, labeled "variable keyword". Below `quantity` is a dashed box with a vertical line pointing to it, labeled "variable name".

Once you've declared the variable, you can **assign** it a **value** to store.

`quantity = 3;`



The diagram shows the code `quantity = 3;` with annotations. The word `quantity` is highlighted in red, the equals sign `=` is highlighted in blue, and the number `3` is highlighted in red. Below `quantity` is a dashed box with a vertical line pointing to it, labeled "variable name". Below `3` is a dashed box with a vertical line pointing to it, labeled "variable value".

## NAMING VARIABLES

- No spaces
- Letters, numbers, underscores and dollar signs
- Can't start with a number
- Use camel case

# DATA TYPES

A variable can store a:

NUMBER:

19

STRING:

"Good Evening"

BOOLEAN:

true

# NUMBERS

The numeric data type handles **numbers**.

```
var hourNow;  
hourNow = 19;
```

For larger numbers, there are **no commas**. There can also be **negative numbers** and **decimals**.

15734

- 23678

0.75

# STRINGS

Strings consist of **letters** and **other characters**.

The string data type is enclosed between a **pair of quotes** .  
(single or double)

```
var greeting;  
greeting = 'Good Evening!';
```

Strings must always be written on **one line**.

```
greeting = 'Good  
Evening!';
```



# BOOLEANS

When a value can be either **true** or **false**, the data type is known as a **boolean**.

Booleans are often used in **conditional** testing.

```
var available;  
available = true;
```

## DECLARING VARIABLES - SHORTHANDS

You can **declare** a variable and **assign** it a value in the same statement.

```
var cats = 9;  
var dogs = 5;
```

You can declare **more than one variable** in the same statement. And assign each one a value on a separate line.

```
var cats, dogs;  
cats = 9;  
dogs = 5;
```

You can declare **several variables** and assign them a **value** on the same line.

```
var cats = 9, var dogs = 5;
```

# ARRAYS

An array is a particular **type of variable**. It allows us to store a **series** of related values.

You do **not** need to **specify how many** values the array will hold upon creating it.

The values in an array can include **different data types**.



## CREATING AN ARRAY

A new variable is declared with the keyword “**var**” and is assigned a **name** – just like any variable.

The values are assigned to the array inside **square brackets**. Each value is separated by a **comma**.

```
var students;  
students = ['Will', 'Sophie', 'Asia'];
```

## ACCESSING ARRAY VALUES

Values in an array are accessed as if they are in a numbered list.  
Each number in an array is given an index number.

!!!! Index values start at 0 (not 1) !!!!

To retrieve a particular item within the array, the **array name** is called, along with the desired **index number** in **square brackets**.

```
var students;  
students = ['Will', 'Sophie', 'Asia'];
```

```
var best;  
best = students [0];
```

## EXPRESSIONS

An expression is any part of a statement that **resolves** to a **value**.

There are **two types** of expression:

1. Those that assign a value to a variable:

```
var cats = 9;
```

2. Those that use **multiple values** to return a **single** value:

```
var greeting = 'Hi' + 'Sophie!';
```

## ARITHMETIC OPERATORS

NAME	OPERATOR	EXAMPLE	RESULT
Addition	+	4 + 3	7
Subtraction	-	4 - 3	1
Multiplication		4 * 3	12
Division	/	12 / 4	3
Increment	++	i = 4; i ++;	5
Decrement	--	i = 4; i --;	3
Modulus	%	10 % 3	1

# Concatenation

- `var h = "hello";`
- `var w = "world";`
- `X+W;`
- `"hello" + "world";`
- `"hello " + "world";`

# Increment, decrement

- `var x = 0;`
- `x++;`
- `var y = ++x;`
- `var z = x++;`
- `x--;`
- `x-= 5;`

# Addition vs. concatenation

- `var x = 1;`
- `var y = "7";`
- `x+y;`
- `typeof x + y;`
- `y + x;`
- `x+ +y;`
- `x + number(y);`

[www.jshint.com](http://www.jshint.com)



p5

Color, Stroke, Fill

By default, the `color()` function will take either 1 argument for grayscale or 3 arguments for rgb

```
var gray = color(100);
```

```
var pink = color(255,0,200);
```



We can also use conventional html color names and hex values.

```
var blue = color("blue");
```

```
var red = color(#ff0000);
```



The `fill()` function sets the color of the fill for all the drawable elements like shapes and lines.

```
fill(51);
```

```
rect(25,25,100,100);
```



Color values saved in variables can also be used as colors.

```
fill(pink);
```

```
rect(125,25,100,100);
```



The function `noFill()` turns the fill off

```
noFill();
```

```
rect(150,50,100,100);
```



The stroke() functions works in a similar way, but changes the stroke of the drawable element.

```
stroke(blue);
```

```
Triangle (50,50,50,100,100,50);
```



The `noStroke()` turn the stroke off (remember to turn the fill back on or you won't see anything!);

```
noStroke();
```

```
fill(red);
```

```
ellipse(200,200,60,60);
```



The `background()` function sets the color of the canvas element and can use grayscale, rgb or html, just like stroke and fill.

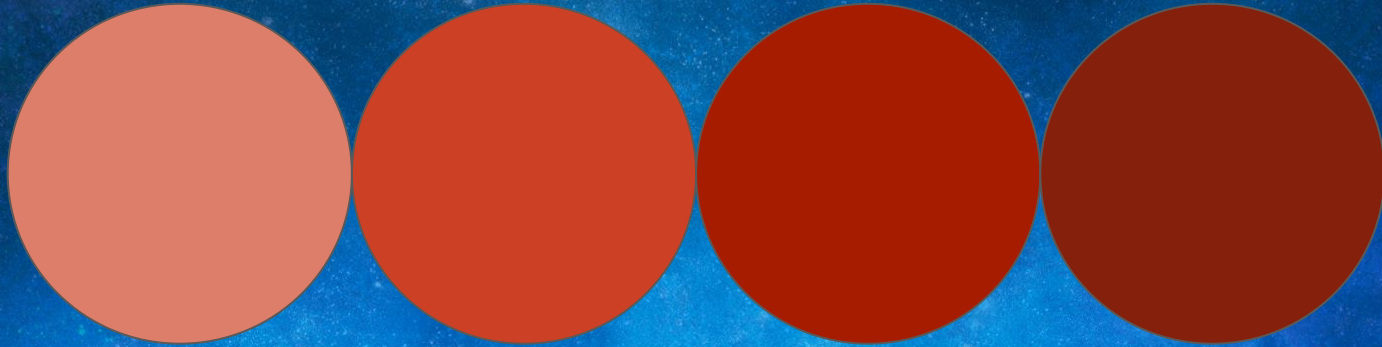
```
var green = color(0,190,120);
```

```
background(green);
```

Make sure to draw your background before drawing anything else.



Now lets try the statements and operators we learned to change a color over time.





# Homework

Using p5, create a design that uses variables as the arguments for the drawing functions.

- Drawing functions include fill(), stroke(), rect(), line(), ellipse(), triangle() etc...
  - `fill(0);` // NOT VALID
  - `var c = color(255,0,255); fill ( c );` // VALID

**There should be at least 5 colors and 5 shapes.**



# Homework

Using JavaScript, make an HTML page with

- at least 2 variables
- 1 arithmetic operator
- 1 string concatenation