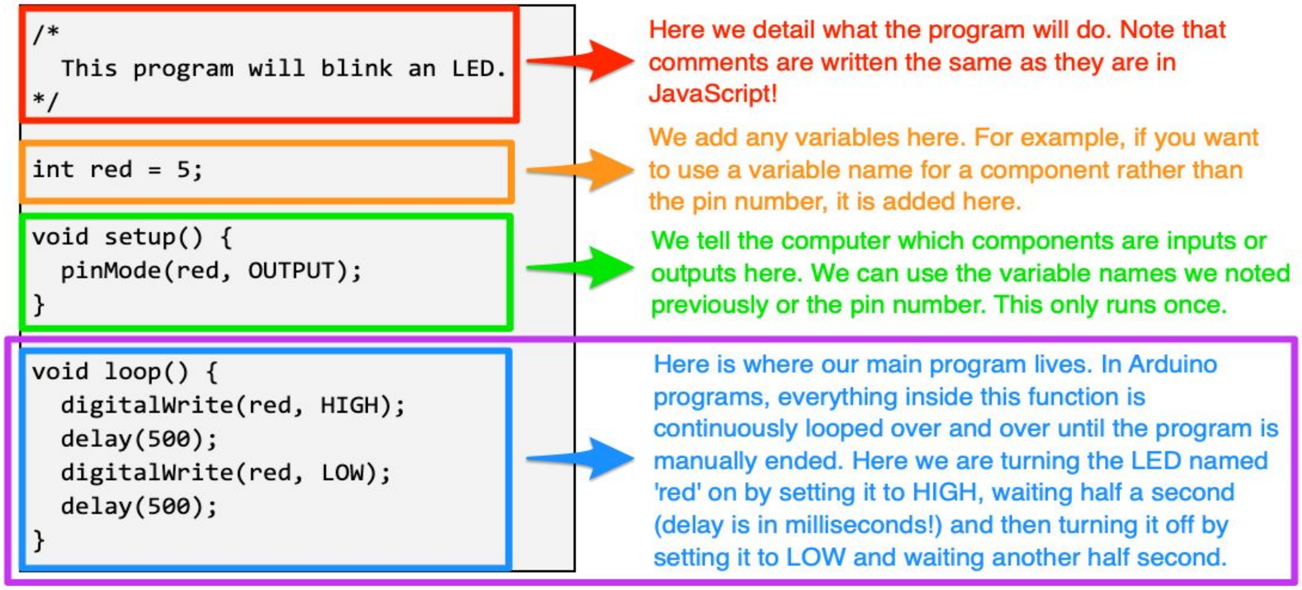
Arduino Reference Sheet

More info at: https://www.arduino.cc/reference/en/

Program Skeleton



```
/*
  This program will blink an LED.
*/

int red = 5;

void setup() {
  pinMode(red, OUTPUT);
}

void loop() {
  digitalWrite(red, HIGH);
  delay(500);
  digitalWrite(red, LOW);
  delay(500);
}
```

Here we detail what the program will do. Note that comments are written the same as they are in JavaScript!

We add any variables here. For example, if you want to use a variable name for a component rather than the pin number, it is added here.

We tell the computer which components are inputs or outputs here. We can use the variable names we noted previously or the pin number. This only runs once.

Here is where our main program lives. In Arduino programs, everything inside this function is continuously looped over and over until the program is manually ended. Here we are turning the LED named 'red' on by setting it to HIGH, waiting half a second (delay is in milliseconds!) and then turning it off by setting it to LOW and waiting another half second.

Basic Commands in Arduino IDE

`delay(`*`value`*`)`: waits a noted amount of time before moving on to following commands

- `value`: a number denoting time in milliseconds (1000 milliseconds = 1 second)

`digitalWrite(`*`variable, setting`*`)`: sets a component to on (`HIGH`) or off (`LOW`)

- `variable`: name of the component
- `setting`: Either `HIGH` or `LOW`

`digitalRead(`*`variable`*`)`: returns the value of a noted component, either `HIGH` or `LOW`

- `variable`: name of the component

`analogWrite(`*`variable, value`*`)`: sets a component to a certain value from `0-255`

- `variable`: name of the component
- `value`: A number `0-255`

NOTE: can only be used when a component is plugged into a pin with ~symbol

`analogRead(`*`variable`*`)`: returns the value of a noted component, a number `0-255`

- `variable`: name of the component

NOTE: can only be used when a component is plugged into a pin with ~symbol

Variables & Functions

`int `*`variable`*` = `*`value`*: creates an integer variable and assigns it a value

- `variable`: name of the variable written in camelCase
- `value`: A whole number (positive or negative)

`void `*`function`*`(){}`: defines a function as the commands found between the curly braces that **will not return a value**

- `function`: name of the function, written in camelCase
- To call the function, simply write the function name and a set of parentheses, ie. `myFunction();`
- Parameters can be included in the parentheses

`int `*`function`*`(){}`: defines a function as the commands found between the curly braces that **will return an integer**

- `function`: name of the function, written in camelCase
- To call the function, simply write the function name and a set of parentheses, ie. `myFunction();`
- Parameters can be included in the parentheses

Control Structures in Arduino IDE

If/Else Statements: Chooses actions to perform based on given conditions

```
if (condition) {
    commands;
}
else if (condition) {
    commands;
}
else {
    commands;
}
```

Example:

```
    if (digitalRead(button) == HIGH) {
        digitalWrite(motor, HIGH);
    }
    else {
        digitalWrite(motor, LOW);
    }
```

Sensor Conditions to use in if/else statements:

`digitalRead(`*`component`*`)`: returns the digital value of a component as `1` (on) or `0` (off)

- `component`: name of the component to be checked

`analogRead(`*`component`*`)`: returns the analog value of a component

- `component`: name of the component to be checked

For Loops: Repeats a set of actions a specific number of times

```
for (initialization; condition; increment) {
    commands;
}
```

Example:

```
    for (int i = 0; i < 4; i++) {
        analogWrite(redLED, count);
        count+=50;
    }
```

While Loops: Repeats a set of actions as long as a condition remains true

```
while (condition) {
    commands;
}
```

Example:

```
    while (digitalRead(button) == LOW) {
        digitalWrite(redLED, HIGH);
    }
    digitalWrite(redLED, LOW);
```

Arduino Operators

Mathematical Operators		*Comparison Operators*		*Logical Operators*	
+	addition	==	Equal to	**!**	Not
-	subtraction	!=	Not equal to	**&&**	And
*	multiplication	<	Less than	||	Or
/	division	<=	Less than or equal to		
%		>	Greater than		
		>=	Greater than or equal to		