Encoding Images

Digital Information

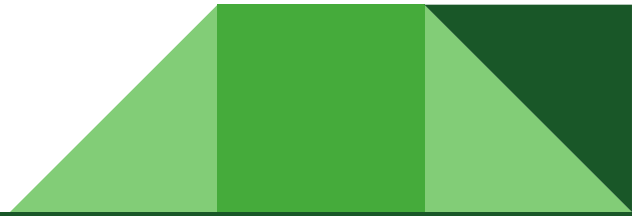
Last class, we discussed how we can encode **text** in Binary, which lets us store it in our computers!

Today, we're going to discuss how computers can store **images**!

Digital Information

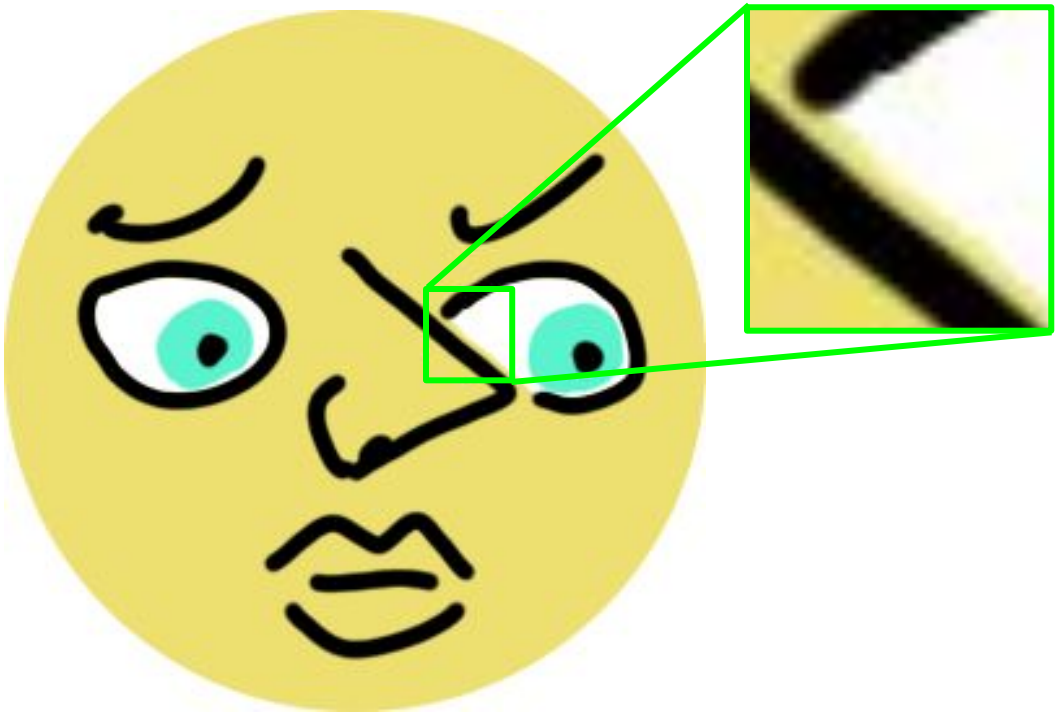
Just like with text, we need a way to break an image down into numbers, so that we can store them in Binary.

The best way to do this is using **Pixels**!

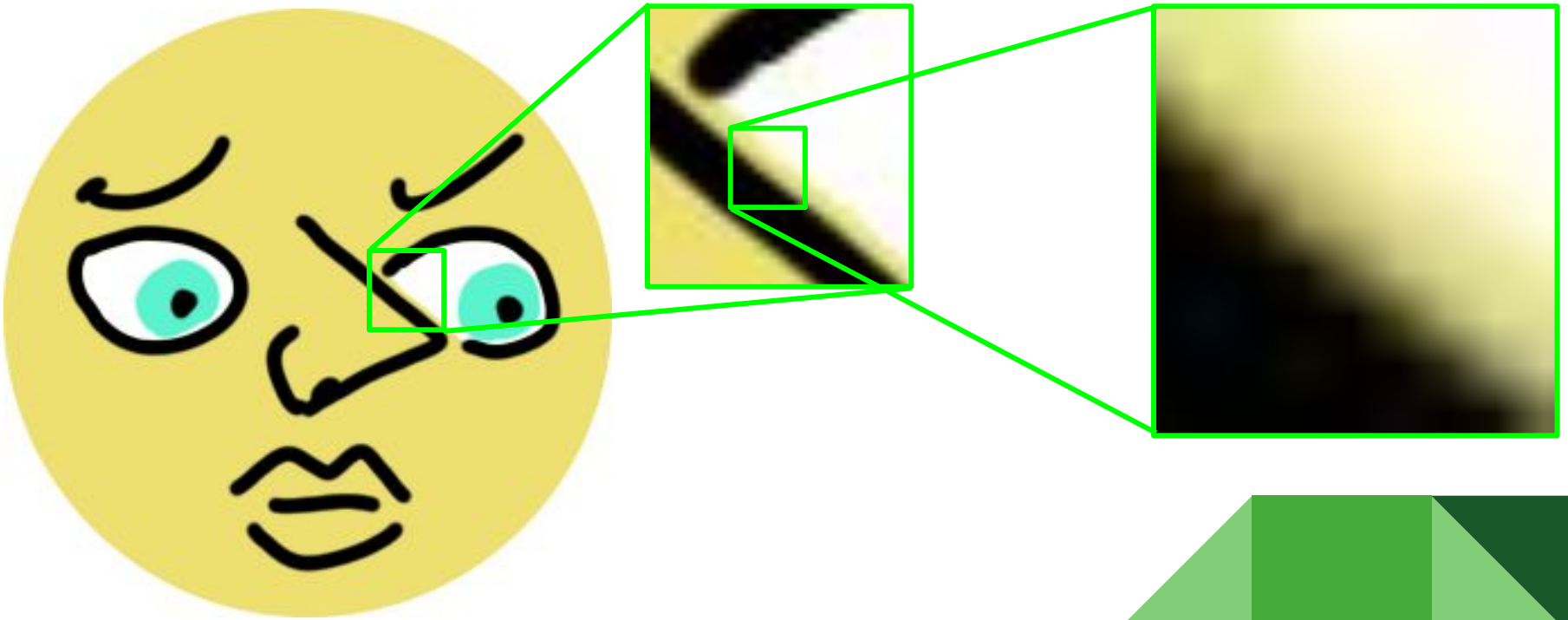


Images

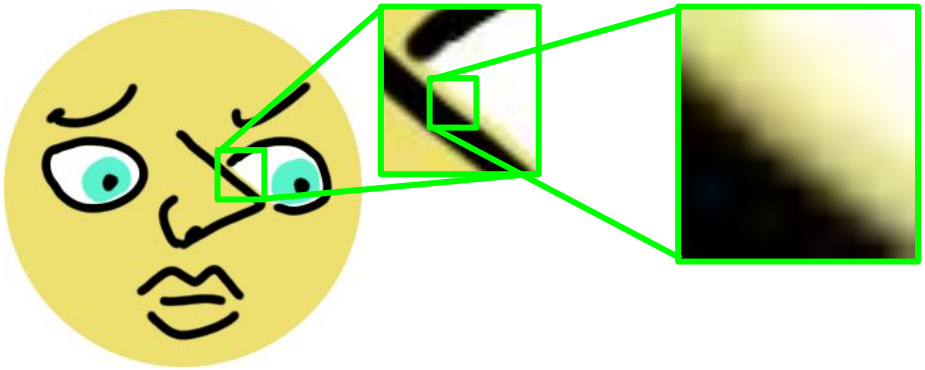
Images



Images



Images



All images are just a grid of different colored squares, called Pixels!

The position and color of each pixel can be encoded in a number, which lets us store all the information about an image using numbers!

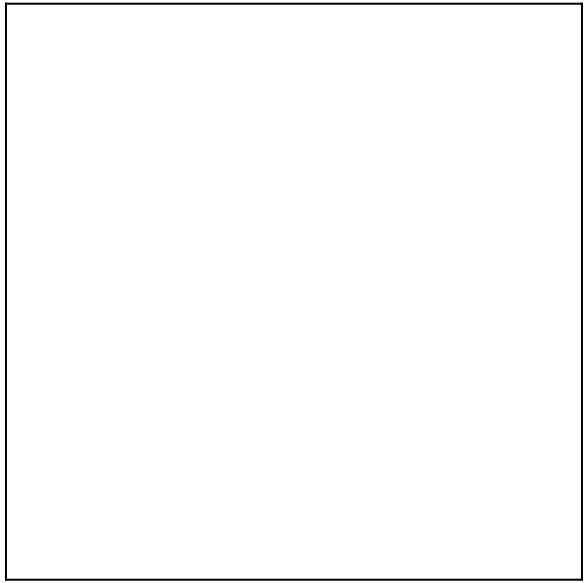
Simplest Pixel Encoding

The simplest possible way we can encode the information for a pixel is using only 0 and 1 - we'll talk about color later today.

If we encode 0 to mean **Black** and 1 to mean **White**, we can create a simple picture!

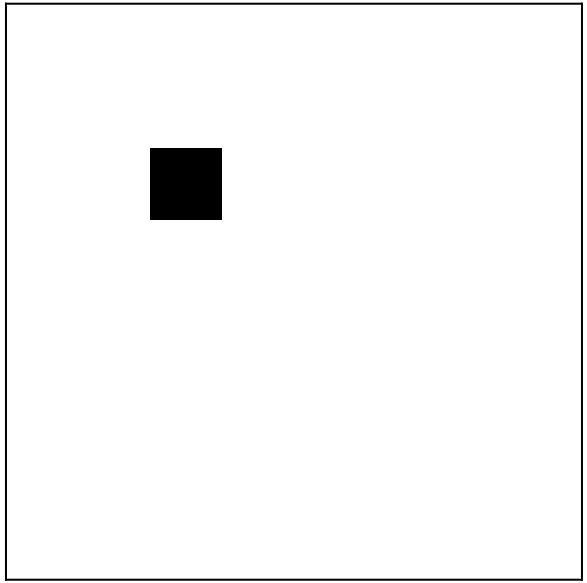
Simplest Pixel Encoding

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1



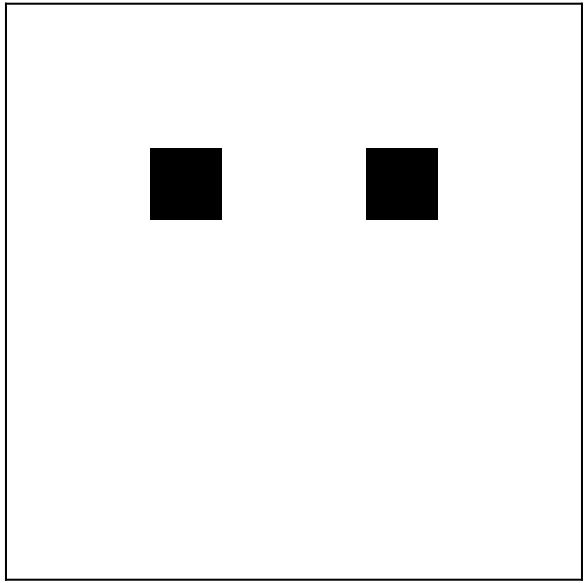
Simplest Pixel Encoding

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1



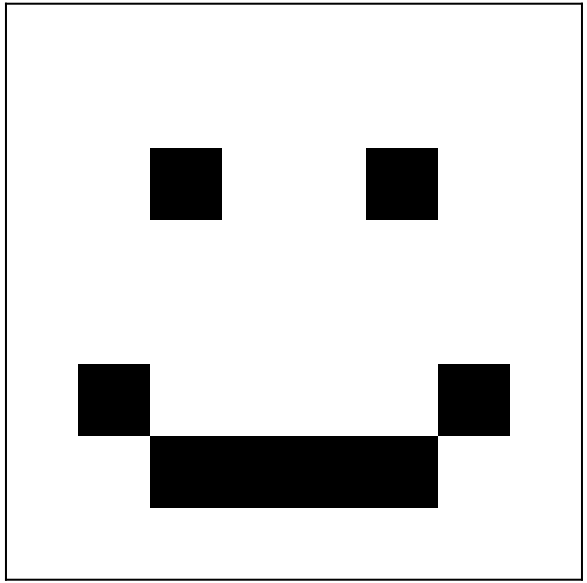
Simplest Pixel Encoding

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1



Simplest Pixel Encoding

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1
1	1	1	1	1	1	1	1



What the Hex?

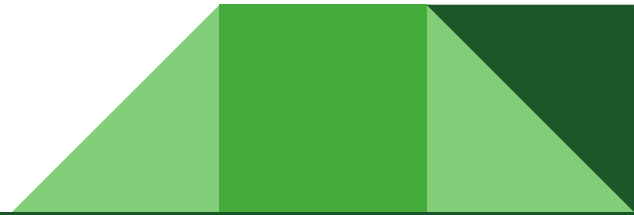
Back to Number Systems

Before we can talk about making our pixels colorful, we have to discuss a new number system - **Hexadecimal**, also known as **Hex**.

We learned about number systems a couple of classes ago, but let's refresh:

Decimal
Base: `10`
Digits: `0 1 2 3 4 5 6 7 8 9`

Binary
Base: `2`
Digits: `0 1`

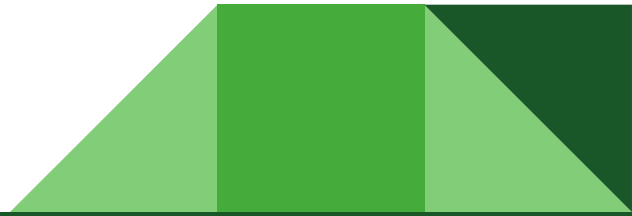


Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 … ???

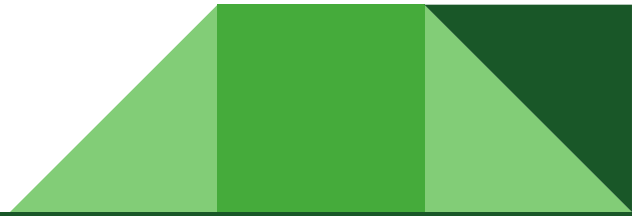


Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F



Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F

$A_{16} = 10_{10}$

$B_{16} = 11_{10}$

$C_{16} = 12_{10}$

$D_{16} = 13_{10}$

$E_{16} = 14_{10}$

$F_{16} = 15_{10}$

Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Example: $43F_{16}$

Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Example: $43F_{16}$

4	3	F

Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Example: $43F_{16}$

4	3	F
		1s
		16^0

Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Example: $43F_{16}$

4	3	F
	16s	1s
	16^1	16^0

Hexadecimal

Hexadecimal

Base: 16

Digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F

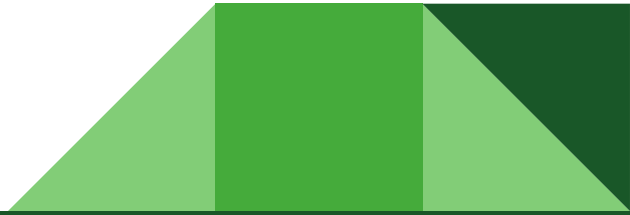
Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0



Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

$4 * 16^2$

Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

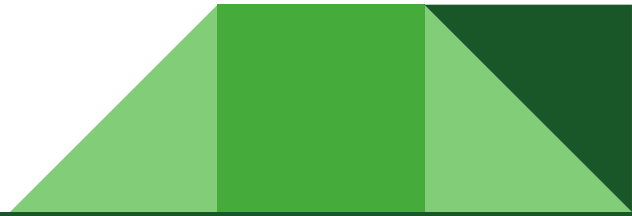
$4 * 16^2 + 3 * 16^1$

Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

$4 * 16^2 + 3 * 16^1 + F * 16^0$



Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

$4 * 16^2 + 3 * 16^1 + F * 16^0$

$4 * 256 + 3 * 16 + 15 * 1$

Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

$4 * 16^2 + 3 * 16^1 + F * 16^0$

$4 * 256 + 3 * 16 + 15 * 1$

$1024 + 48 + 15$

Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

$4 * 16^2 + 3 * 16^1 + F * 16^0$

$4 * 256 + 3 * 16 + 15 * 1$

1087

Hexadecimal

Example: $43F_{16}$

4	3	F
256s	16s	1s
16^2	16^1	16^0

$4 * 16^2 + 3 * 16^1 + F * 16^0$

$4 * 256 + 3 * 16 + 15 * 1$

$43F_{16} = 1087_{10}$

Hex to Binary

Each digit in Hexadecimal can be represented by 4 digits in Binary!

Hex to Binary

Each digit in Hexadecimal can be represented by 4 digits in Binary!

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Hex	Binary
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Hex to Binary

This relationship between Hex and Binary makes it easy for us to store larger numbers in fewer digits, while still being easy for the computer to read!

Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$

Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

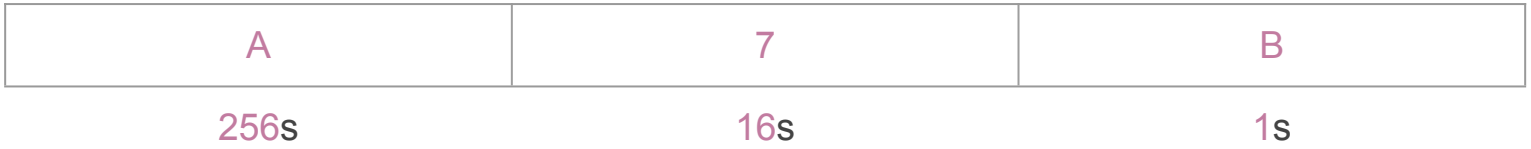
Ex: $A7B_{16}$

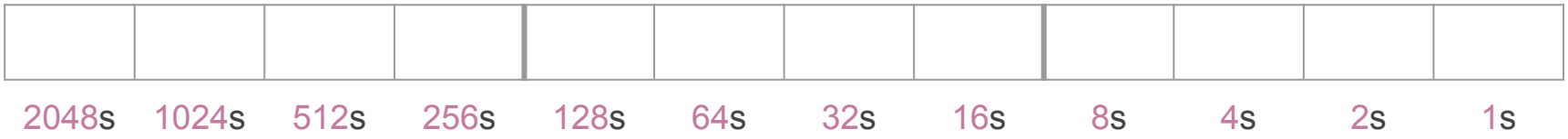
A	7	B
256s	16s	1s

Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$

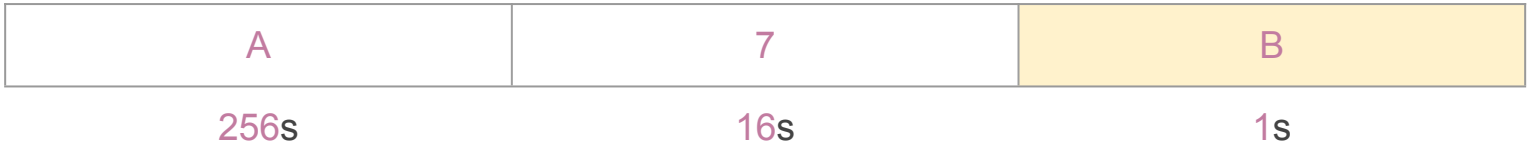


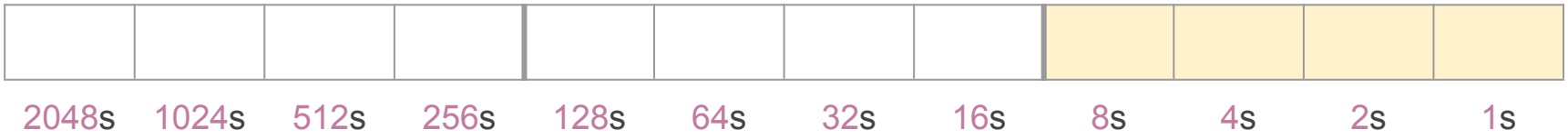


Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$

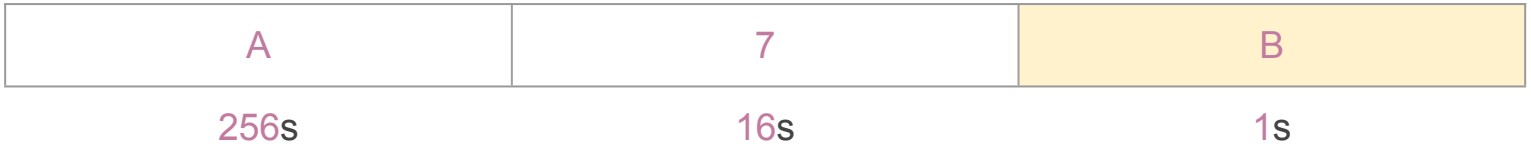


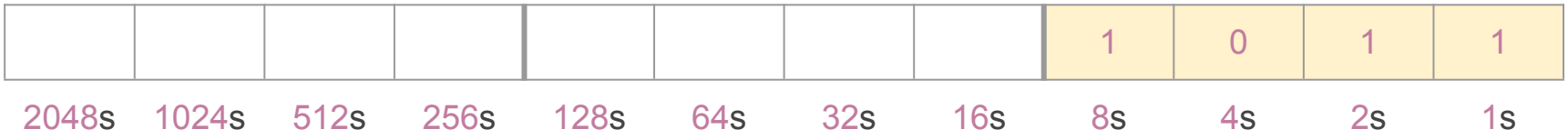


Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$

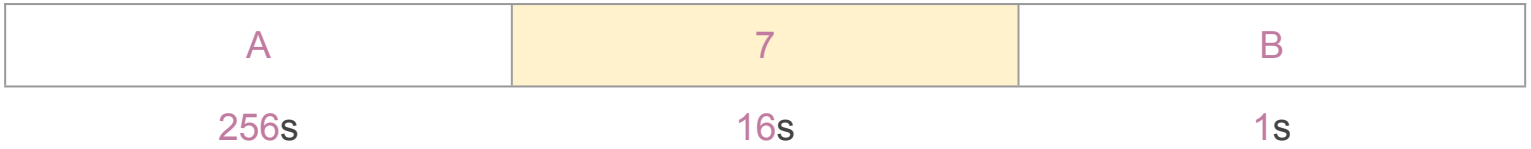


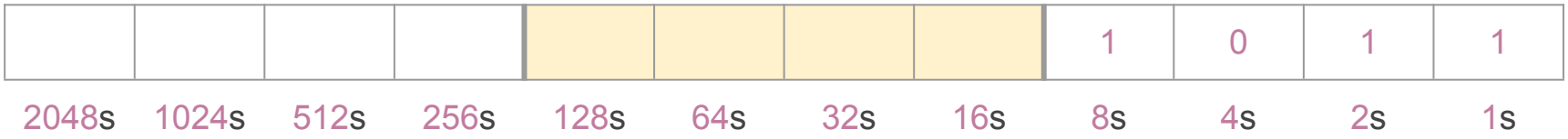


Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$

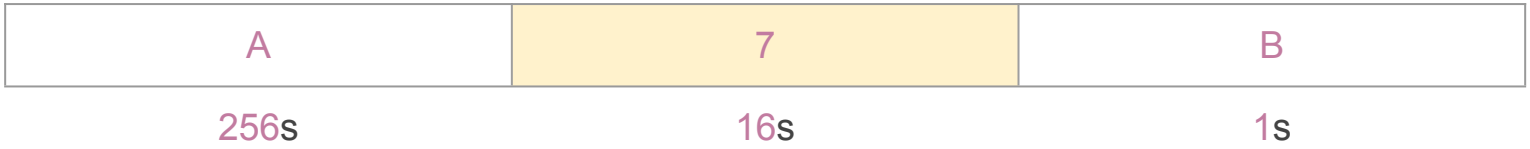


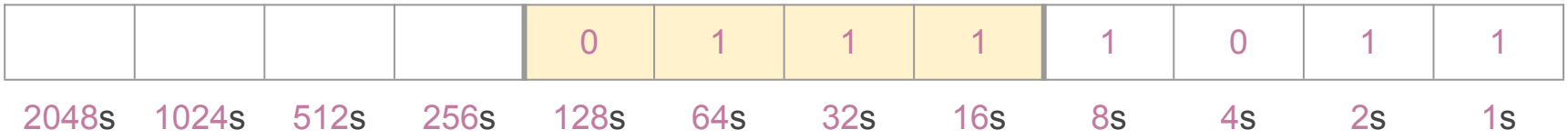


Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$

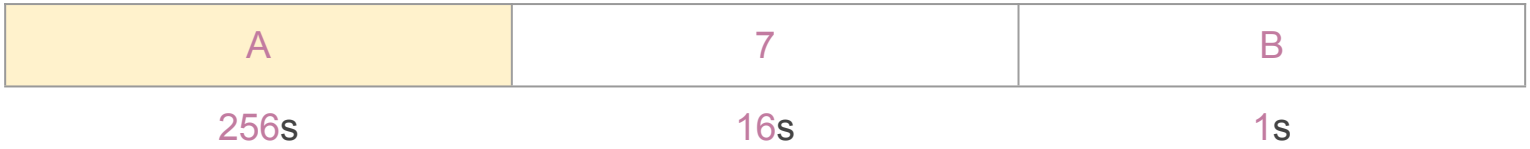


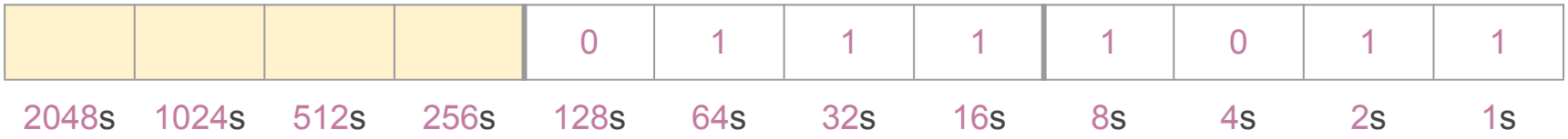


Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$

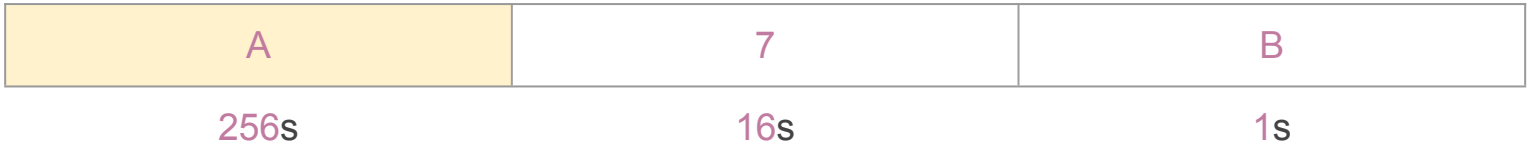


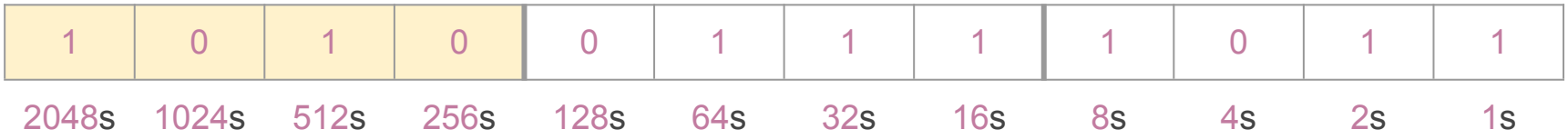


Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16}$





Hex to Binary

To convert a number in Hex to Binary, you just need to take each digit and convert it to its Binary equivalent!

Ex: $A7B_{16} = 101001111011_2$

A	7	B
256s	16s	1s

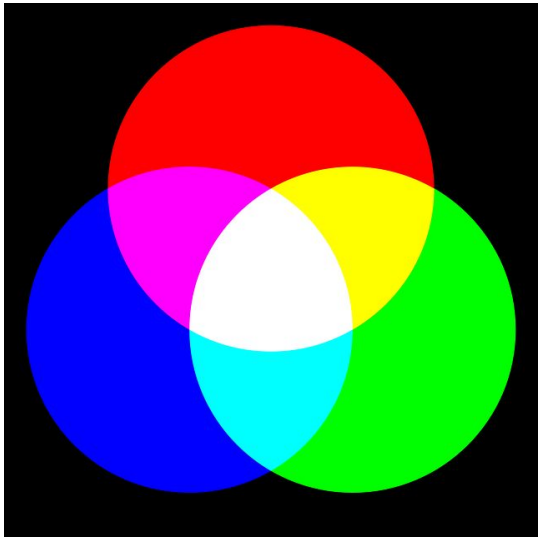
1	0	1	0	0	1	1	1	1	0	1	1
2048s	1024s	512s	256s	128s	64s	32s	16s	8s	4s	2s	1s

Let's Talk About Color

RGB Encoding

One of the most common methods of encoding colors as numbers is **RGB**.

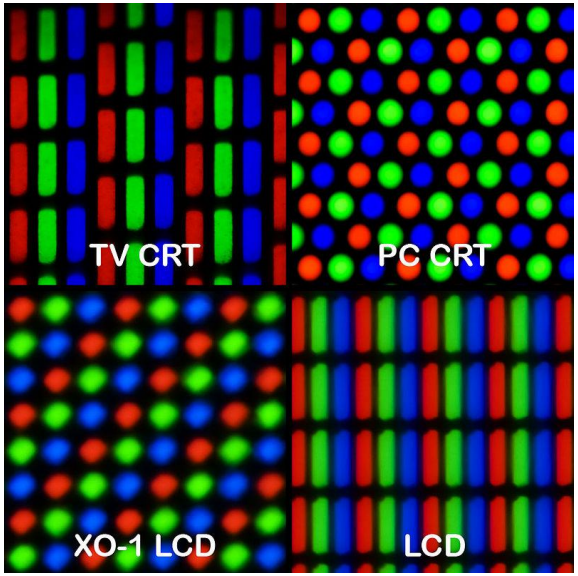
Any color can be created by combining those 3 colors!



RGB Encoding

One of the most common methods of encoding colors as numbers is **RGB**.

Every pixel on any display device actually has R, G, and B



Source: http://commons.wikimedia.org/wiki/File:Pixel_geometry_01_Pengo.jpg Peter Halasz

RGB Encoding

One of the most common methods of encoding colors as numbers is **RGB**.

An RGB code consists of 3 parts:

RGB Encoding

One of the most common methods of encoding colors as numbers is **RGB**.

An RGB code consists of 3 parts:

- The amount of **Red** to add
-
-

RGB Encoding

One of the most common methods of encoding colors as numbers is **RGB**.

An RGB code consists of 3 parts:

- The amount of **Red** to add
- The amount of **Green** to add
-

RGB Encoding

One of the most common methods of encoding colors as numbers is **RGB**.

An RGB code consists of 3 parts:

- The amount of **Red** to add
- The amount of **Green** to add
- The amount of **Blue** to add

RGB Encoding

One of the most common methods of encoding colors as numbers is **RGB**.

An RGB code consists of 3 parts:

- The amount of **Red** to add
- The amount of **Green** to add
- The amount of **Blue** to add

Each color channel gets a value in the range 0_{10}- 255_{10}.
One channel can be represented using 1 Binary byte: 00000000_{2} - 11111111_{2}
If we use Hex, we can represent a channel using **2 digits**: 00_{16} - FF_{16}

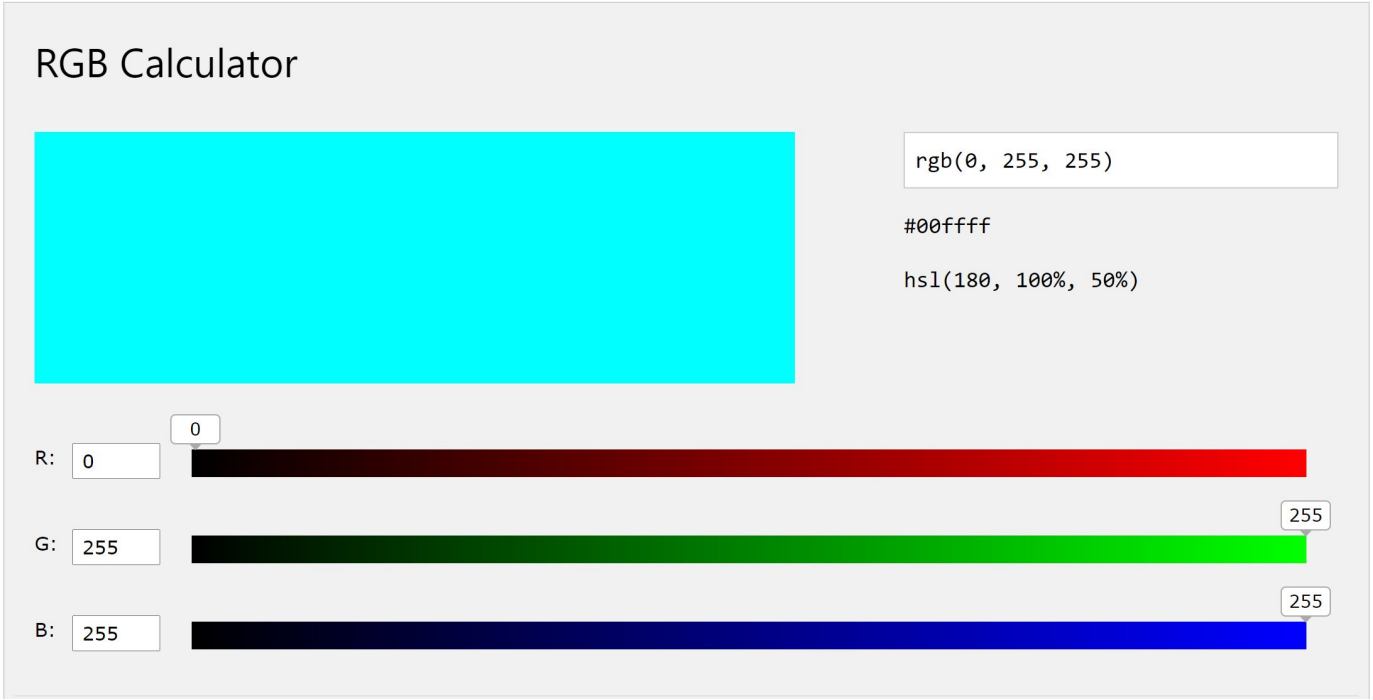
Visual Example

Source: https://www.w3schools.com/colors/colors_rgb.asp

Visual Example

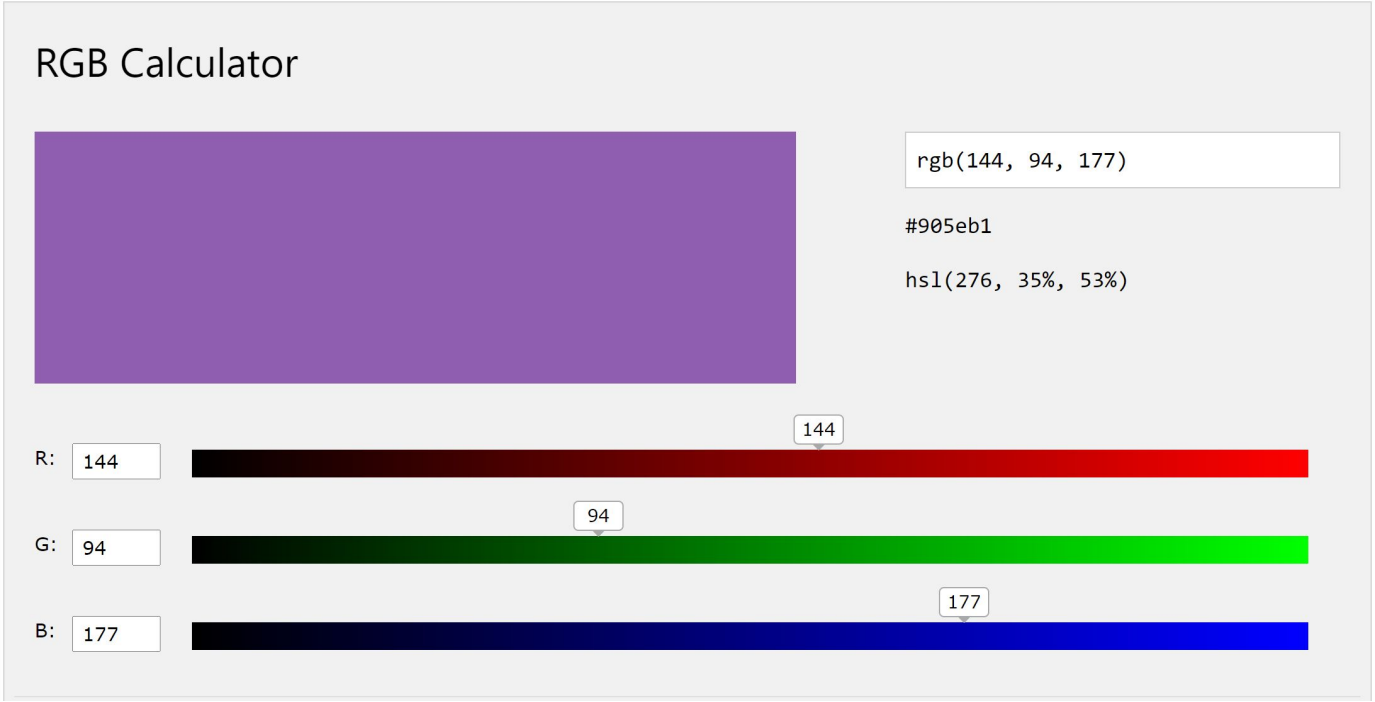
Source: https://www.w3schools.com/colors/colors_rgb.asp

Visual Example



Source: https://www.w3schools.com/colors/colors_rgb.asp

Visual Example



Source: https://www.w3schools.com/colors/colors_rgb.asp

Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000 00FF00 0000FF`

Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

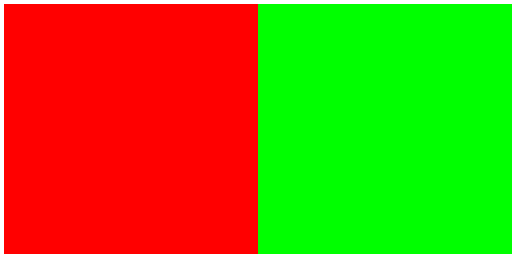
`FF0000 00FF00 0000FF`

Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000 00FF00 0000FF`

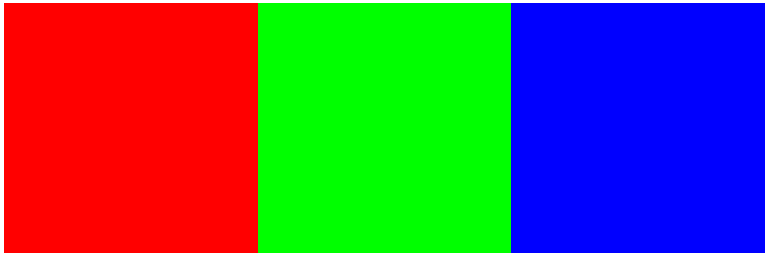


Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000 00FF00 0000FF`

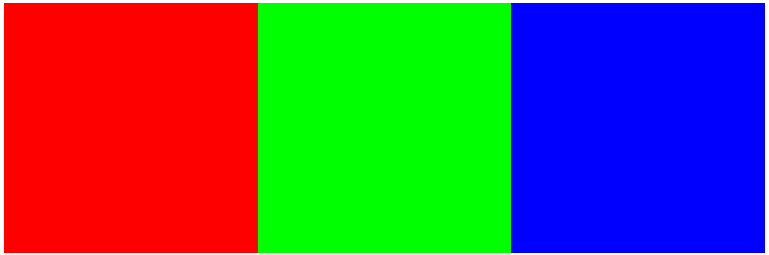


Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000` `00FF00` `0000FF`

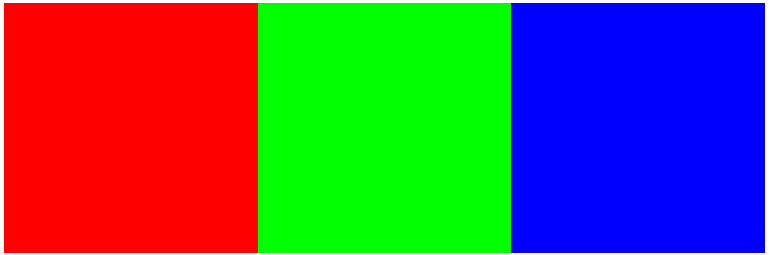


Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000` `00FF00` `0000FF` `000000`

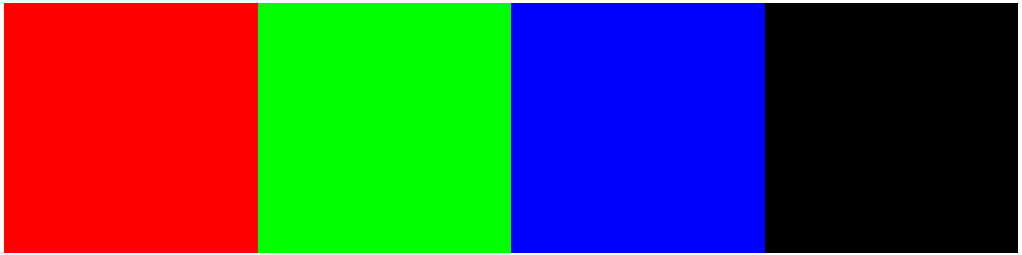


Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000` `00FF00` `0000FF` `000000`

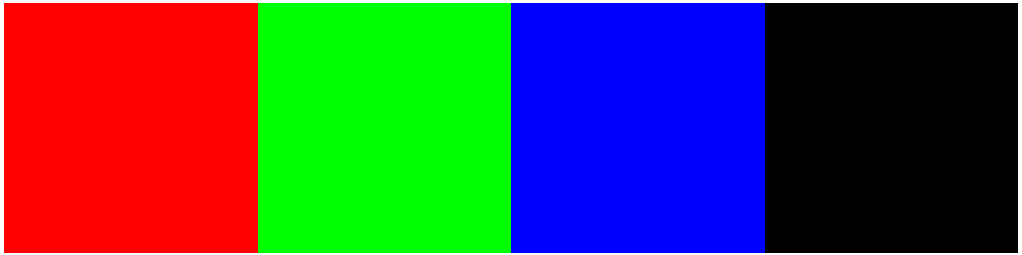


Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000` `00FF00` `0000FF` `000000` `FFFFFF`



Applying This

With this in mind, we can write sequences of Binary or Hex that will represent different pixels on our canvas!

What would the following sequence look like?

`FF0000` `00FF00` `0000FF` `000000` `FFFFFF`