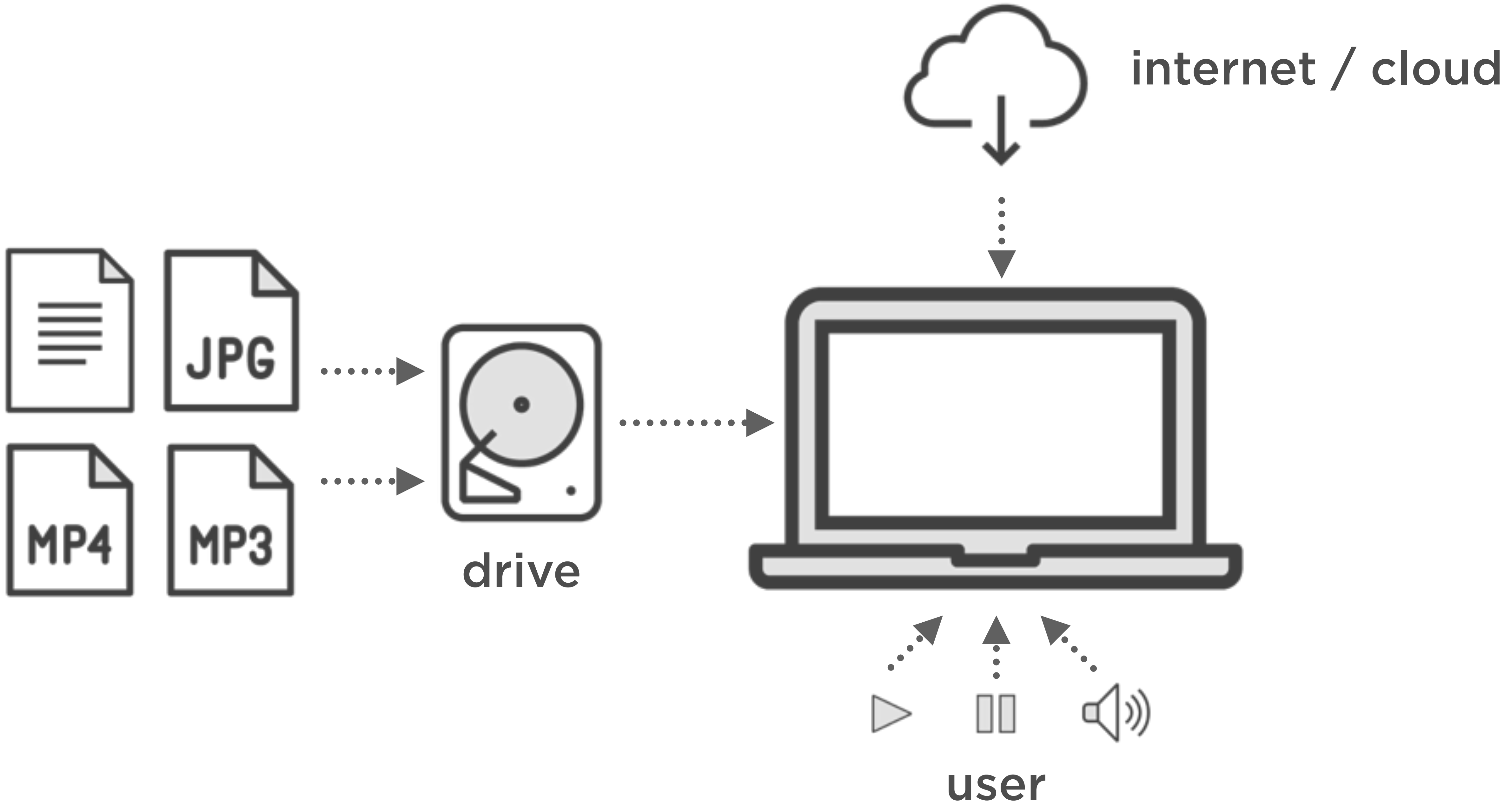
Working with Data

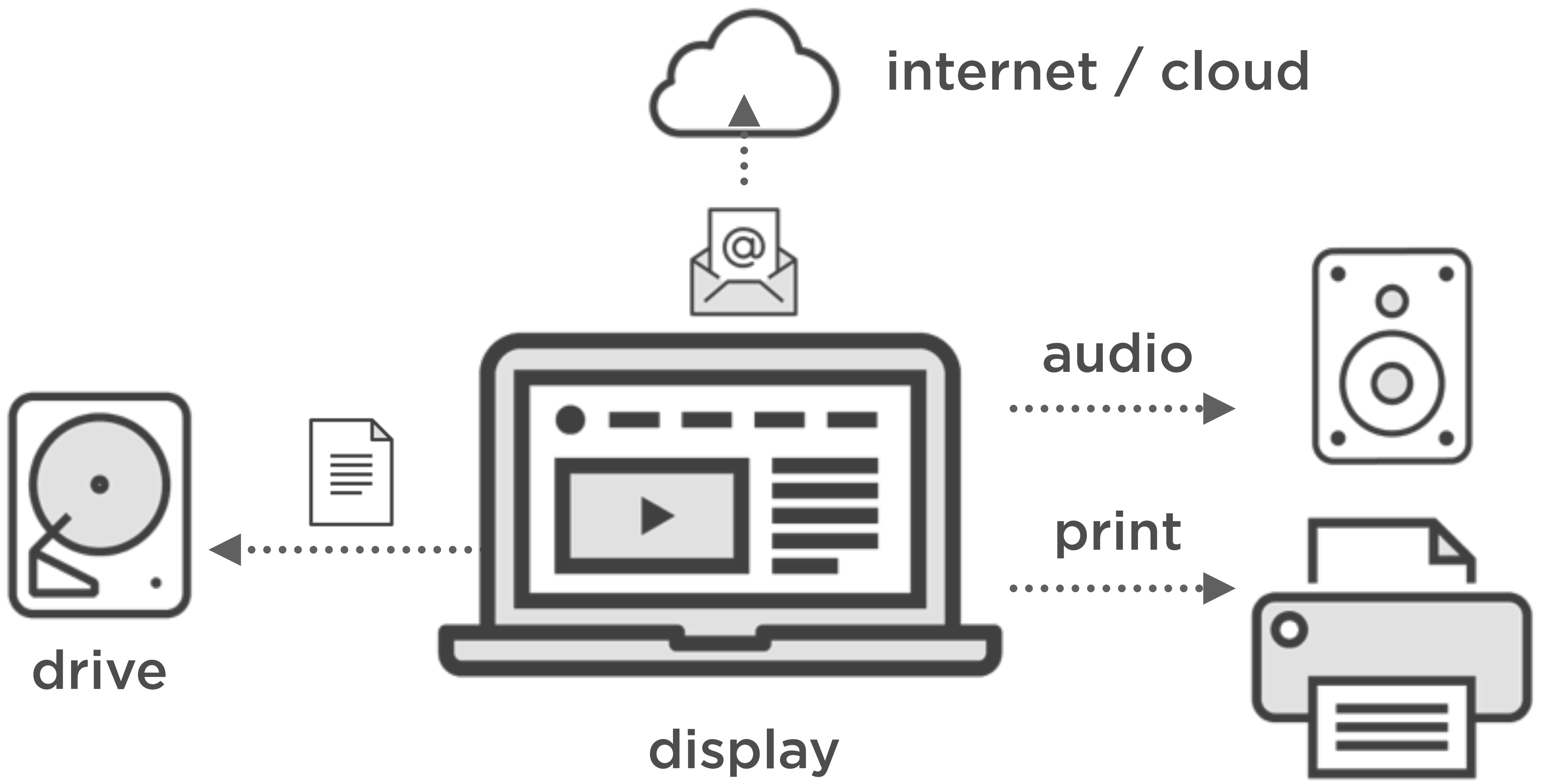
INTRO: INPUT, OUTPUT AND EVERYTHING IN-BETWEEN

Simon Allardice
STAFF AUTHOR, PLURALSIGHT
@allardice www.pluralsight.com

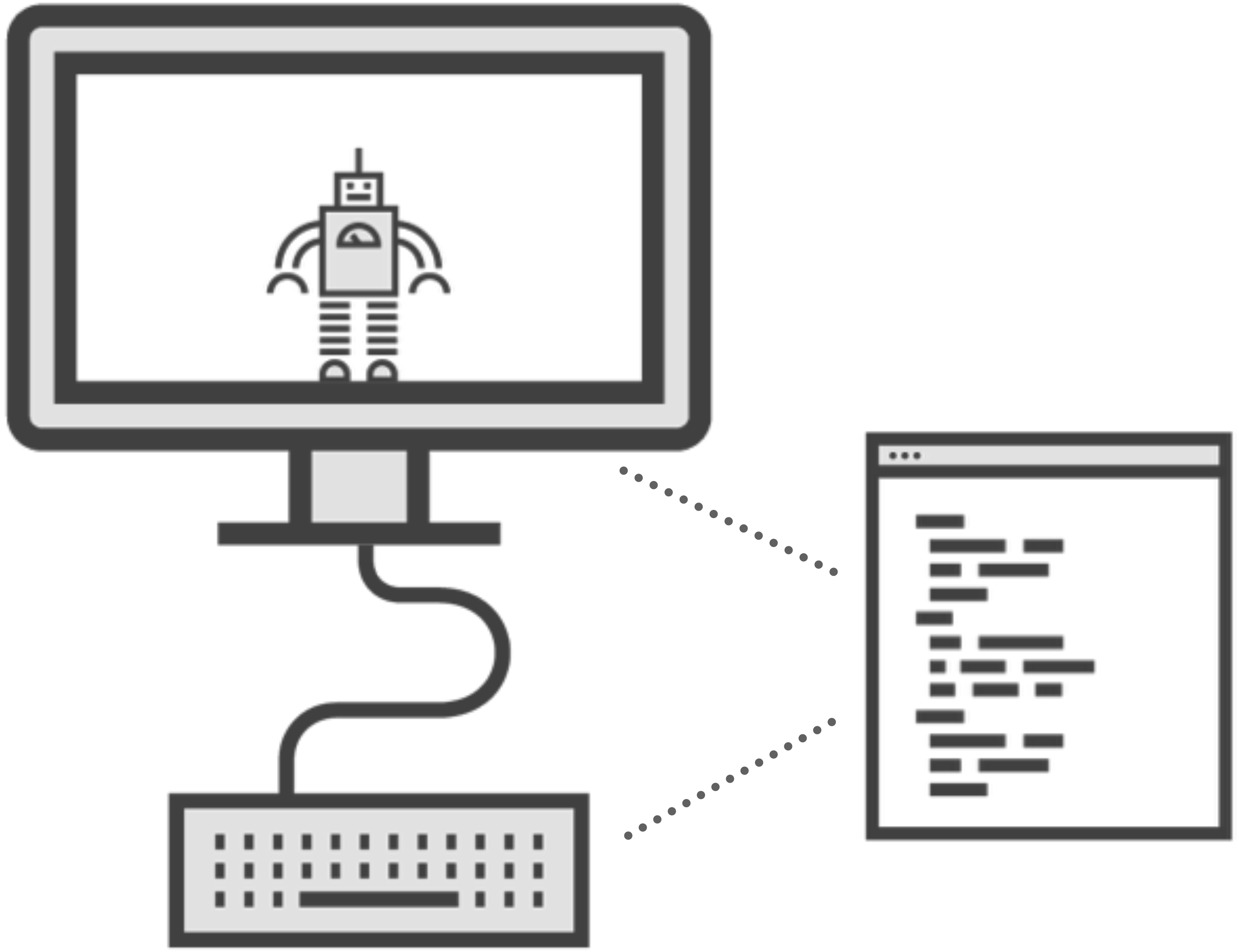
Sources of Input



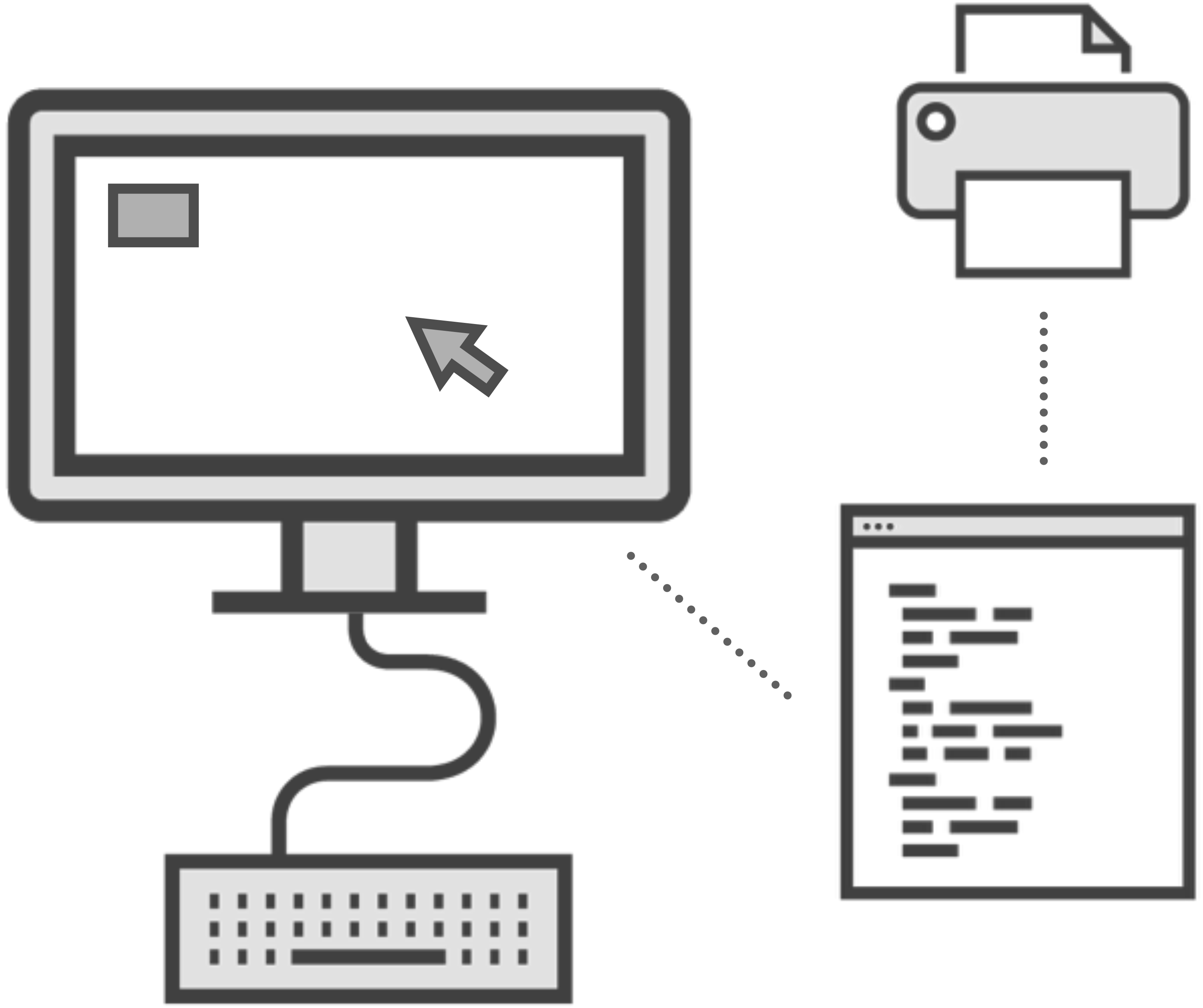
Sources of Output



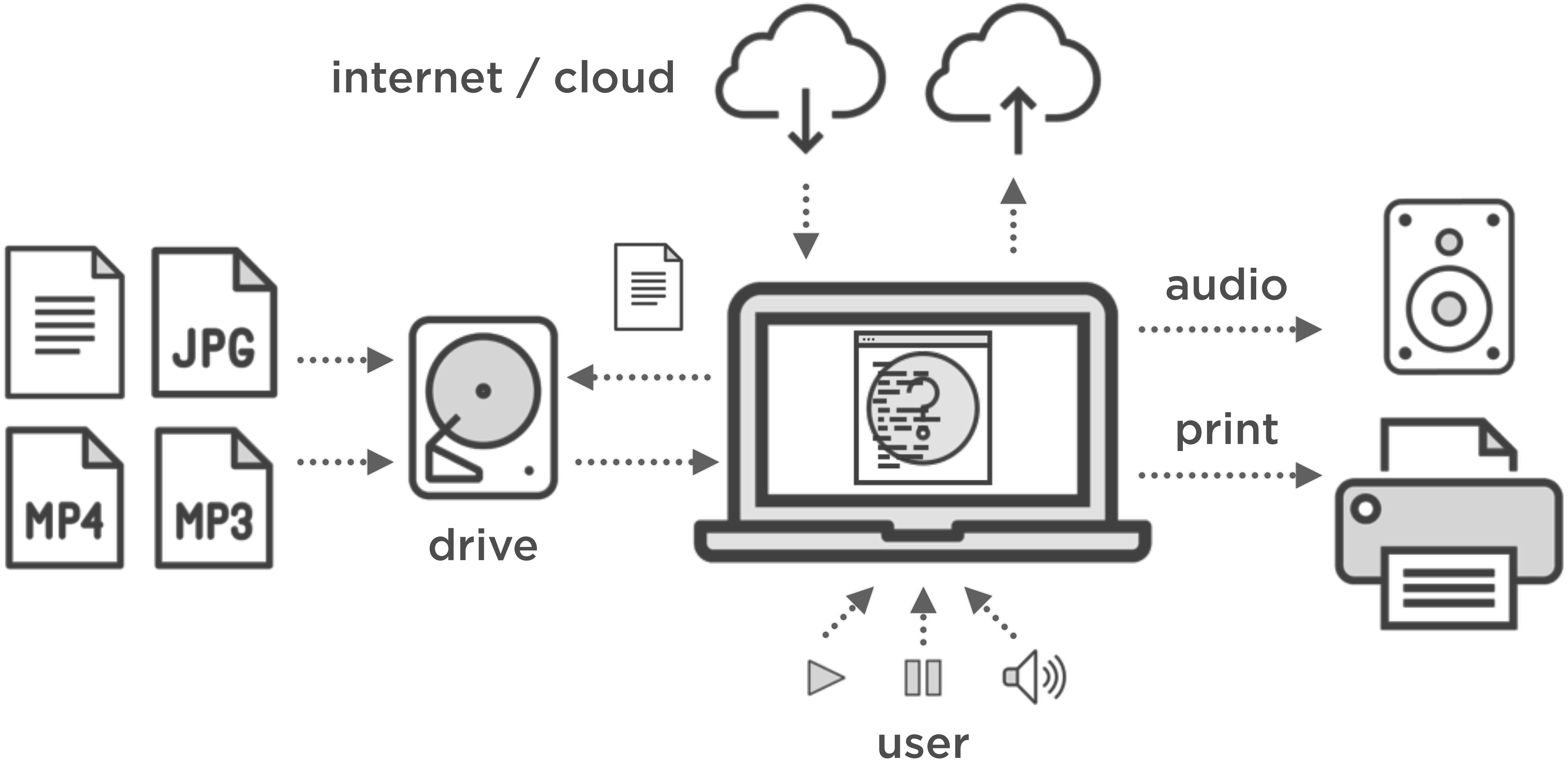
How Do We Turn Input Into Output?



How Do We Turn Input Into Output?



It's All Our Data



Data: Creating and Naming Variables

Data: Creating and Naming Variables

3 Important Things About Variables

3 Important Things About Variables

1: **Name**

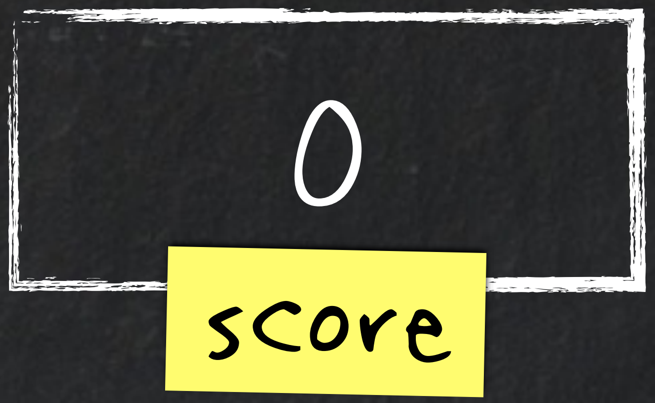
3 Important Things About Variables

1: **Name** 2: **Value**

3 Important Things About Variables

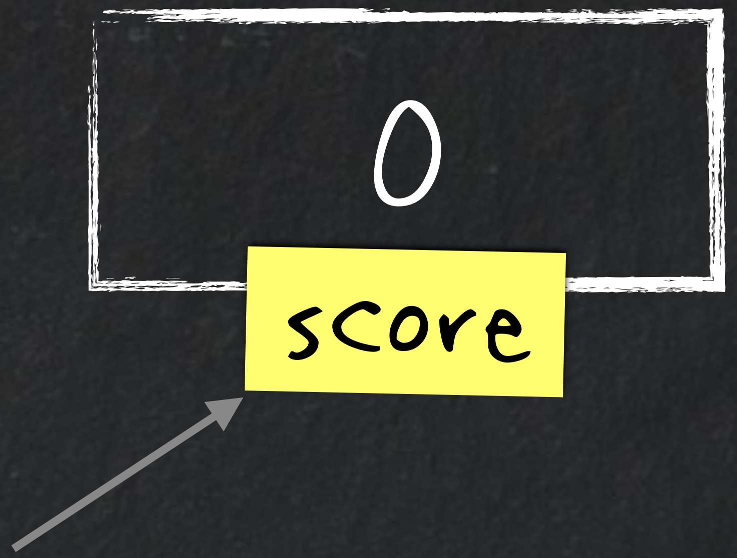
1: **Name** 2: **Value** 3: **Type**

3 Important Things About Variables



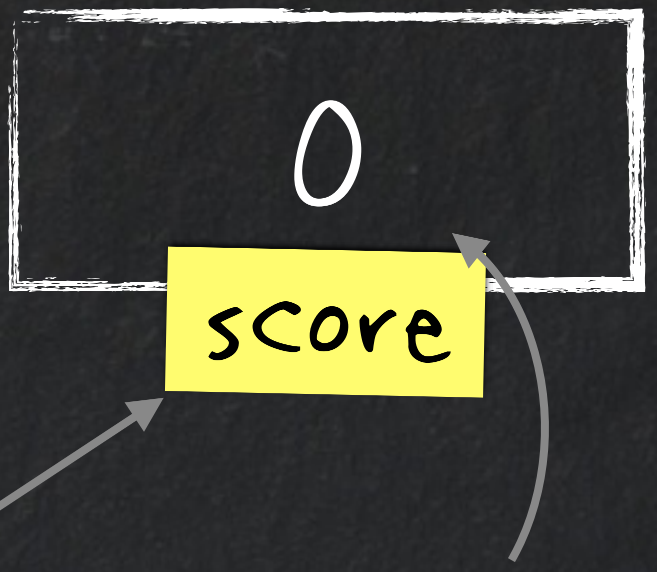
1: **Name** 2: **Value** 3: **Type**

3 Important Things About Variables



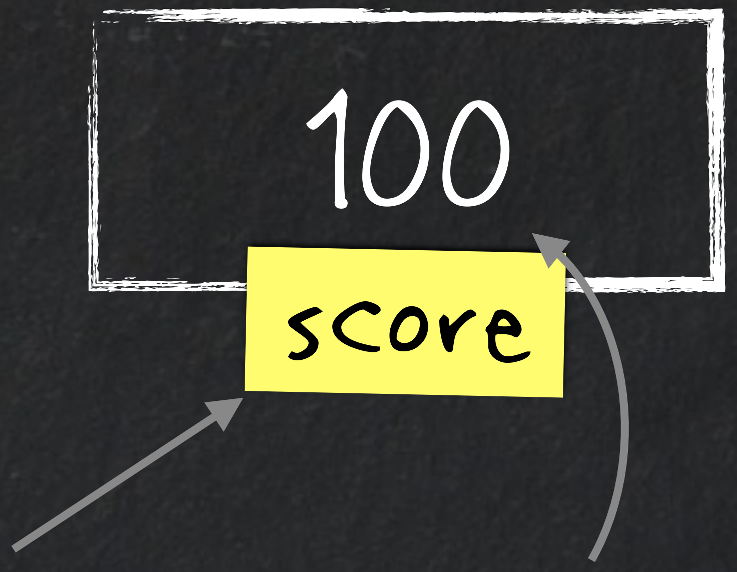
1: Name   2: Value   3: Type

3 Important Things About Variables



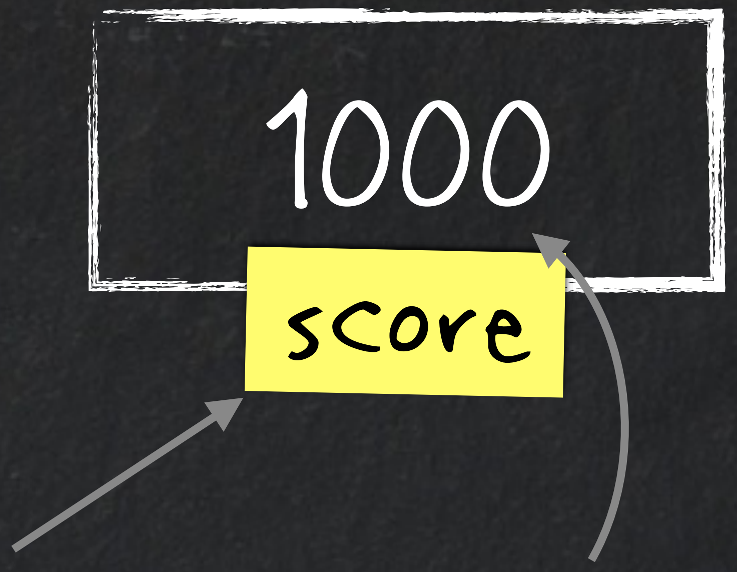
1: Name 2: Value 3: Type

3 Important Things About Variables



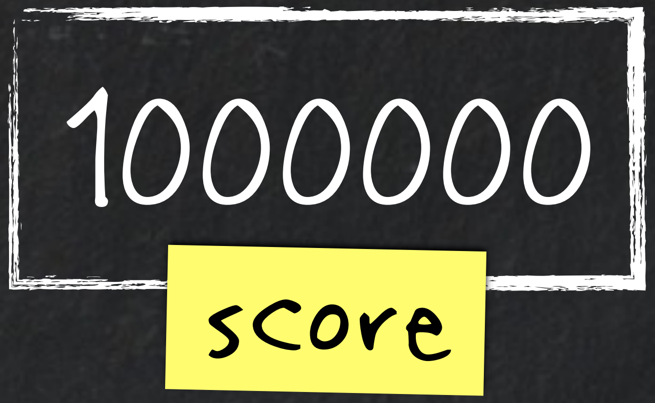
1: Name   2: Value   3: Type

3 Important Things About Variables



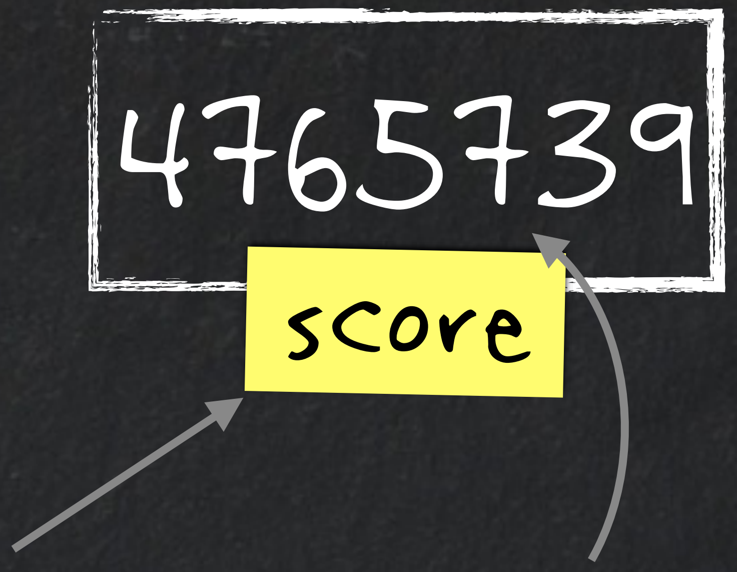
1: Name   2: Value   3: Type

3 Important Things About Variables



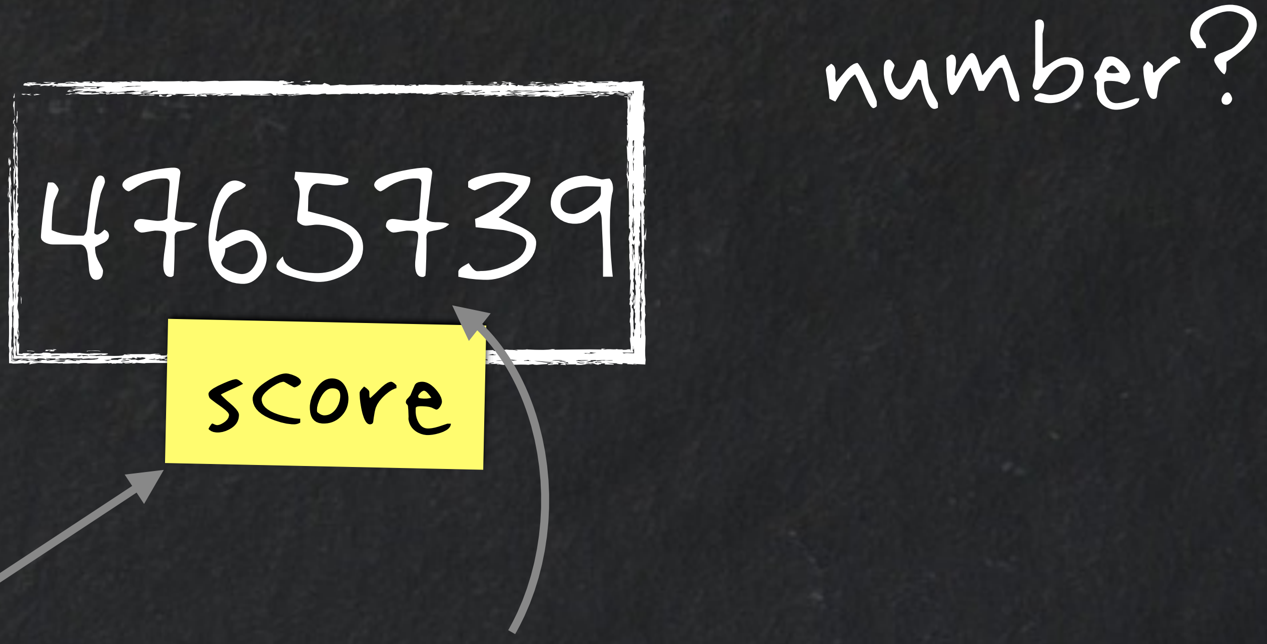
1: Name **2: Value** **3: Type**

3 Important Things About Variables



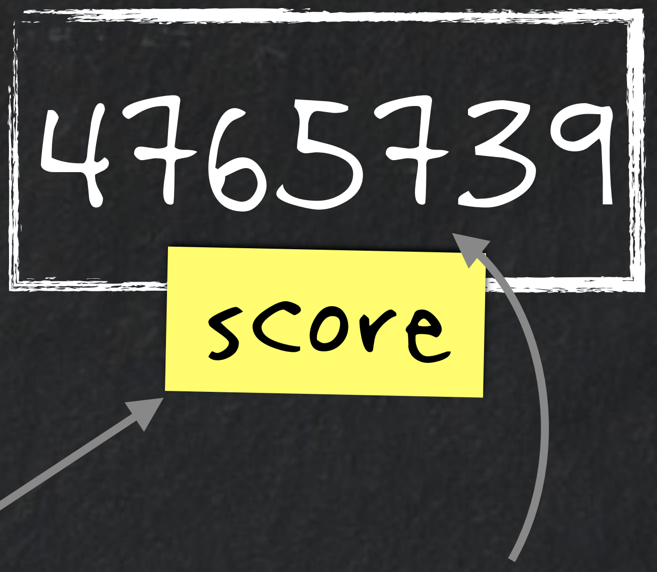
1: Name **2: Value** **3: Type**

3 Important Things About Variables



1: Name **2: Value** **3: Type**

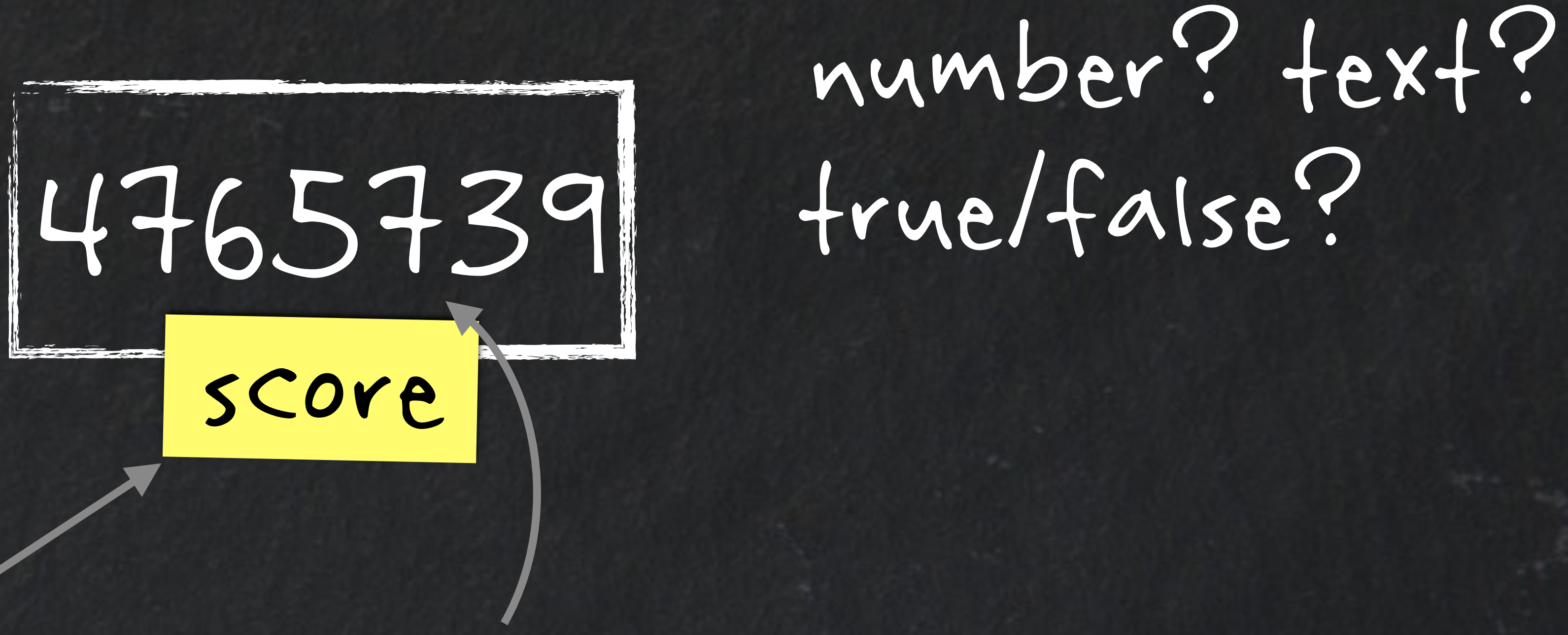
3 Important Things About Variables



number? text?

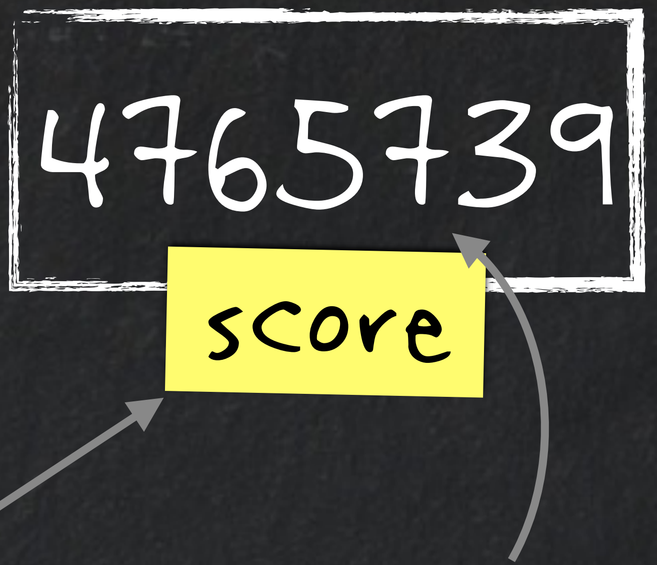
1: Name 2: Value 3: Type

3 Important Things About Variables



1: Name **2: Value** **3: Type**

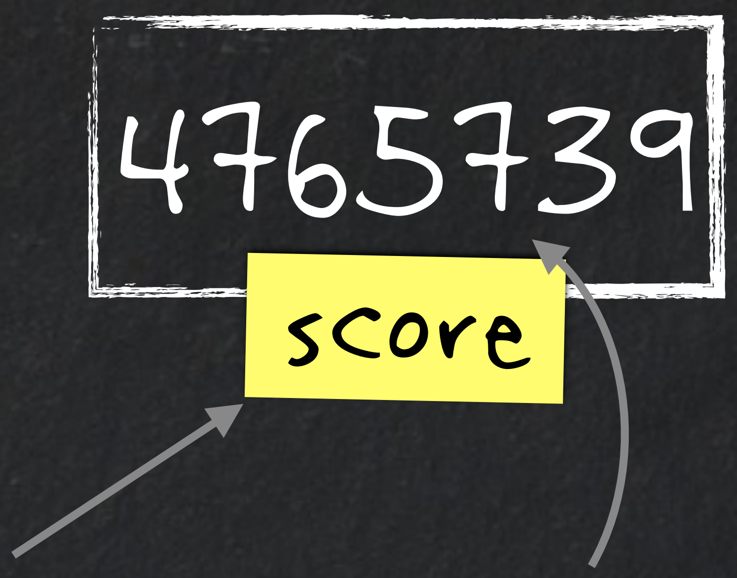
3 Important Things About Variables



number? text?
true/false? currency?

1: Name 2: Value 3: Type

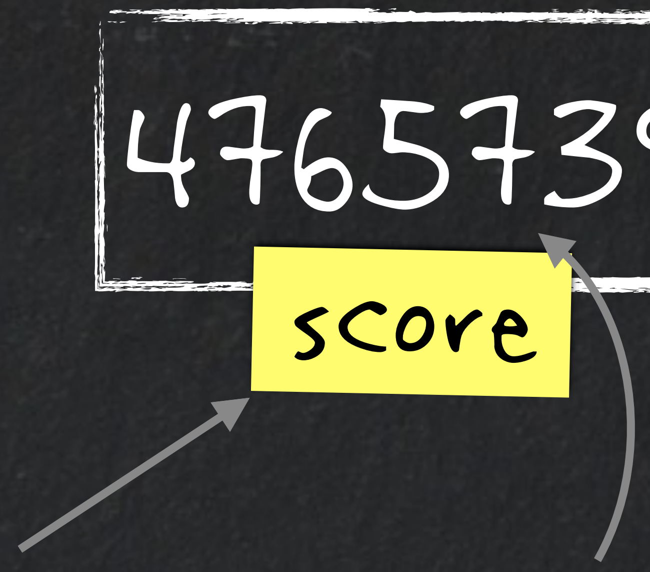
3 Important Things About Variables



1: Name 2: Value 3: Type

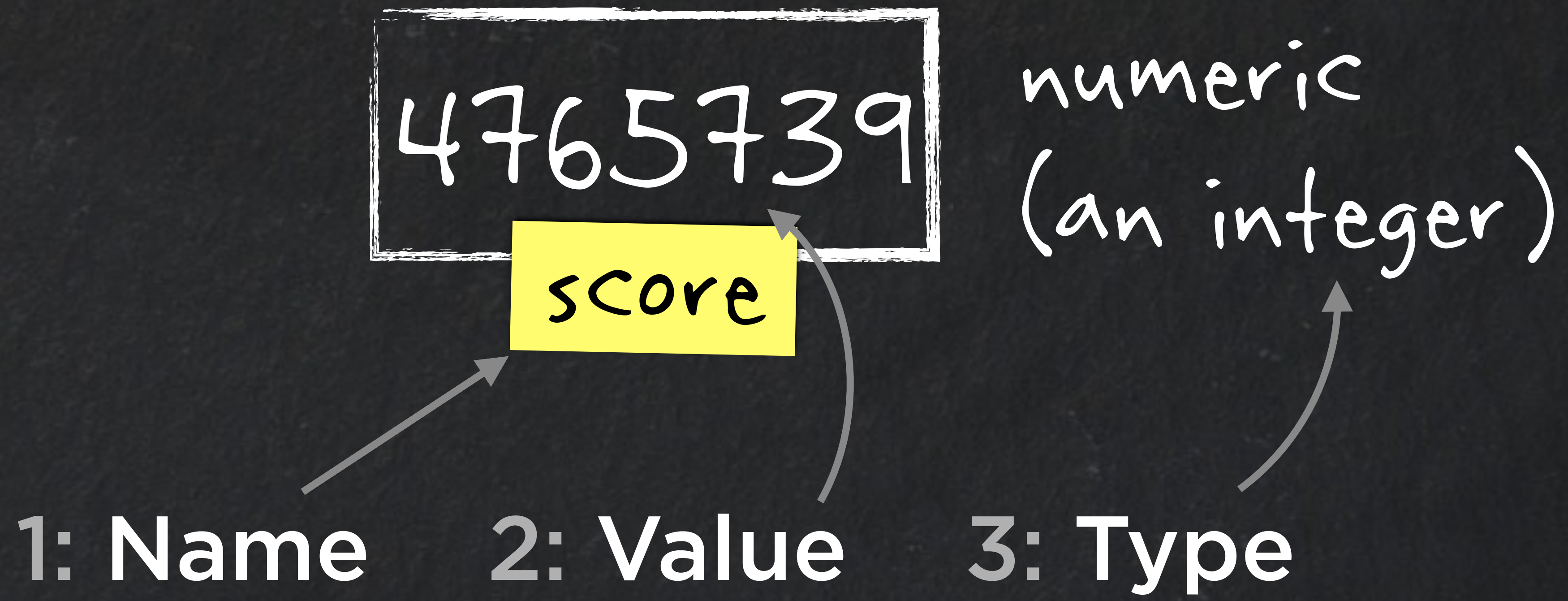
number? text?
true/false? currency?
something else?

3 Important Things About Variables



1: Name   2: Value   3: Type

3 Important Things About Variables



Declaring a Variable

Swift

```swift
var score: Int
```

Visual Basic

```vb
Dim score As Integer
```

C++

```cpp
int score;
```

JavaScript

```javascript
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Memory Requirements

Memory Requirements

Memory Requirements

Memory Requirements

Memory Requirements

Memory Requirements

Declaring a Variable

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Choosing a Variable Name (the *Identifier*)

Choosing a Variable Name (the *Identifier*)

Rules (Syntax)

What are you **allowed** to do?

Choosing a Variable Name (the *Identifier*)

Rules (Syntax)

What are you allowed to do?

Guidelines (Style)

What are you supposed to do?

```
// This variable name is within the rules
var CuRReNt_Sc0_rE;
```

Naming Variables

Just because you can, doesn't mean you should

Rule: **No Reserved Words / Keywords**

You can't use a word the language already owns

```
var if;      // not allowed
var true;    // not allowed
```

Rule: **No Reserved Words / Keywords**

You can't use a word the language already owns

Acceptable Variable Identifiers

Acceptable Variable Identifiers

```
score
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name

$score        // PHP requires leading $
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name

$score        // PHP requires leading $
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name

$score        // PHP requires leading $

highscore
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name

$score        // PHP requires leading $

highscore
high_score   // With multiple words,
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name

$score        // PHP requires leading $

highscore
high_score   // With multiple words,
highScore    // there are different "styles".
```

Acceptable Variable Identifiers

```
score
Score  // case sensitive: "score" is different from "Score"

x      // Nothing enforces a meaningful name

$score        // PHP requires leading $

highscore
high_score   // With multiple words,
highScore    // there are different "styles".
HighScore    // Most languages have a preferred style.
```

Non-Roman Alphabet Identifiers

```
// Swift Variable Names

var 如此这般      // Mandarin
var צ'וּפְּצִ'יק    // Hebrew
var كذا          // Arabic
var 😃☹️          // Emoji
```

Rule: **Don't Start with a Digit**

In most languages, you can use numbers but you cannot begin with one

```
var rule22;     // is allowed

var 22ndrule;   // is not allowed
```

Rule: **Don't Start with a Digit**

In most languages, you can use numbers but you cannot begin with one

Style Example: **Camel Case**

Not required, but commonly seen in many languages

```
var score          // one word - lower case
var name
var department
```

Style Example: **Camel Case**

Not required, but commonly seen in many languages

```
var score           // one word - lower case
var name
var department

var highScore                // multiple words - capitalize
var firstName                // every word but the first
var veryLongVariableName
```

Style Example: **Camel Case**

Not required, but commonly seen in many languages

Data: Using Variables and Operators

Providing an Initial Value

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Providing an Initial Value

Swift

```
var score: Int
score = 0
```

Visual Basic

```
Dim score As Integer
score = 0
```

C++

```
int score;
score = 0;
```

JavaScript

```
var score;
 score = 0;
```

Providing an Initial Value

Swift

```
var score: Int
score = 0
```

Visual Basic

```
Dim score As Integer
score = 0
```

C++

```
int score;
score = 0;
```

JavaScript

```
var score;
score = 0;
```

Operators

A shortcut to perform a specific task - one operation

```
100 + 75
```

Operators

A shortcut to perform a specific task - one operation

```
100 + 75
```

addition operator

Operators

A shortcut to perform a specific task - one operation

```
100 + 75
```

addition operator

```
500 - 300
```

Operators

A shortcut to perform a specific task - one operation

```
100 + 75
```

addition operator

```
500 - 300
```

subtraction operator

Operators

A shortcut to perform a specific task - one operation

```
100 + 75
```

addition operator

```
500 - 300
```

subtraction operator

```
100 * 5
```

multiplication operator

Operators

A shortcut to perform a specific task - one operation

```
100 + 75
```
addition operator

```
500 - 300
```
subtraction operator

```
100 * 5
```
multiplication operator

```
100 / 10
```
division operator

Operators

A shortcut to perform a specific task - one operation

```
100 + 75                100 * 5

500 - 300               100 / 10
```

Operators

A shortcut to perform a specific task - one operation

```
100 + 75                    100 * 5

500 - 300                   100 / 10
```

Operators

A shortcut to perform a specific task - one operation

```
100 + 75

500 - 300
```

```
100 * 5

100 / 10
```

Operators

A shortcut to perform a specific task - one operation

Operators

A shortcut to perform a specific task - one operation

```
100 + 75
500 - 300
```

```
100 * 5
100 / 10
```

Binary operators require two values (operands) to work

```
500 - 300
```

Operators

A shortcut to perform a specific task - one operation

```
500 - 300
```

result : 200

Operators

A shortcut to perform a specific task - one operation

```
500 - 300
300 - 500
```

result : 200

Operators

A shortcut to perform a specific task - one operation

```
500 - 300
300 - 500
```

result : 200

result : -200

Operators

A shortcut to perform a specific task - one operation

Assignment Operator (Single Equals Sign)

The shortcut to assign (set) a new value

```
score = 0
```

Assignment Operator (Single Equals Sign)

The shortcut to assign (set) a new value

```
score = 0    // score is now 0
```

Assignment Operator (Single Equals Sign)

The shortcut to assign (set) a new value

```
score = 0    // score is now 0

score = 105 + 42
```

Assignment Operator (Single Equals Sign)

The shortcut to assign (set) a new value

```
score = 0    // score is now 0

score = 105 + 42
```

Assignment Operator (Single Equals Sign)

The shortcut to assign (set) a new value

```
score = 0    // score is now 0

score = 105 + 42  147
```

Assignment Operator (Single Equals Sign)

The shortcut to assign (set) a new value

```
score = 0    // score is now 0

score = 105 + 42  // score is now 147
```

Assignment Operator (Single Equals Sign)

The shortcut to assign (set) a new value

Combining Declaration and Initial Value

Swift

```
var score: Int
score = 0
```

Visual Basic

```
Dim score As Integer
score = 0
```

C++

```
int score;
score = 0;
```

JavaScript

```
var score;
score = 0;
```

Combining Declaration and Initial Value

Swift

```
var score: Int
```

Visual Basic

```
Dim score As Integer
```

C++

```
int score;
```

JavaScript

```
var score;
```

Combining Declaration and Initial Value

Swift

```
var score: Int = 0
```

Visual Basic

```
Dim score As Integer = 0
```

C++

```
int score = 0;
```

JavaScript

```
var score = 0;
```

Combining Declaration and Initial Value

Swift

```
var score: Int = 0

score = 500
```

Visual Basic

```
Dim score As Integer = 0

score = 100 + 50
```

C++

```
int score = 0;

score = 5 * 50;
```

JavaScript

```
var score = 0;

score = score + 100;
```

Combining Declaration and Initial Value

Swift

```
var score: Int = 0
score = 500
```

Visual Basic

```
Dim score As Integer = 0
score = 100 + 50
```

C++

```
int score = 0;
score = 5 * 50;
```

JavaScript

```
var score = 0;
score = score + 100;
```

Combining Declaration and Initial Value

Swift

```
var score: Int = 0
score = 500
```

Visual Basic

```
Dim score As Integer = 0
score = 100 + 50
```

C++

```
int score = 0;
score = 5 * 50;
```

JavaScript

```
var score = 0;
score = score + 100;
```

Combining Declaration and Initial Value

Swift

```
var score: Int = 0
score = 500
```

Visual Basic

```
Dim score As Integer = 0
score = 100 + 50
```

C++

```
int score = 0;
score = 5 * 50;
```

JavaScript

```
var score = 0;
score = score + 100;
```

Combining Declaration and Initial Value

Swift

```
var score: Int = 0
score = 500
```

Visual Basic

```
Dim score As Integer = 0
score = 100 + 50
```

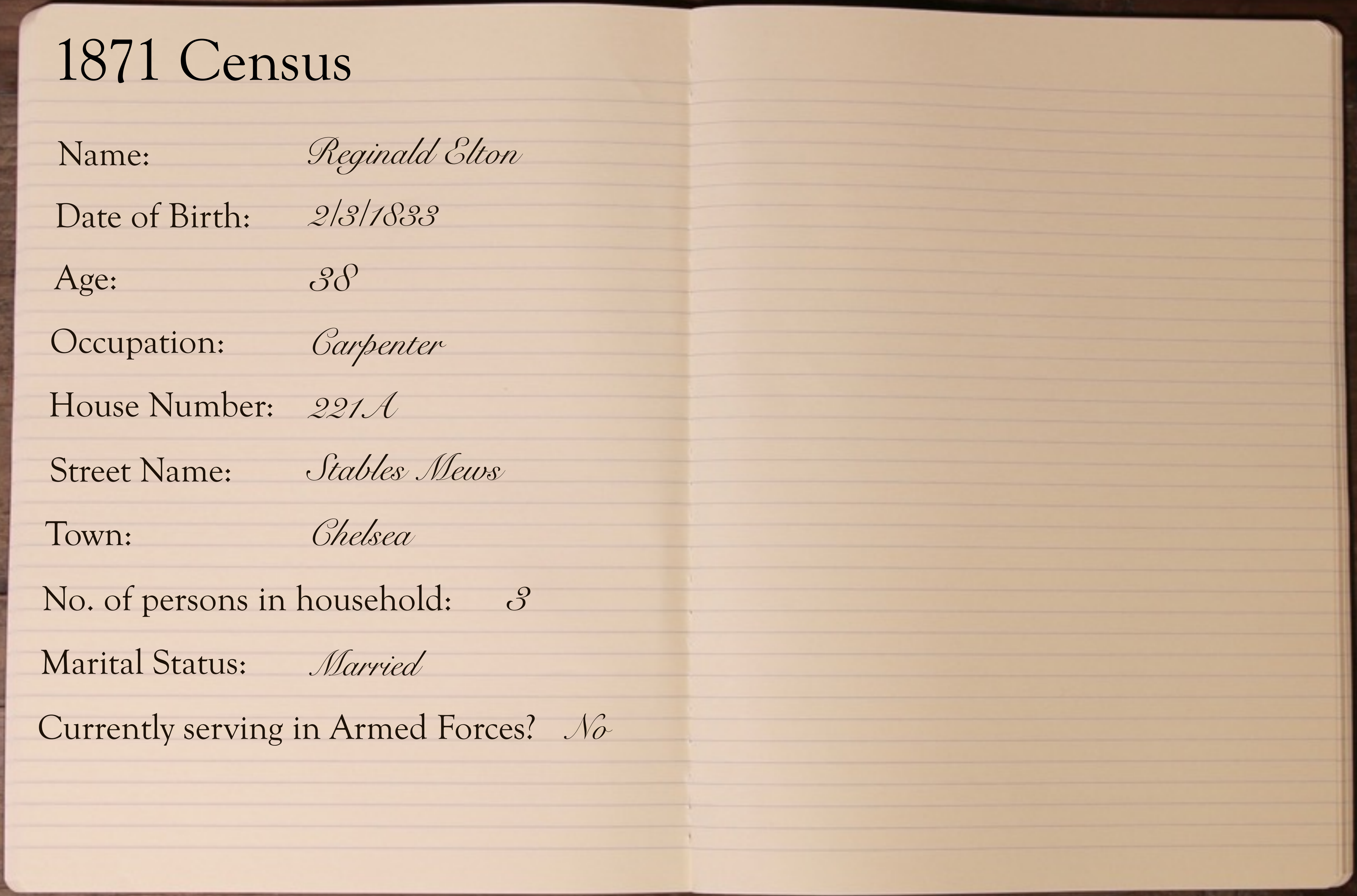
C++

```
int score = 0;
score = 5 * 50;
```

JavaScript

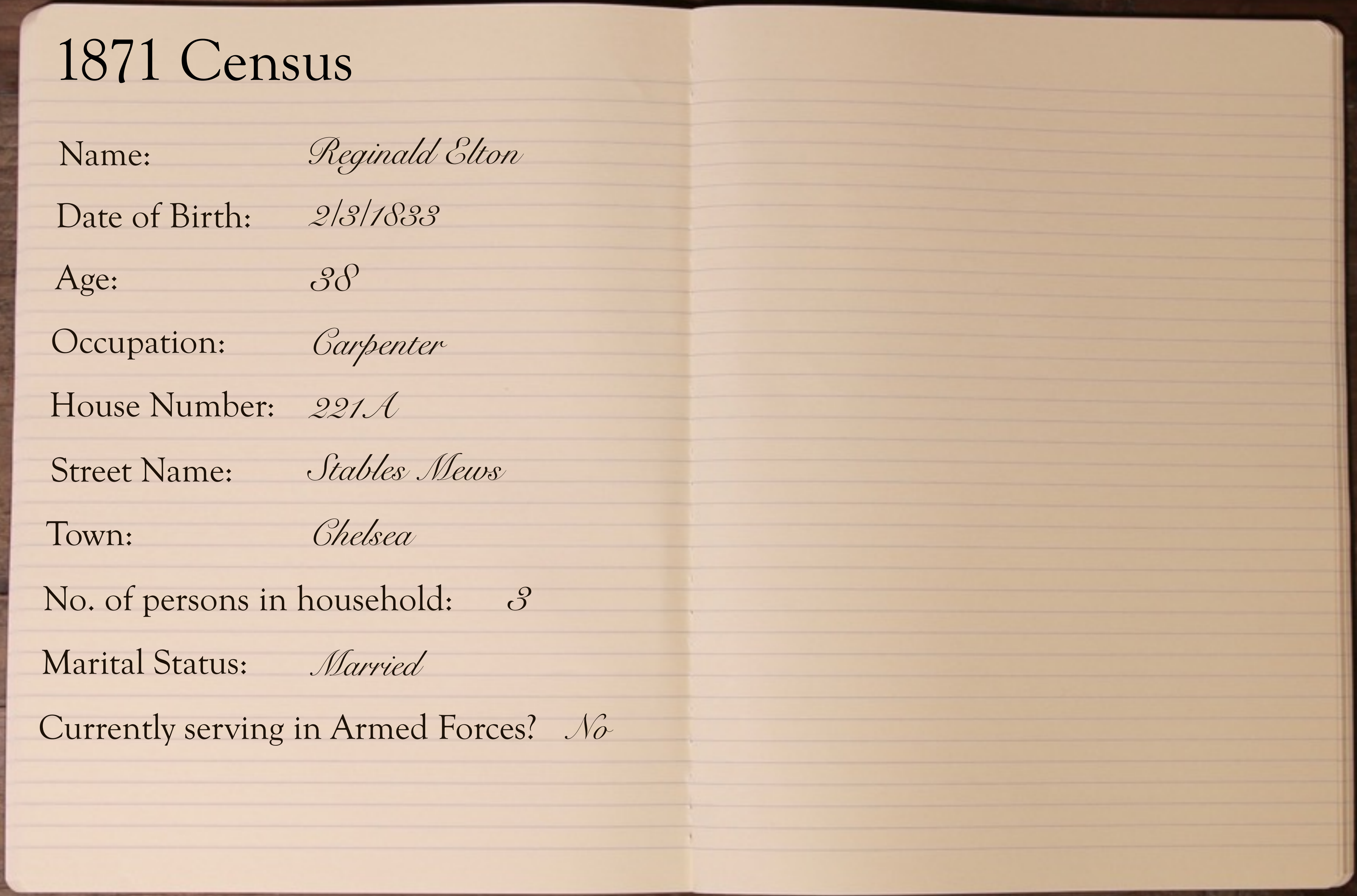
```
var score = 0;
score = score + 100;
```

Data: Choosing and Using Data Types



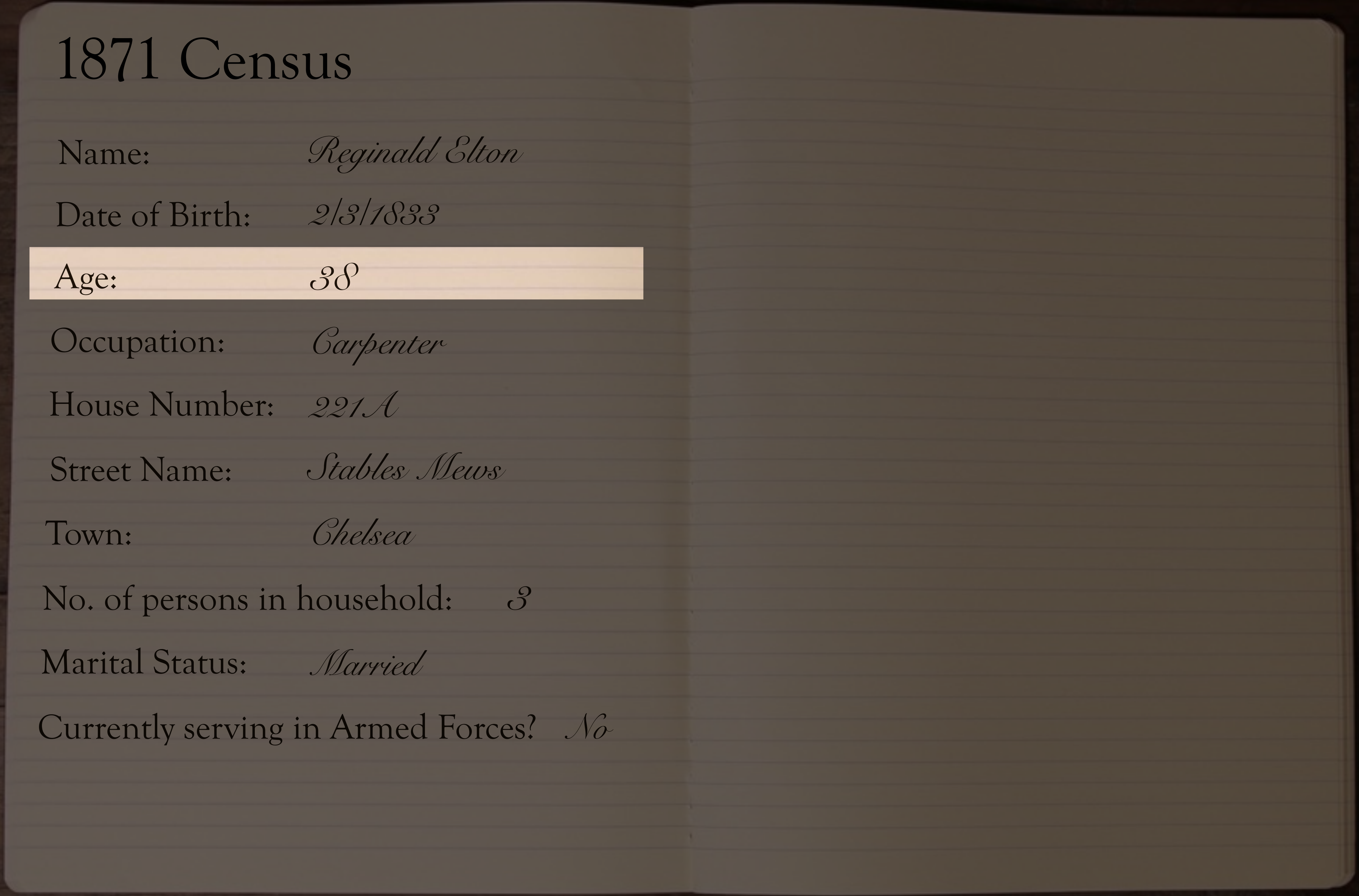
1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No



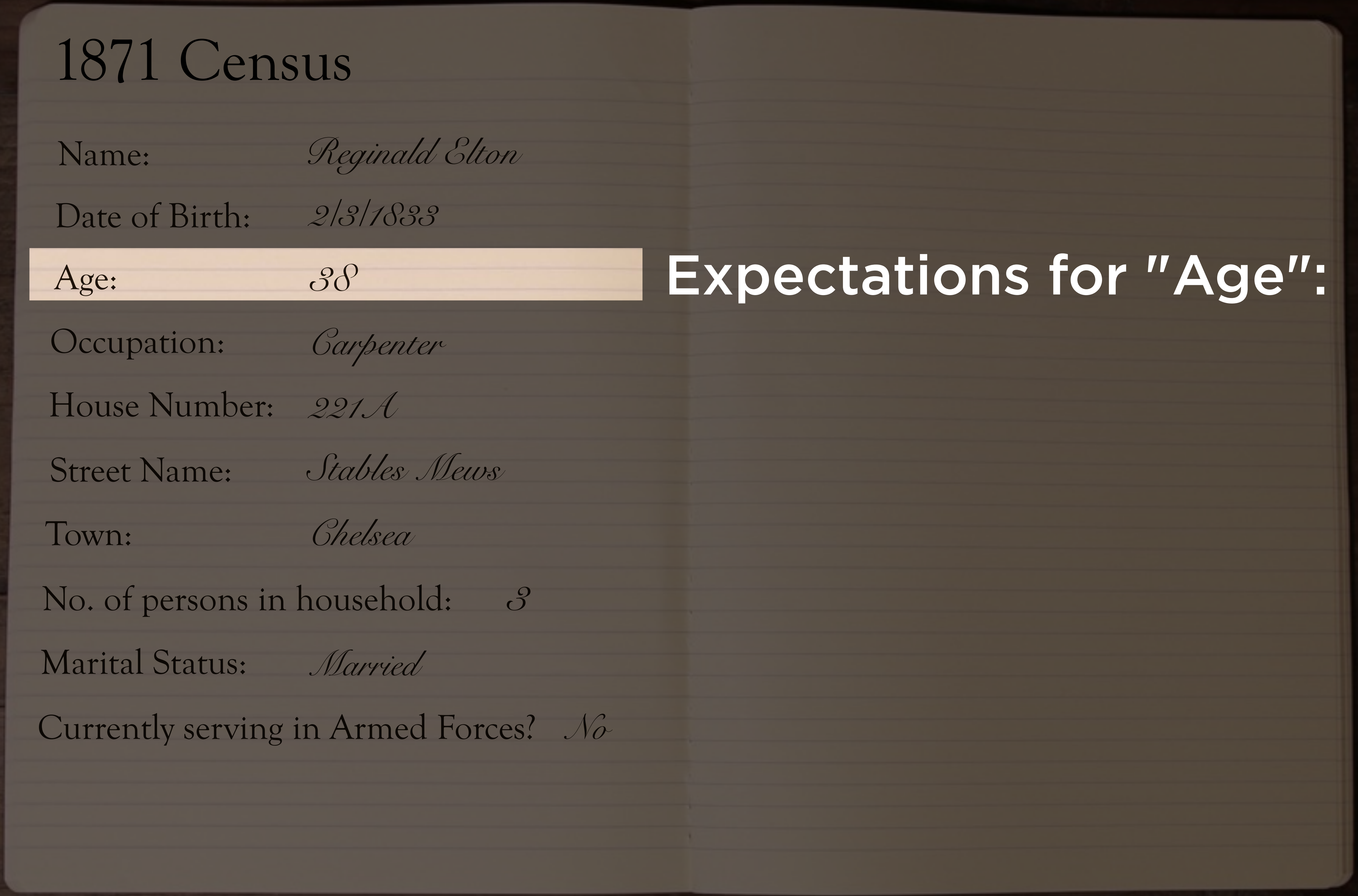
1871 Census

Name:	*Reginald Elton*
Date of Birth:	*2/3/1833*
Age:	*38*
Occupation:	*Carpenter*
House Number:	*221A*
Street Name:	*Stables Mews*
Town:	*Chelsea*
No. of persons in household:	*3*
Marital Status:	*Married*
Currently serving in Armed Forces?	*No*



1871 Census

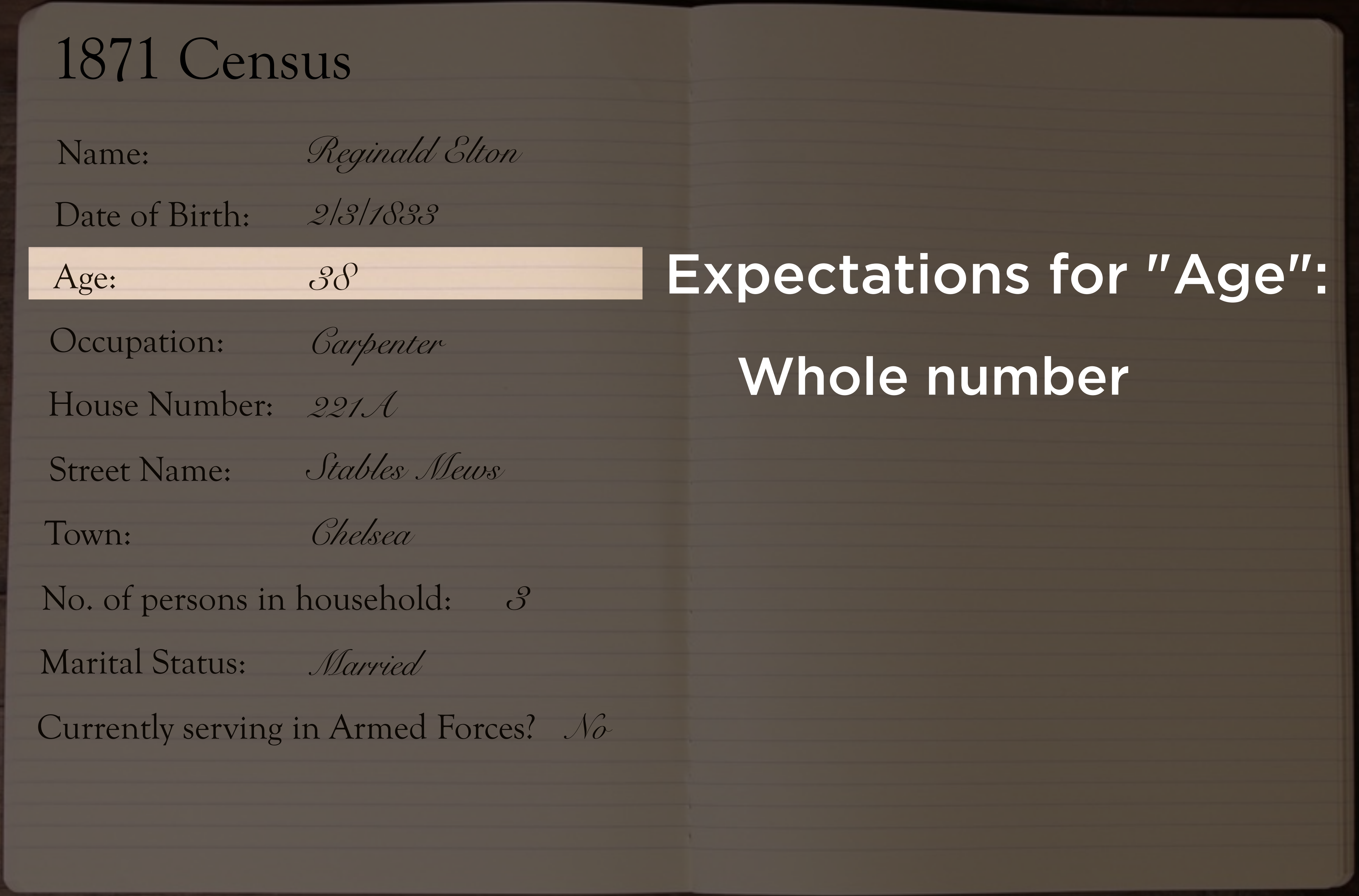
Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No



1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No

Expectations for "Age":

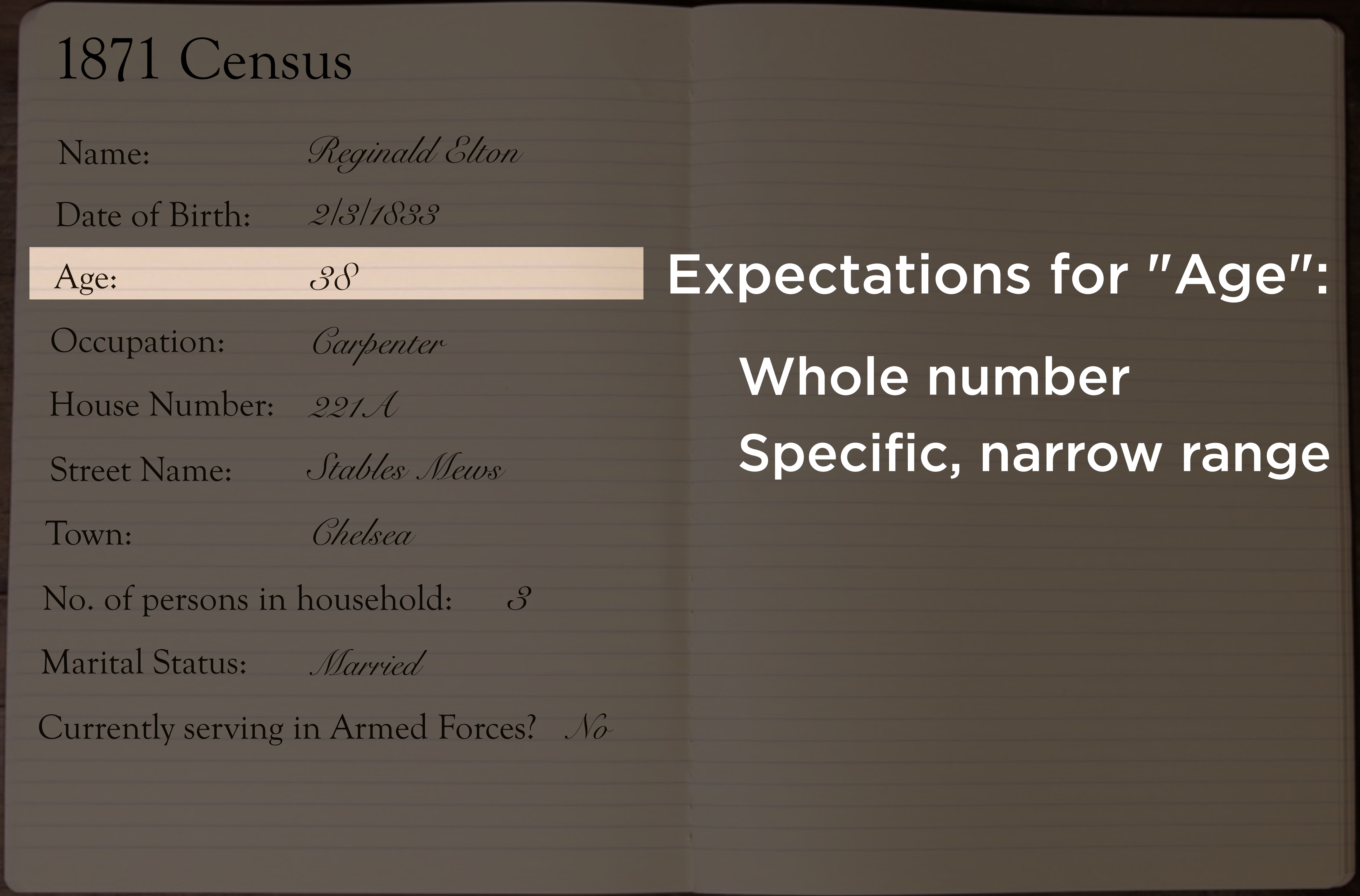


1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No

Expectations for "Age":

Whole number

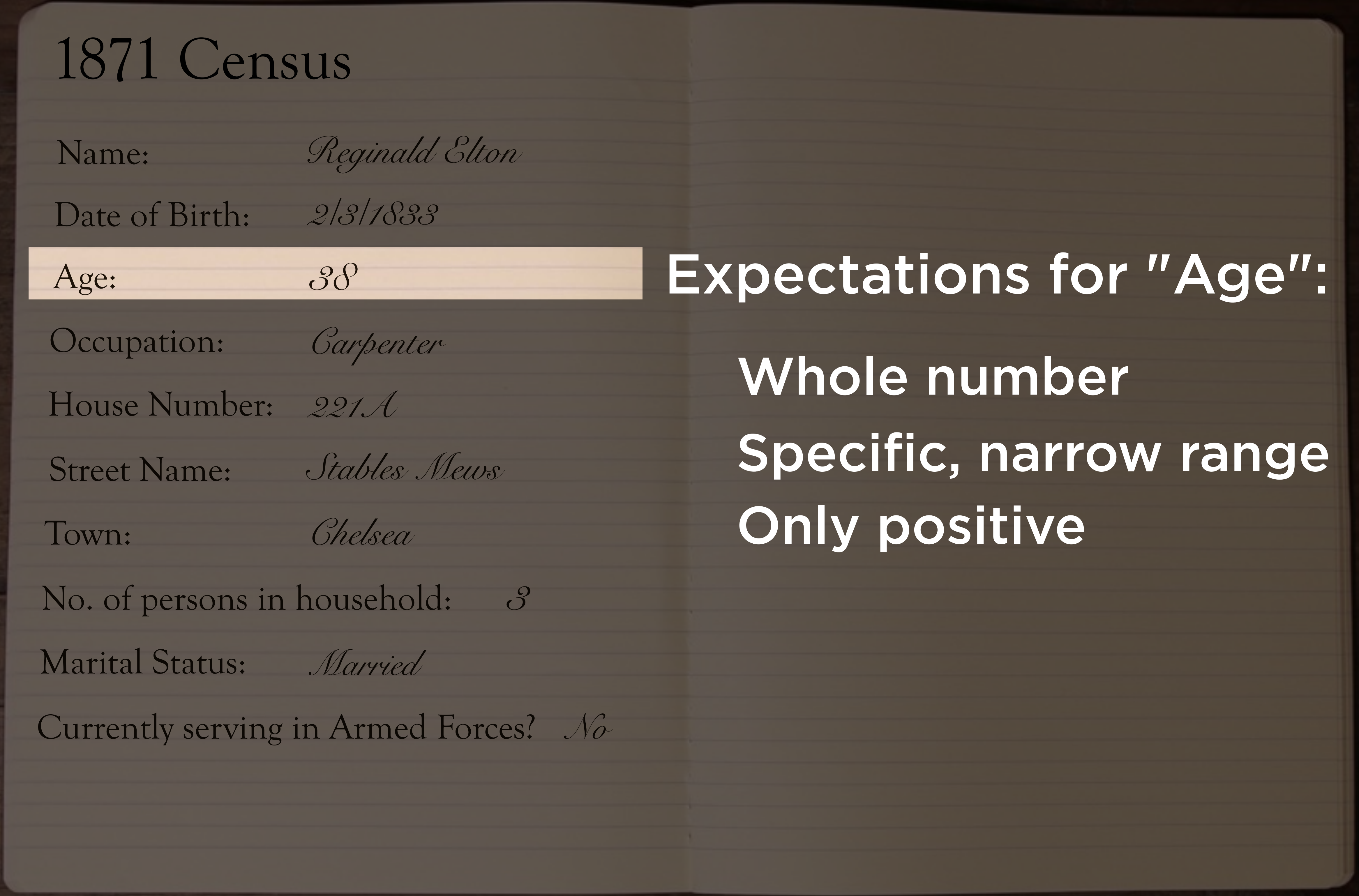


1871 Census

Name:	*Reginald Elton*
Date of Birth:	*2/3/1833*
Age:	*38*
Occupation:	*Carpenter*
House Number:	*221A*
Street Name:	*Stables Mews*
Town:	*Chelsea*
No. of persons in household:	*3*
Marital Status:	*Married*
Currently serving in Armed Forces?	*No*

Expectations for "Age":

- Whole number
- Specific, narrow range

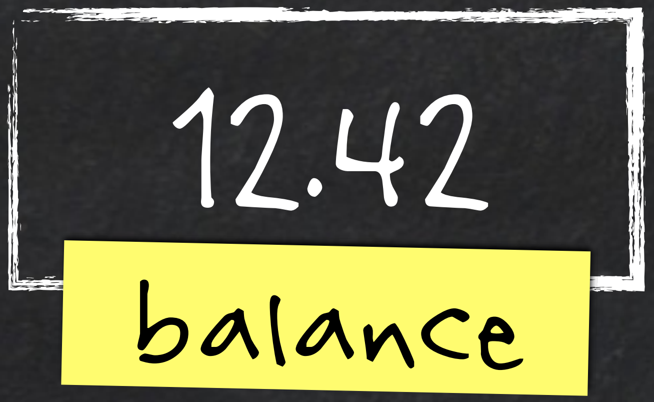


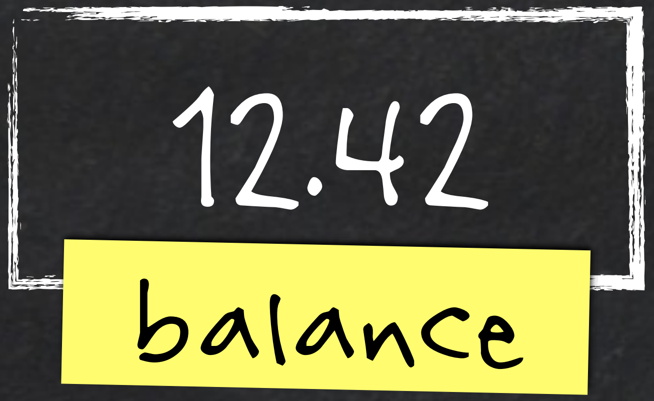
1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No

Expectations for "Age":

- Whole number
- Specific, narrow range
- Only positive


12.42
balance



Expectations for a Bank Balance:

12.42
balance

Expectations for a Bank Balance:

Can be positive or negative

12.42
balance

Expectations for a Bank Balance:

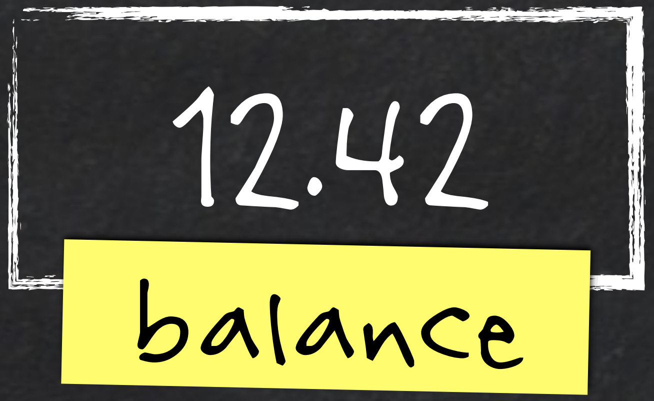
Can be positive or negative

Fractional values

12.42
balance

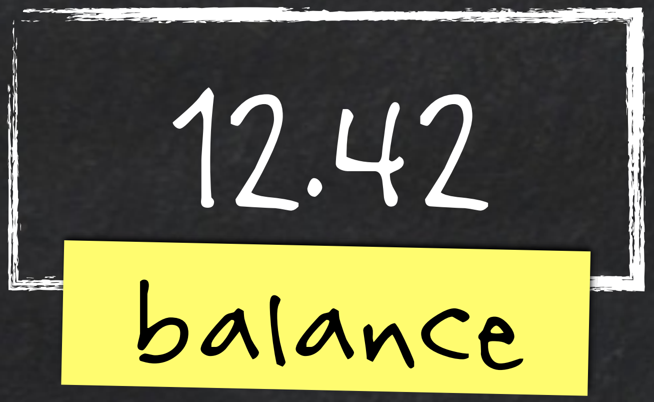
Expectations for a Bank Balance:

- Can be positive or negative
- Fractional values
- Wider (but still limited) range



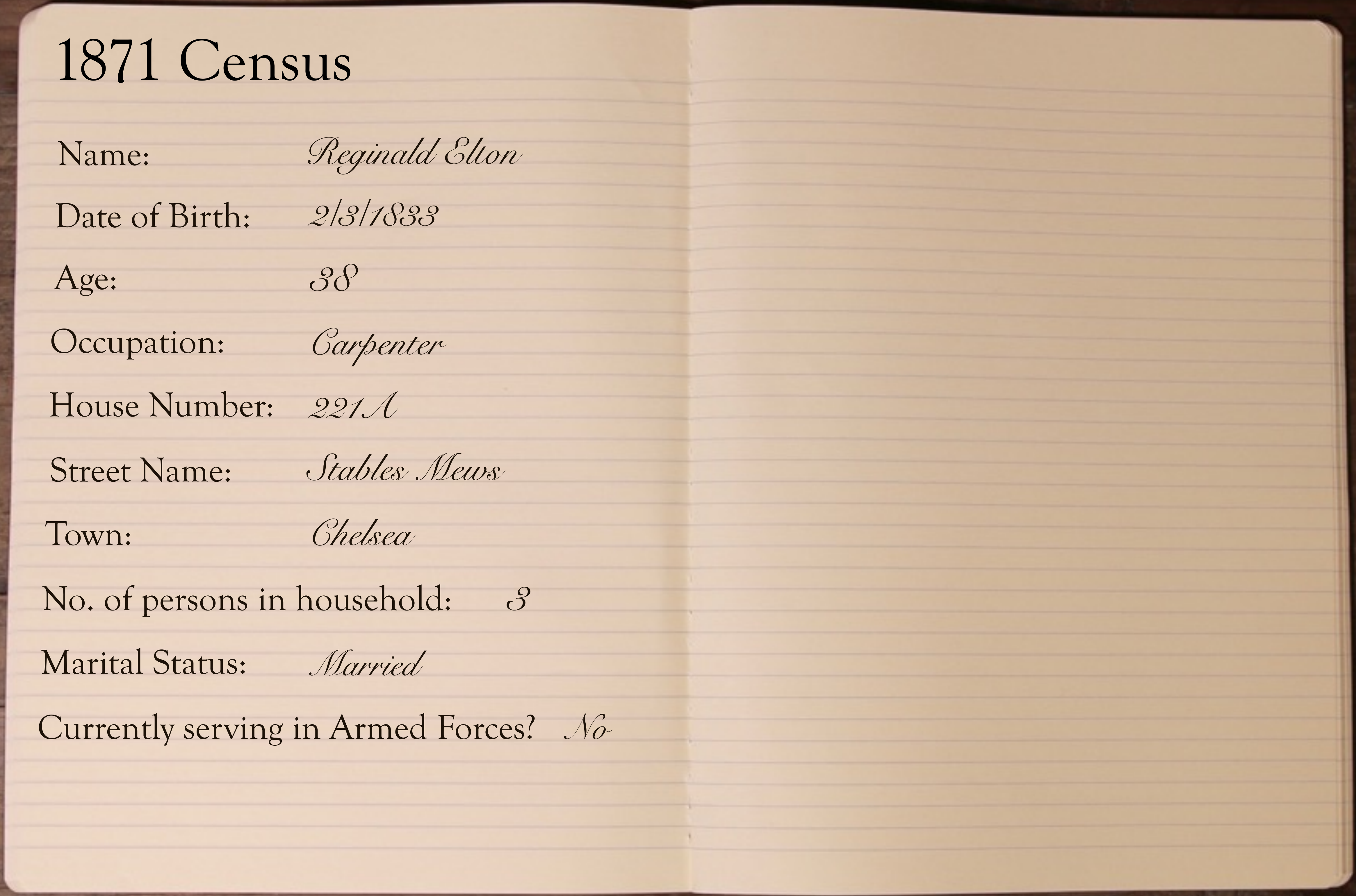
Expectations for a Bank Balance:

- Can be positive or negative
- Fractional values
- Wider (but still limited) range
- Volatile - changes often



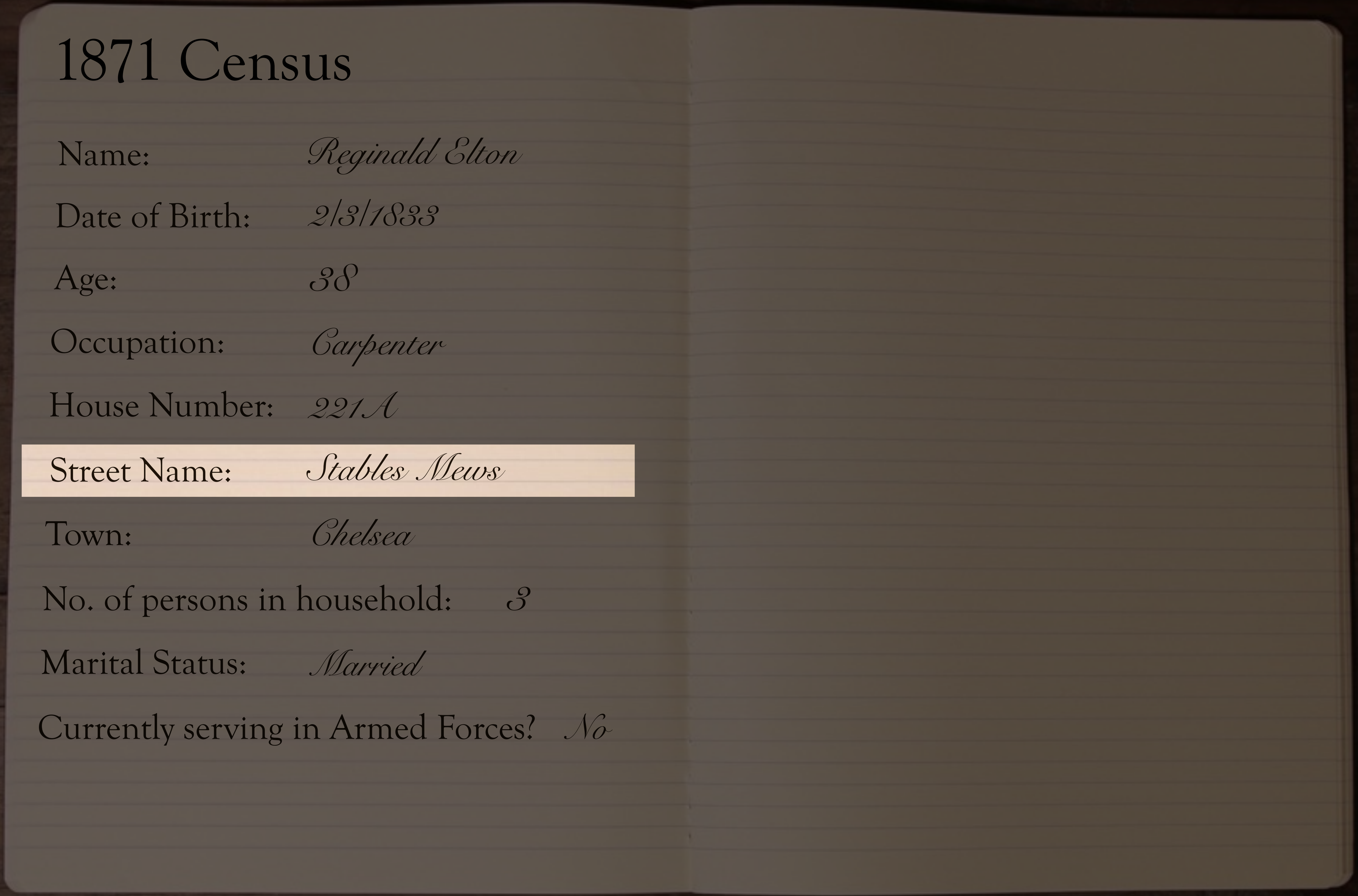
Expectations for a Bank Balance:

- Can be positive or negative
- Fractional values
- Wider (but still limited) range
- Volatile - changes often
- Expected operations: add, subtract



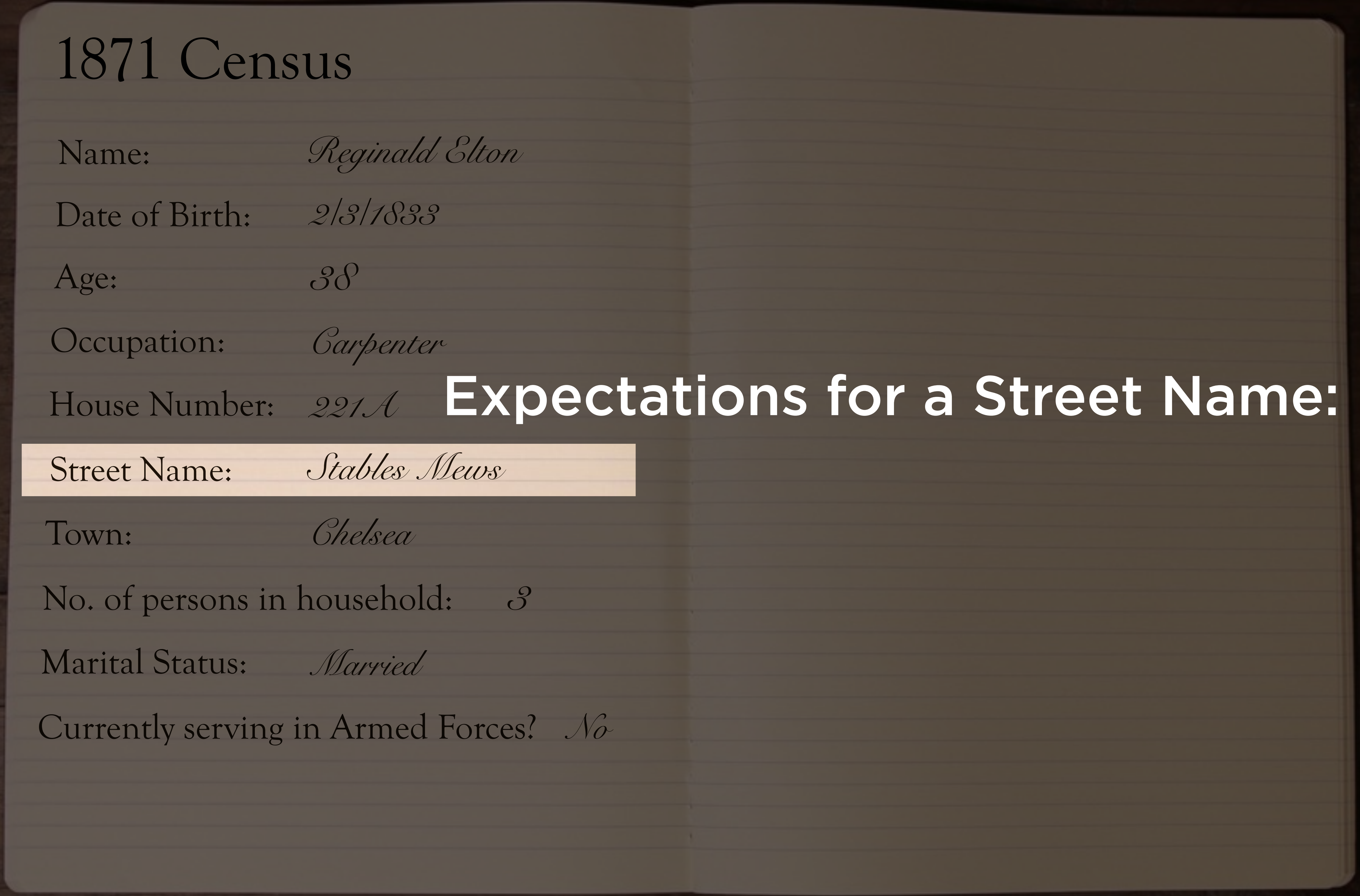
1871 Census

Name:	*Reginald Elton*
Date of Birth:	*2/3/1833*
Age:	*38*
Occupation:	*Carpenter*
House Number:	*221A*
Street Name:	*Stables Mews*
Town:	*Chelsea*
No. of persons in household:	*3*
Marital Status:	*Married*
Currently serving in Armed Forces?	*No*



1871 Census

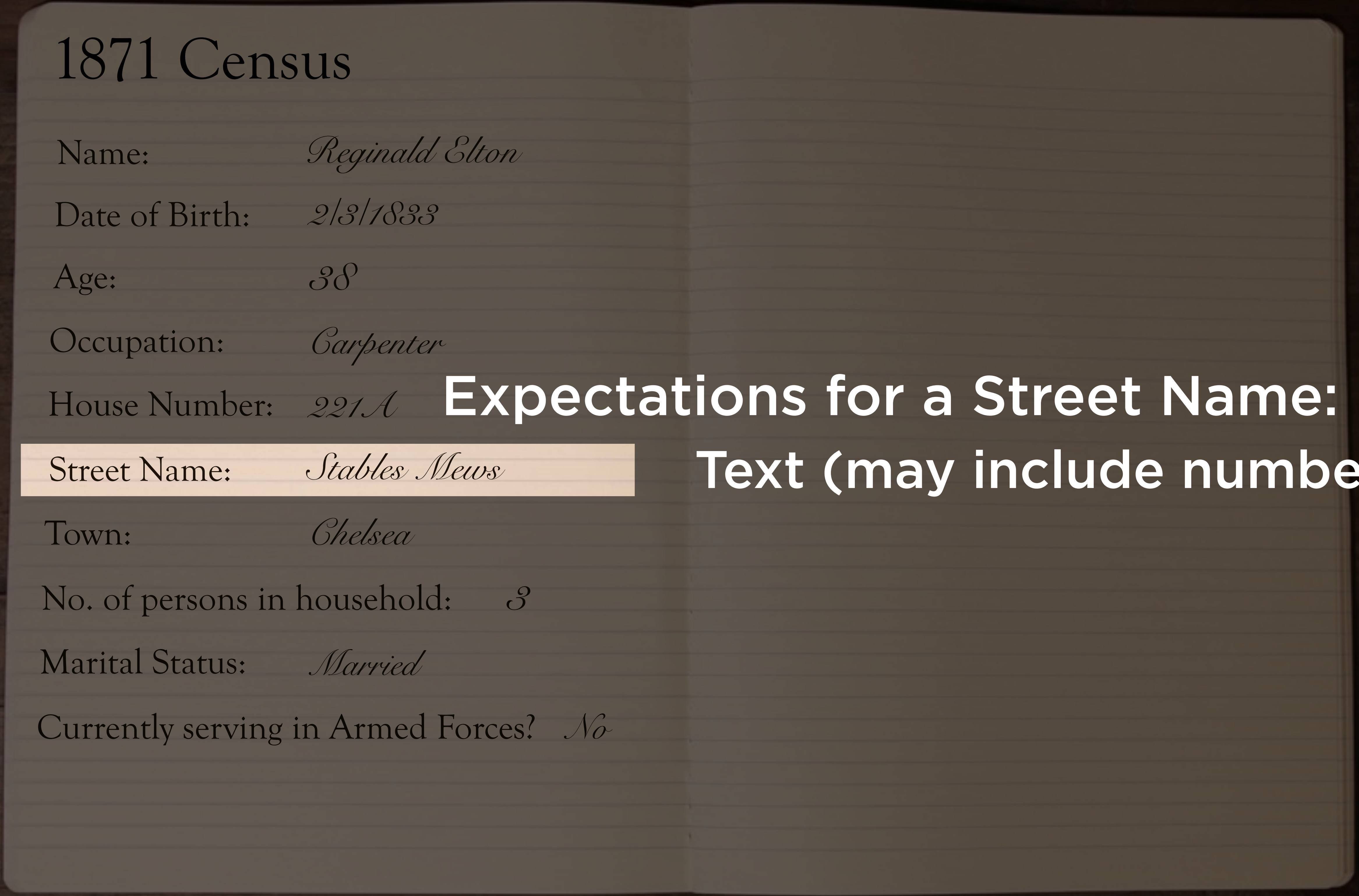
Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No



1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No

Expectations for a Street Name:

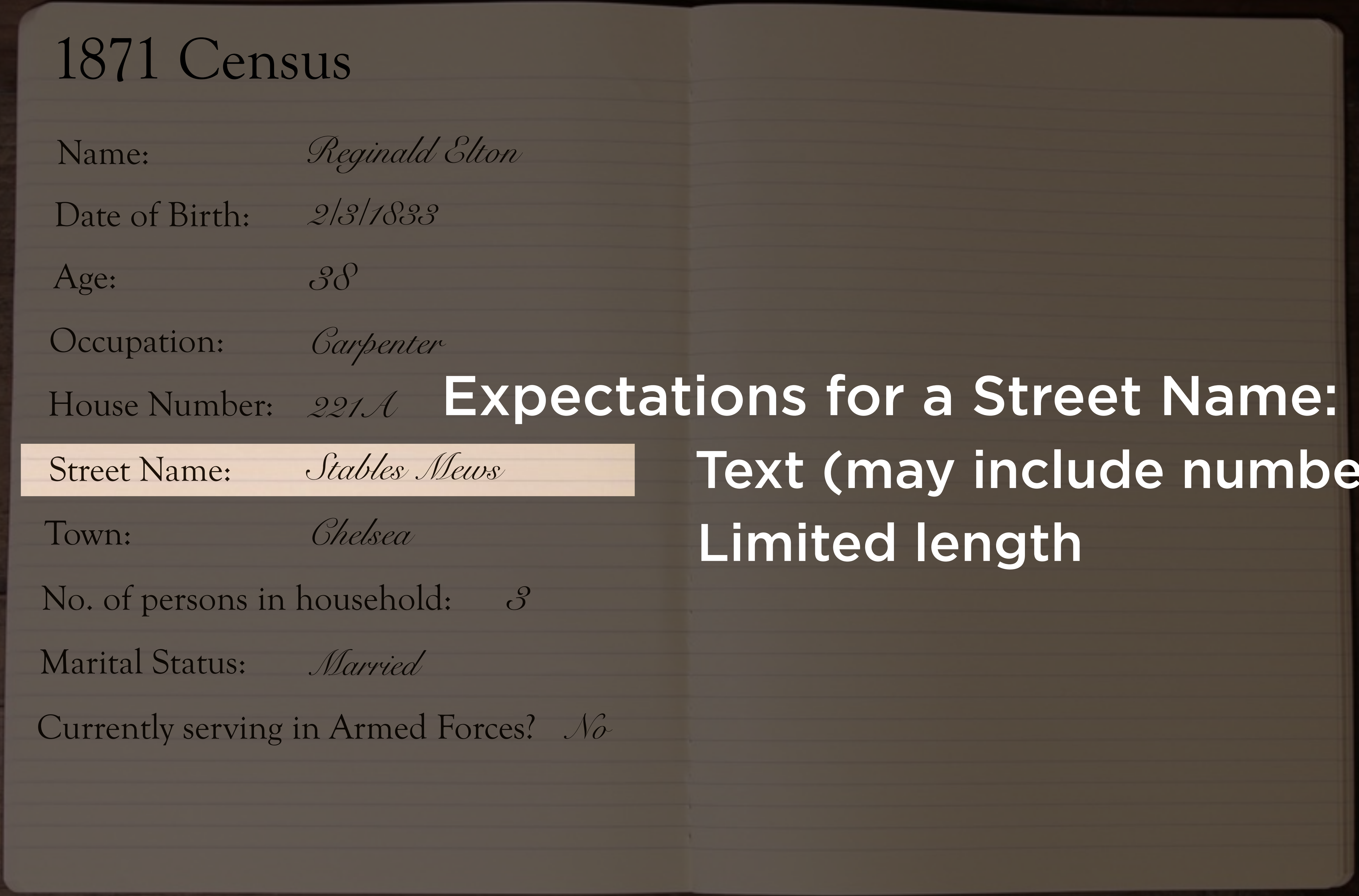


1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No

Expectations for a Street Name:

Text (may include numbers)

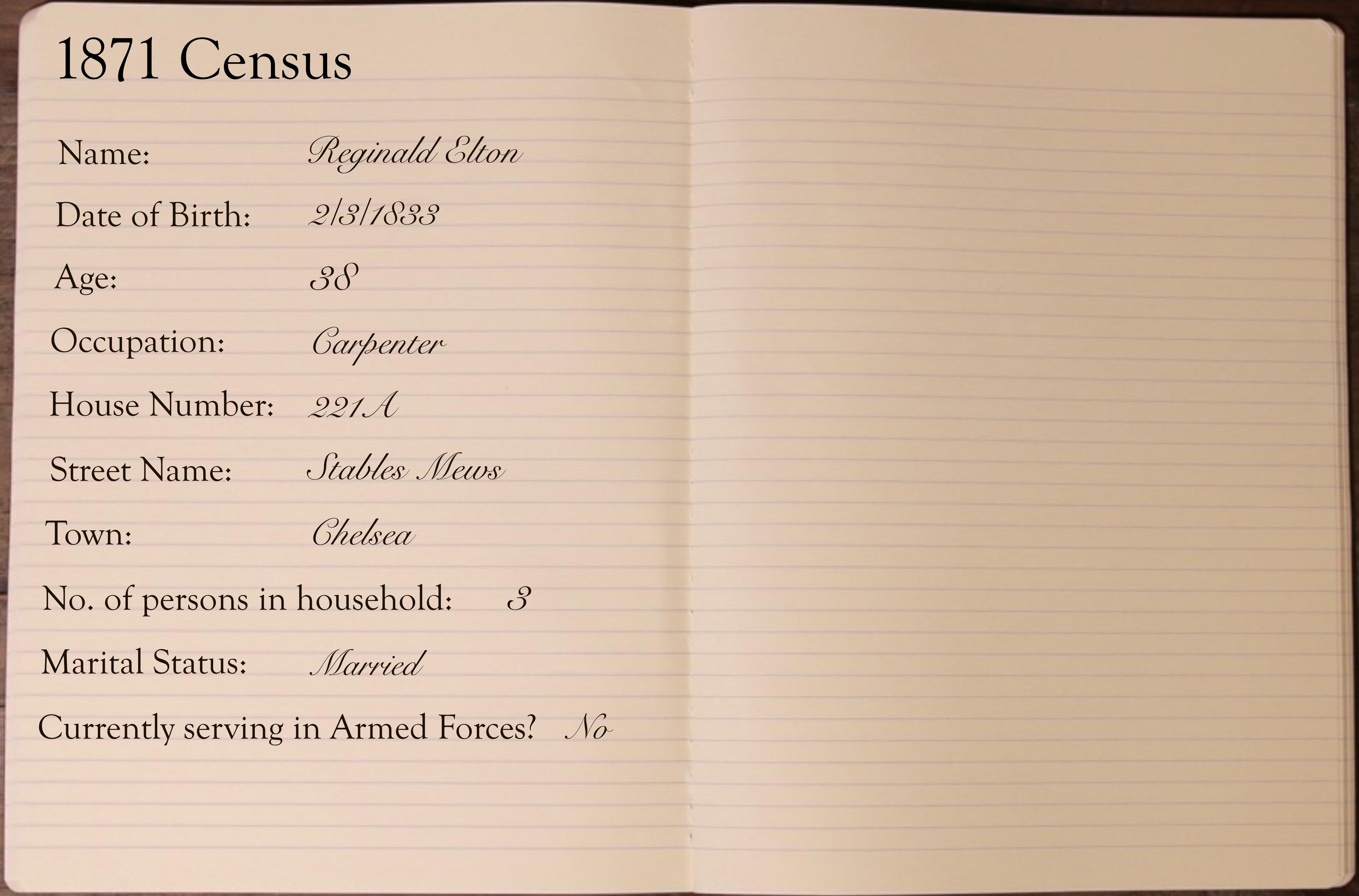


1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No

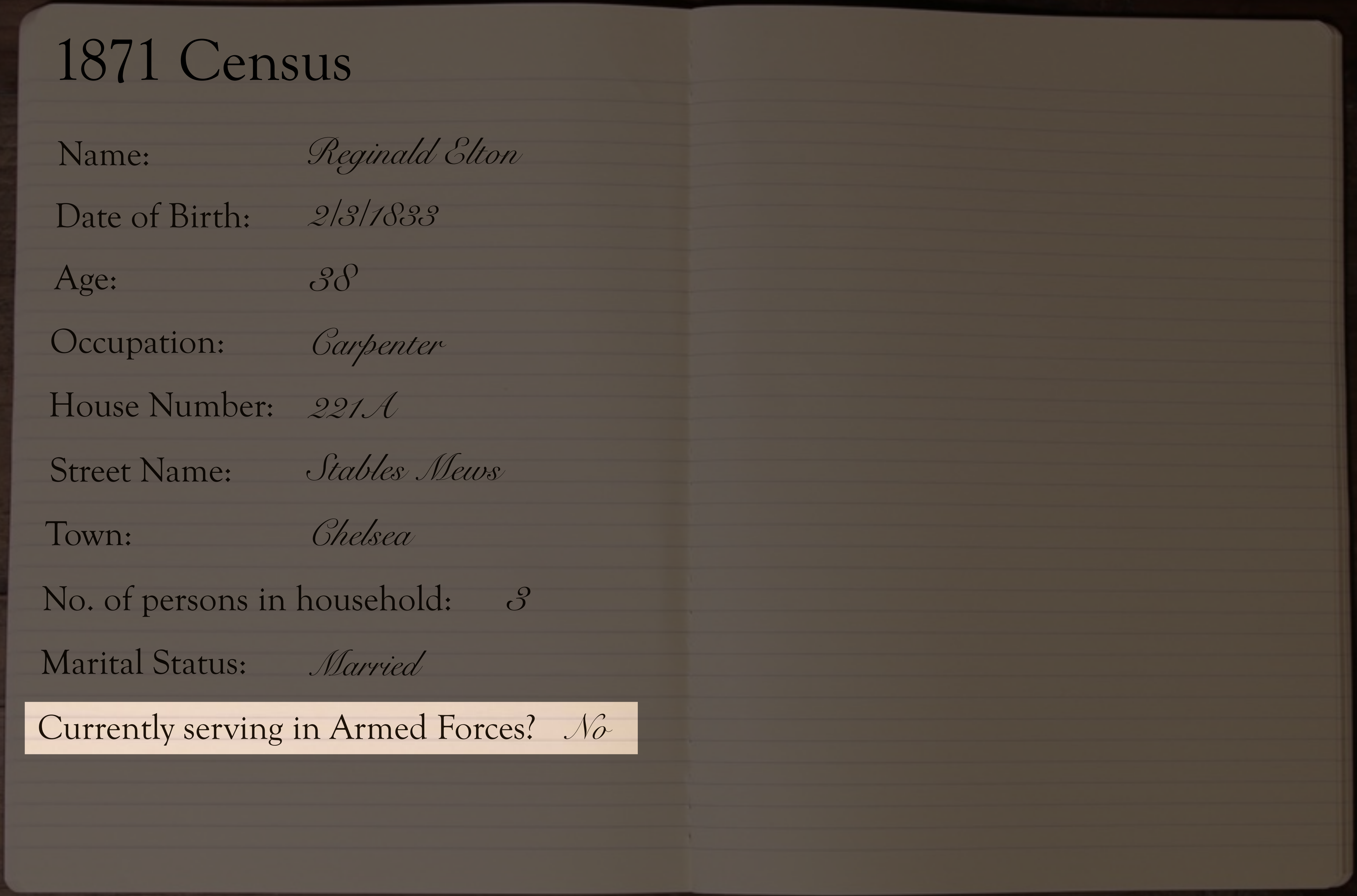
Expectations for a Street Name:

- Text (may include numbers)
- Limited length



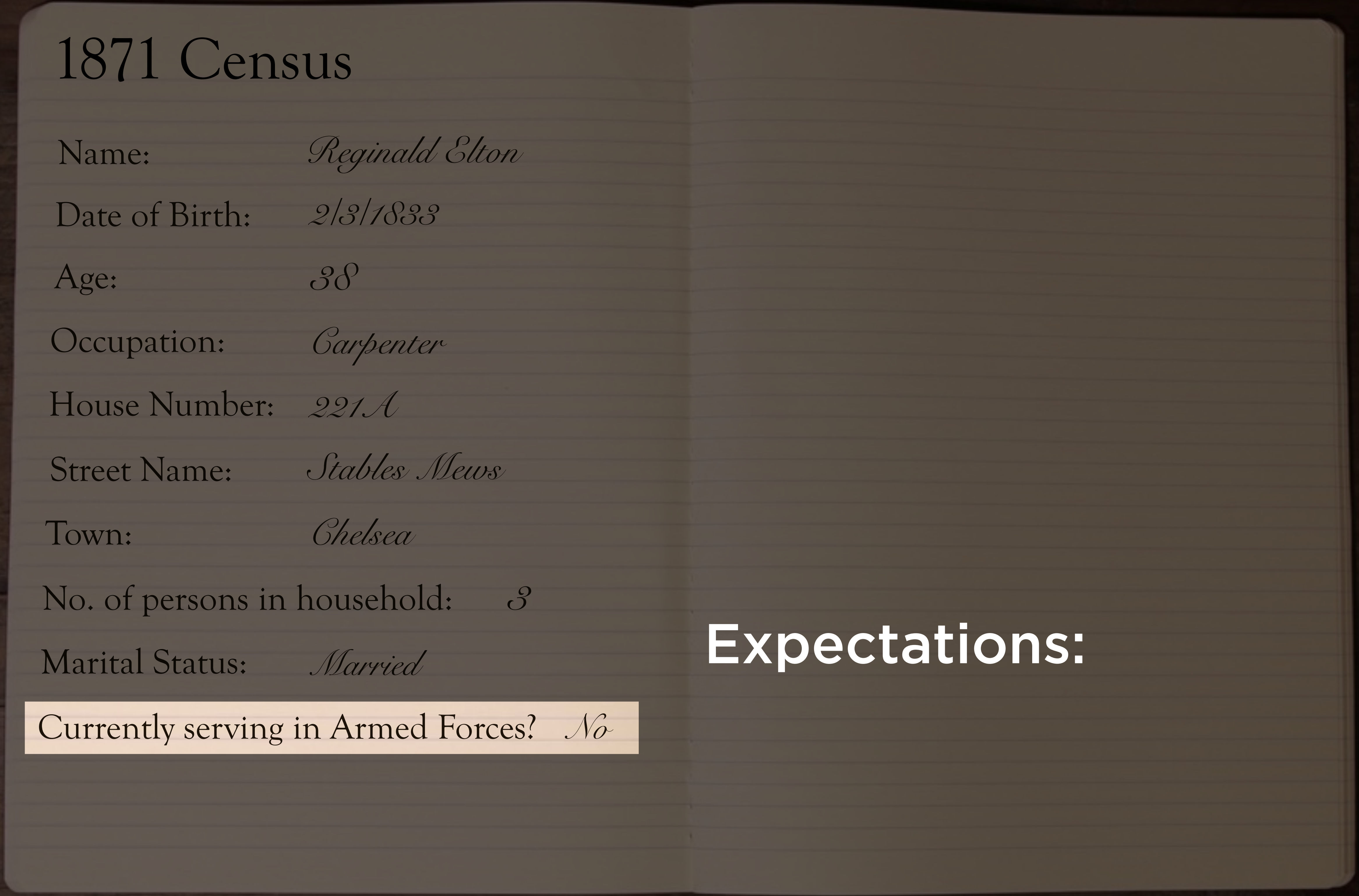
1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No



1871 Census

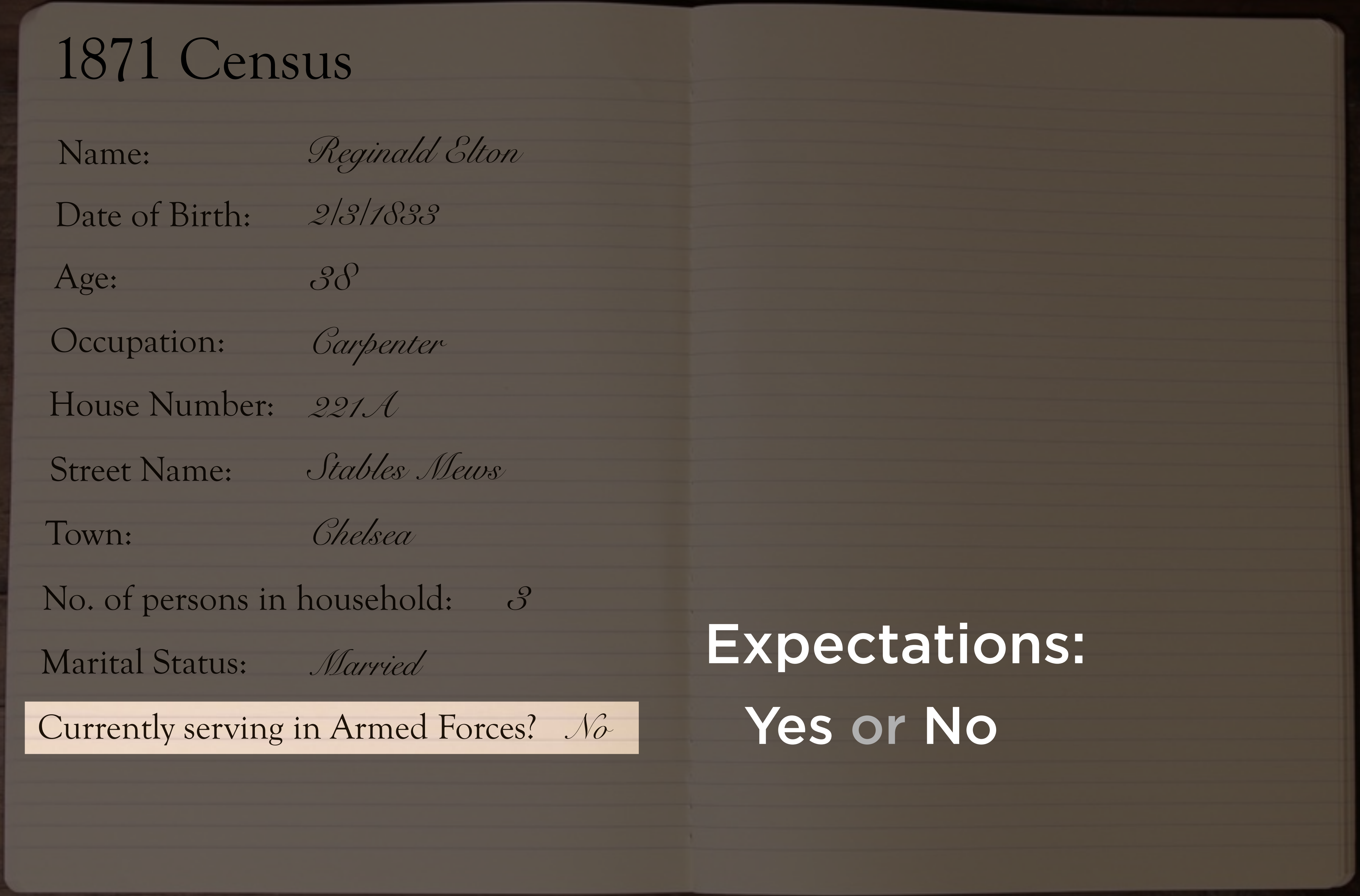
Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No



1871 Census

Name:	*Reginald Elton*
Date of Birth:	*2/3/1833*
Age:	*38*
Occupation:	*Carpenter*
House Number:	*221A*
Street Name:	*Stables Mews*
Town:	*Chelsea*
No. of persons in household:	*3*
Marital Status:	*Married*
Currently serving in Armed Forces?	*No*

Expectations:

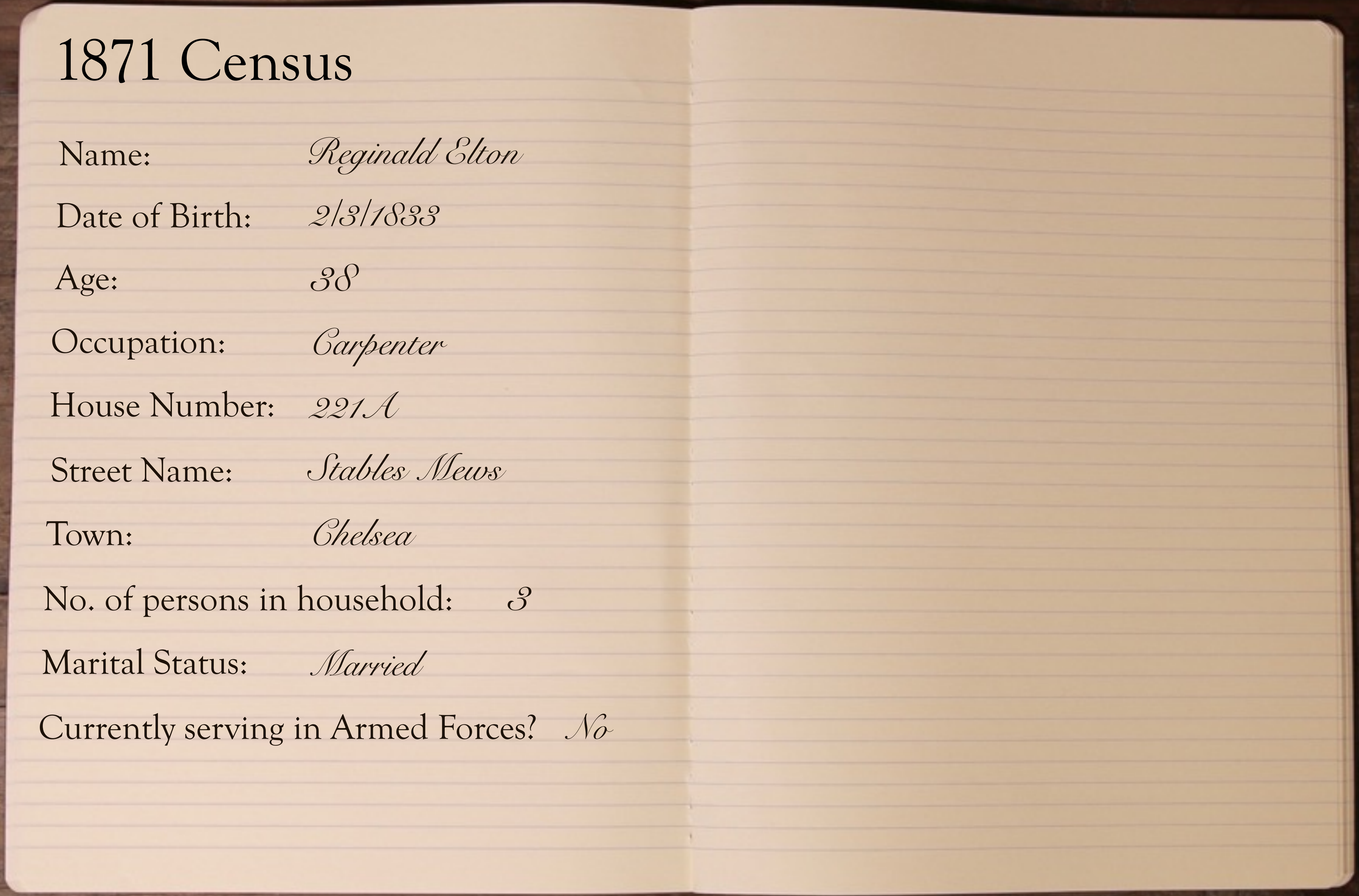


1871 Census

Name:	*Reginald Elton*
Date of Birth:	*2/3/1833*
Age:	*38*
Occupation:	*Carpenter*
House Number:	*221A*
Street Name:	*Stables Mews*
Town:	*Chelsea*
No. of persons in household:	*3*
Marital Status:	*Married*
Currently serving in Armed Forces?	*No*

Expectations:

Yes or **No**



1871 Census

Name:	Reginald Elton
Date of Birth:	2/3/1833
Age:	38
Occupation:	Carpenter
House Number:	221A
Street Name:	Stables Mews
Town:	Chelsea
No. of persons in household:	3
Marital Status:	Married
Currently serving in Armed Forces?	No

Data: Applying Data Types

Different Levels of Detail

Different Levels of Detail

C++, Swift, Java, Boo

Create a variable called "score".

Different Levels of Detail

C++, Swift, Java, Boo

Create a variable called "score".
It's a number.

Different Levels of Detail

C++, Swift, Java, Boo

Create a variable called "score".

It's an integer.

Different Levels of Detail

C++, Swift, Java, Boo

Create a variable called "score".

It's a positive integer.

Different Levels of Detail

C++, Swift, Java, Boo

Create a variable called "score".

It's a positive integer between 0 and 65,535.

Different Levels of Detail

C++, Swift, Java, Boo

Create a variable called "score".

It's a positive integer between 0 and 65,535.

Python, JavaScript

Create a variable called "score".

Different Levels of Detail

C++, Swift, Java, Boo

Create a variable called "score".

It's a positive integer between 0 and 65,535.

Python, JavaScript

Create a variable called "score".

We'll decide what type it is later.

Numeric Variables

Numeric Variables

```
// integers
int age = 21;
int pages = 542;
int numberOfEmployees = 1204;
int speedLimit = 45;
int bestsellerListPosition = 1;
int numberOfFloors = 20;
```

Numeric Variables

```
// integers
int age = 21;
int pages = 542;
int numberOfEmployees = 1204;
int speedLimit = 45;
int bestsellerListPosition = 1;
int numberOfFloors = 20;

// floating-point numbers
float temperature = 72.4;
float snailSpeed = 0.029;
```

Numeric Variables

```
// integers
int age = 21;
int pages = 542;
int numberOfEmployees = 1204;
int speedLimit = 45;
int bestsellerListPosition = 1;
int numberOfFloors = 20;

// floating-point numbers
float temperature = 72.4;
float snailSpeed = 0.029;
```

Numeric Variables

```
// integers
int age = 21;
int pages = 542;
int numberOfEmployees = 1204;
int speedLimit = 45;
int bestsellerListPosition = 1;
int numberOfFloors = 20;

// floating-point numbers
float temperature = 72.4;
float snailSpeed = 0.029;
```

C-style keywords controlling positive/negative values:

```
int (positive or negative) / unsigned int (no negative values)
```

C-style keywords controlling integer sizes:

```
int / long int / long long int / short int
```

T-SQL keywords:

```
int / bigint / smallint / tinyint
```

Further Detail

Some languages allow greater control over positive/negative and size of values

Boolean Values

Boolean Values

```
// Java
boolean isUserLoggedIn = true;

// Swift
var currentlyRecording: Bool = true

// Python
onActiveDuty = True

// JavaScript
var hasSpaceshipCrashed = false;
```

Boolean Values

```
// Java
boolean isUserLoggedIn = true;

// Swift
var currentlyRecording: Bool = true

// Python
onActiveDuty = True

// JavaScript
var hasSpaceshipCrashed = false;
```

Boolean Values

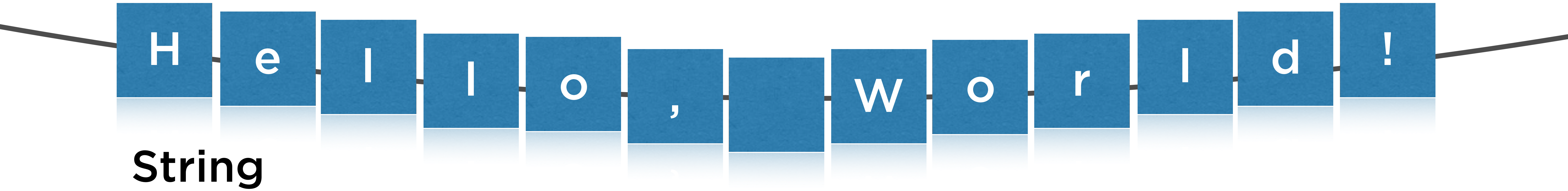
```
// Java
boolean isUserLoggedIn = true;

// Swift
var currentlyRecording: Bool = true

// Python
onActiveDuty = True

// JavaScript
var hasSpaceshipCrashed = false;
```

```
// single character data type
char singleLetter = = 'A'
```

Text / Character Data

Letters, Words, Sentences, Paragraphs - and more

```
// single character data type
char singleLetter = = 'A'
```



String

Text / Character Data

Letters, Words, Sentences, Paragraphs - and more

String Values

String Values

```
// C#
string message = "Thanks for Playing!";

// Swift
var message: String = "Thanks for Playing!"

// Python
message = "Thanks for Playing!"

// JavaScript
var message = "Thanks for Playing!";
```

Literal Values

```
myInteger = 99

myFloat   = 542.5

myBoolean = true

myString  = "This is a message!"
```

Literal Values

```
myInteger = 99

myFloat   = 542.5

myBoolean = true

myString  = "This is a message!"
```

Literal Values

```
myInteger = 99

myFloat   = 542.5

myBoolean = true

myString  = "This is a message!"
```

Literal Values

```
myInteger = $\boxed{99}$
myFloat   = $\boxed{542.5}$
myBoolean = $\boxed{true}$
myString  = $\boxed{"This is a message!"}$
```

Literal Values

```
myInteger = 99      // integer literal

myFloat   = 542.5

myBoolean = true

myString  = "This is a message!"
```

Literal Values

```
myInteger = 99     // integer literal

myFloat   = 542.5  // float literal - sometimes 542.5f

myBoolean = true

myString  = "This is a message!"
```

Literal Values

```
myInteger = 99     // integer literal

myFloat   = 542.5  // float literal - sometimes 542.5f

myBoolean = true   // boolean literal

myString  = "This is a message!"
```

Literal Values

```
myInteger = 99      // integer literal

myFloat   = 542.5   // float literal - sometimes 542.5f

myBoolean = true    // boolean literal

myString  = "This is a message!" // string literal
```

Built-In "Primitive" Types

Built-In "Primitive" Types

C++
Numeric Data Types

short, int, long, long long,
unsigned short, unsigned int,
unsigned long, unsigned long long,
float, double, long double (etc)

Built-In "Primitive" Types

C++
Numeric Data Types

short, int, long, long long, unsigned short, unsigned int, unsigned long, unsigned long long, float, double, long double (etc)

JavaScript
Numeric Data Types

Number

Data: Using Constants

Variables / Constants

Variables / Constants

```
// create some messages to use later
string message = "Thanks for Playing!"
string congrats = "Great High Score!"
string someError = "No connection detected."
```

Variables / Constants

```
// create some messages to use later
string message = "Thanks for Playing!"
string congrats = "Great High Score!"
string someError = "No connection detected."

// useful numbers
float pi = 3.14159
int maximumPlayers = 12
```

Constants and Variables

Constants and Variables

1: **Name**

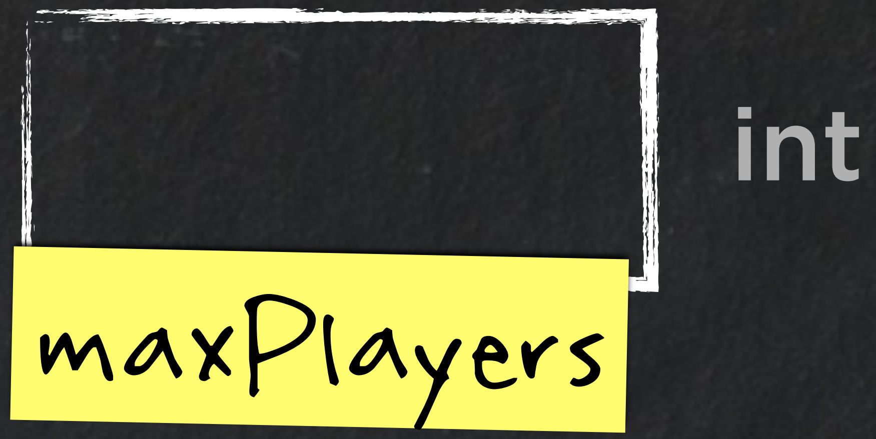
Constants and Variables

1: **Name** 2: **Value**

Constants and Variables

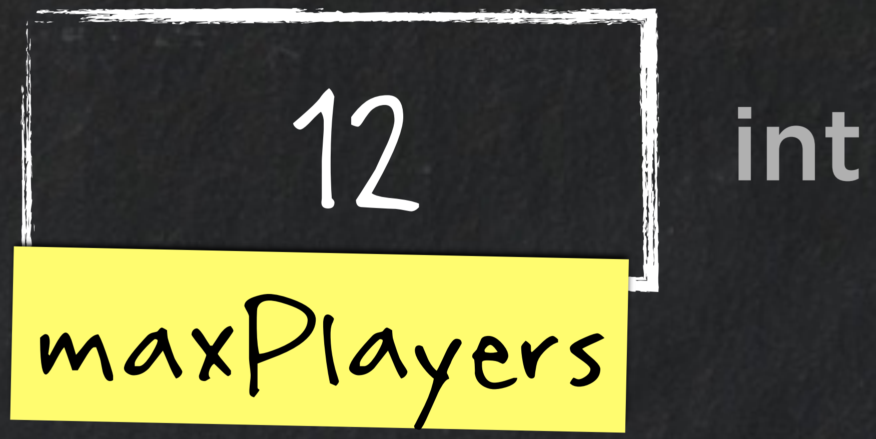
1: **Name** 2: **Value** 3: **Type**

Constants and Variables



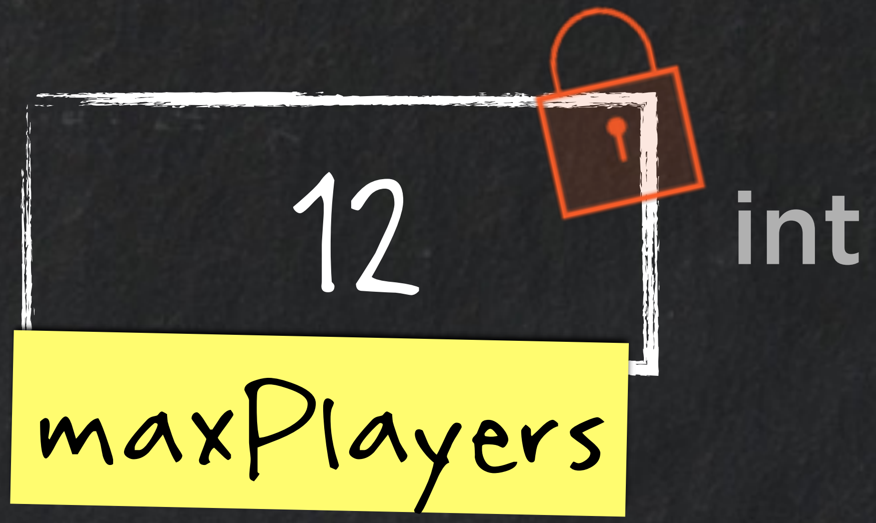
1: **Name** 2: **Value** 3: **Type**

Constants and Variables



1: **Name** 2: **Value** 3: **Type**

Constants and Variables



1: Name 2: Value 3: Type

Creating Constants - Example

Creating Constants - Example

```
// C# requires an additional keyword
```

Creating Constants - Example

```
// C# requires an additional keyword
string message = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"


// Swift requires a different keyword
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"


// Swift requires a different keyword
var message: String = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"

// Swift requires a different keyword
var message: String = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"


// Swift requires a different keyword
let message: String = "Thanks for Playing!"
```

Creating Constants - Example

Creating Constants - Example

```
// C# requires an additional keyword
```

Creating Constants - Example

```
// C# requires an additional keyword
string message = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"


// Swift requires a different keyword
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"


// Swift requires a different keyword
var message: String = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"

// Swift requires a different keyword
var message: String = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"
```

```
// Swift requires a different keyword
let message: String = "Thanks for Playing!"
```

Creating Constants - Example

```
// C# requires an additional keyword
const string message = "Thanks for Playing!"


// Swift requires a different keyword
let message: String = "Thanks for Playing!"


// In some languages, normal to see constant names in ALL_CAPS
const int MAXIMUM_PLAYERS = 12
```

Data: Understanding Language Differences

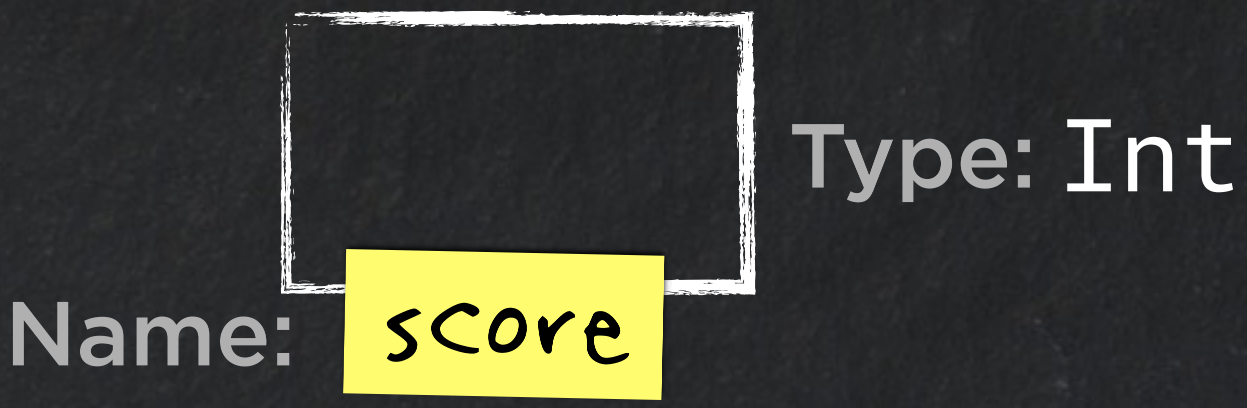
Declaring Variables with Type Information

Declaring Variables with Type Information

```
var score: Int
```

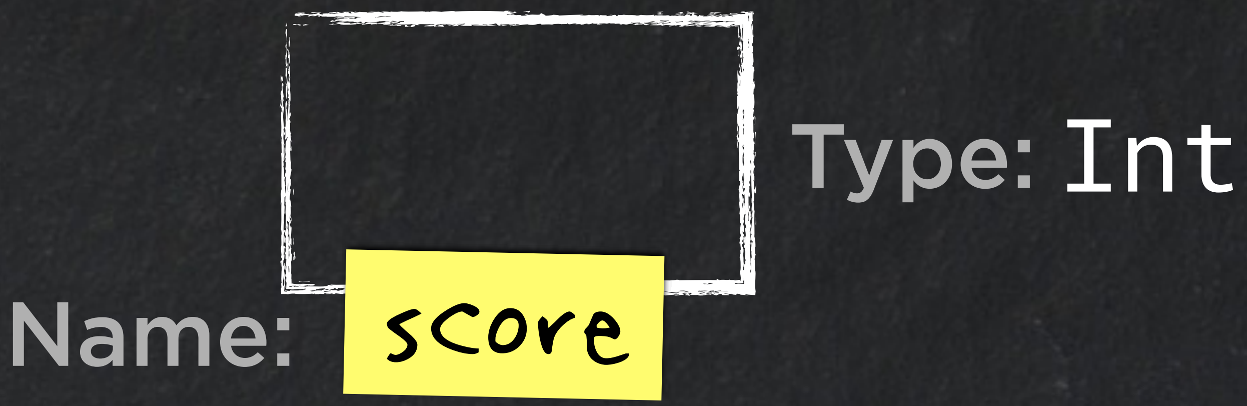
Declaring Variables with Type Information

```
var score: Int
```



Declaring Variables with Type Information

```
var score: Int
```



"Dynamically typed"

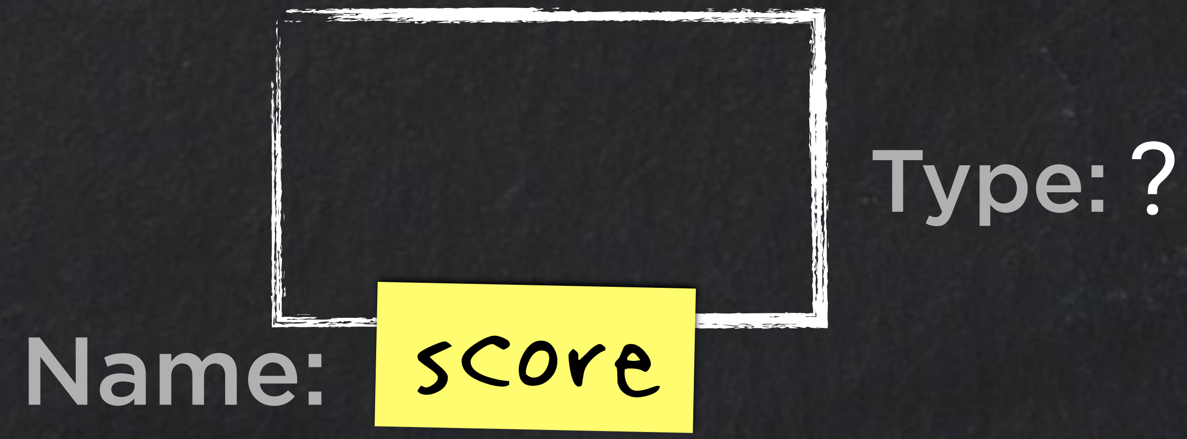
Declaring Variables without Type Information

Declaring Variables without Type Information

```
var score
```

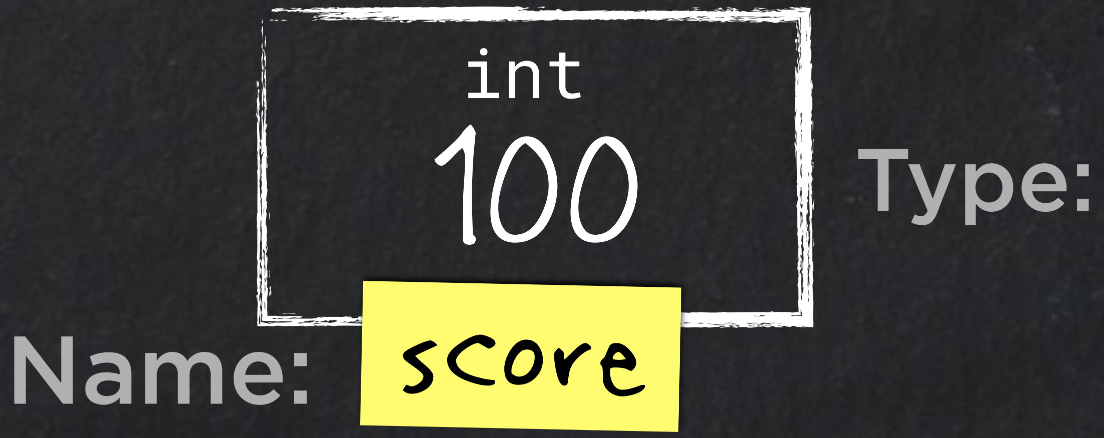
Declaring Variables without Type Information

```
var score
```



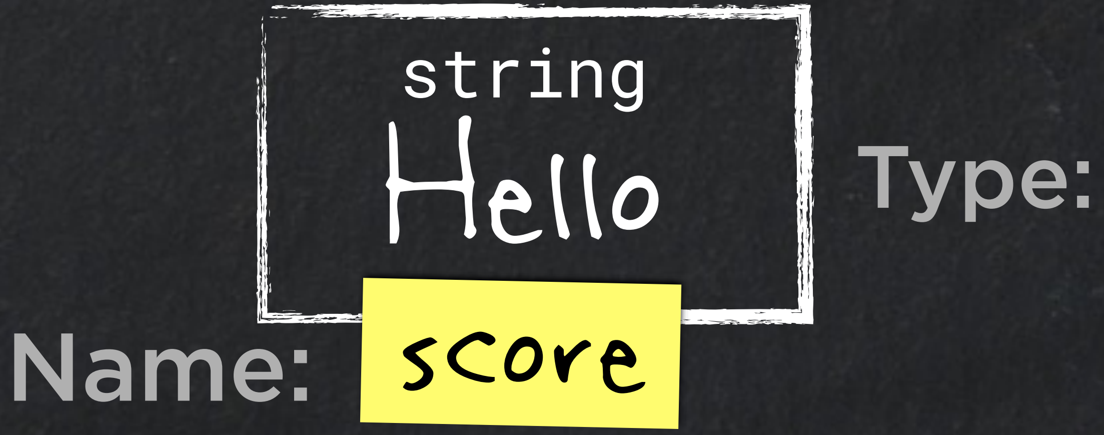
Declaring Variables without Type Information

```
var score
score = 100
```



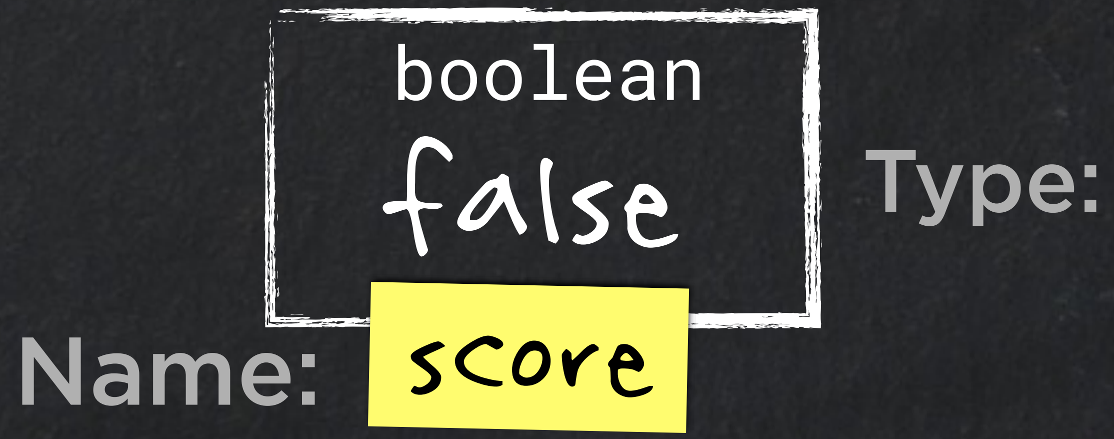
Declaring Variables without Type Information

```
var score
score = 100
score = "Hello"
```



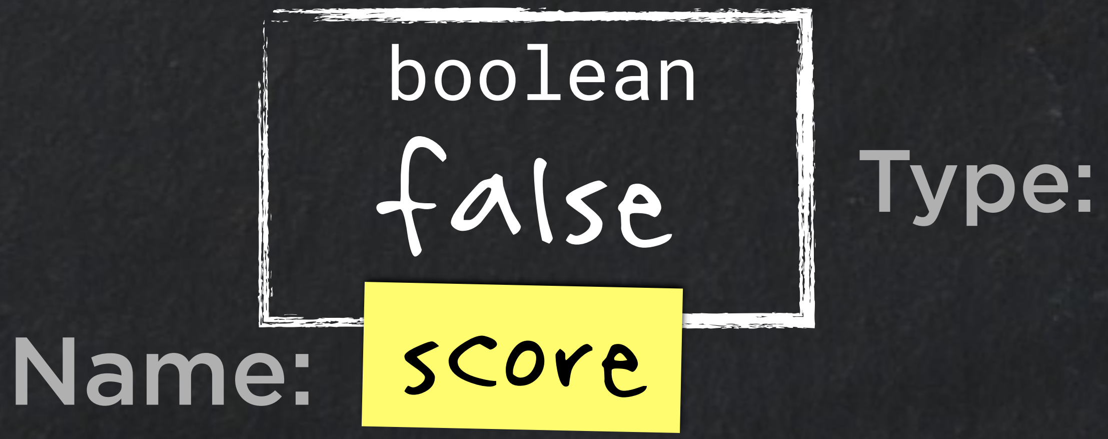
Declaring Variables without Type Information

```
var score
score = 100
score = "Hello"
score = false
```



Declaring Variables without Type Information

```
var score
score = 100
score = "Hello"
score = false
```



"Dynamically typed"

3 Important Things About Variables

3 Important Things About Variables

1: **Name**

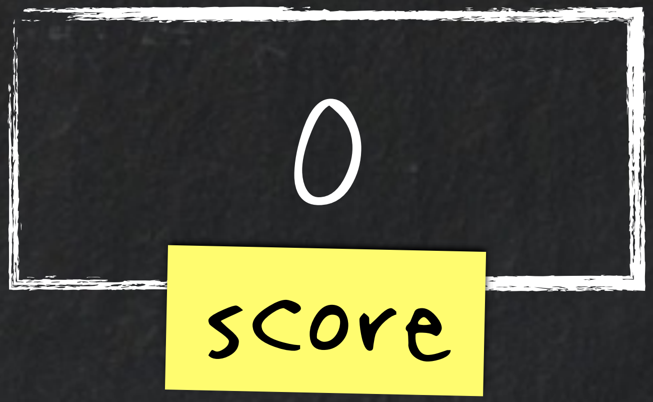
3 Important Things About Variables

1: **Name** 2: **Value**

3 Important Things About Variables

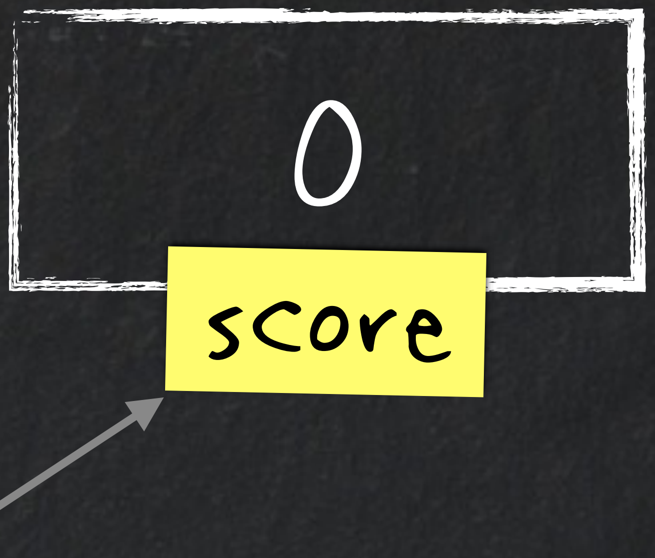
1: **Name** 2: **Value** 3: **Type**

3 Important Things About Variables



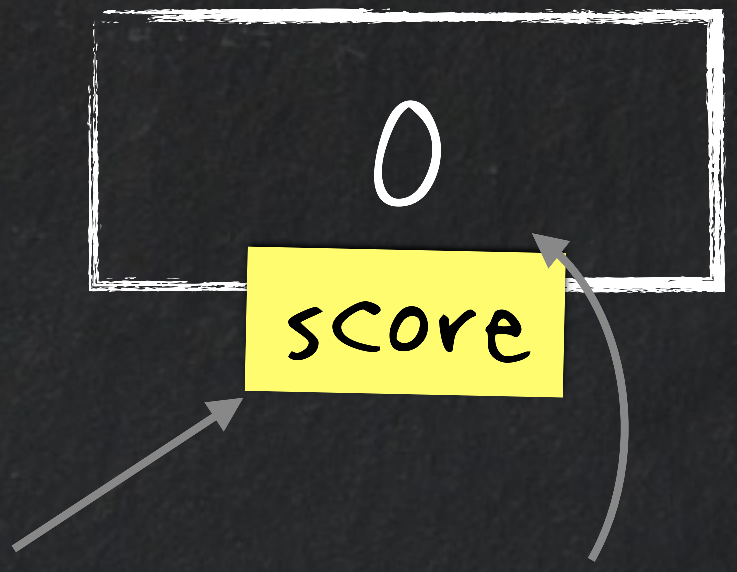
1: **Name** 2: **Value** 3: **Type**

3 Important Things About Variables



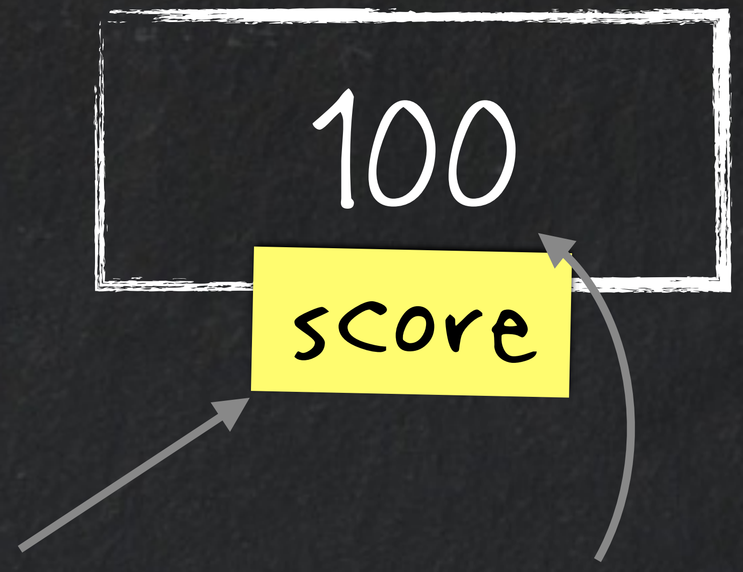
1: Name **2: Value** **3: Type**

3 Important Things About Variables



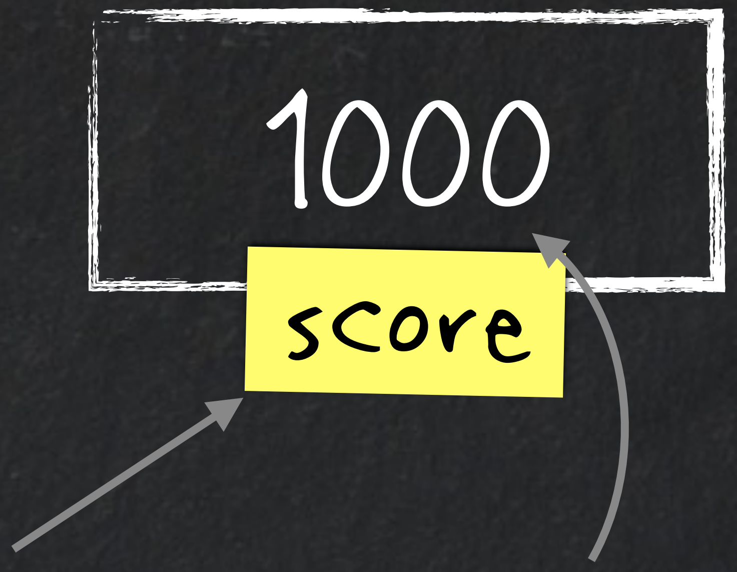
1: Name **2: Value** **3: Type**

3 Important Things About Variables



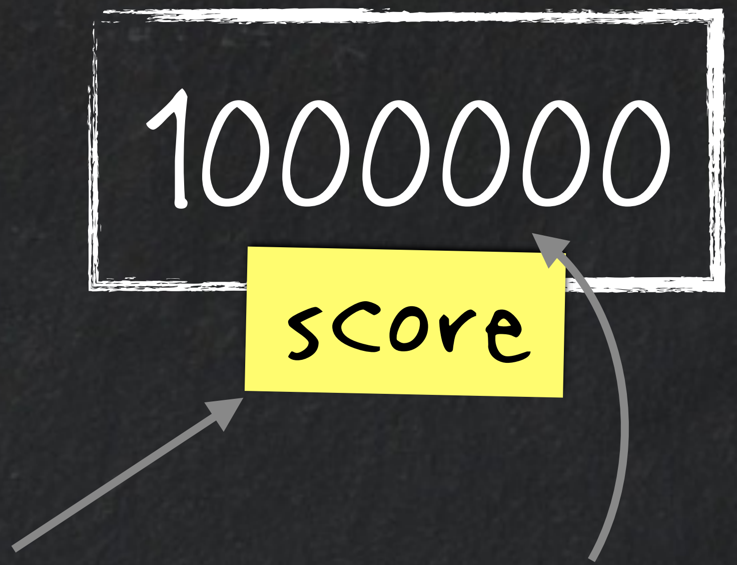
1: Name   2: Value   3: Type

3 Important Things About Variables



1: Name **2: Value** **3: Type**

3 Important Things About Variables



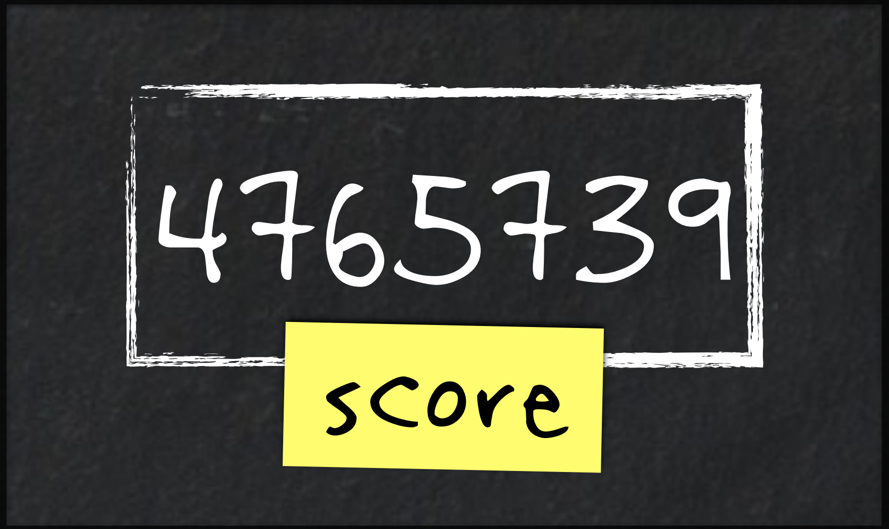
1: Name **2: Value** **3: Type**

3 Important Things About Variables

1000000
score

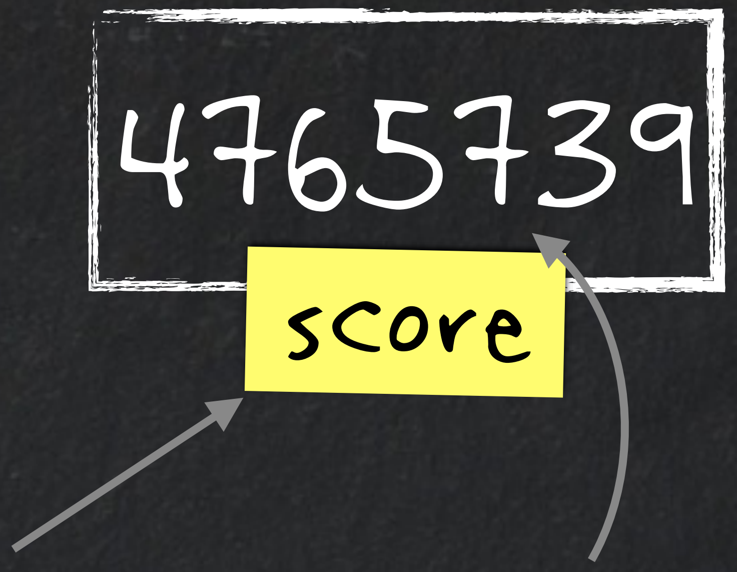
1: Name   2: Value   3: Type

3 Important Things About Variables



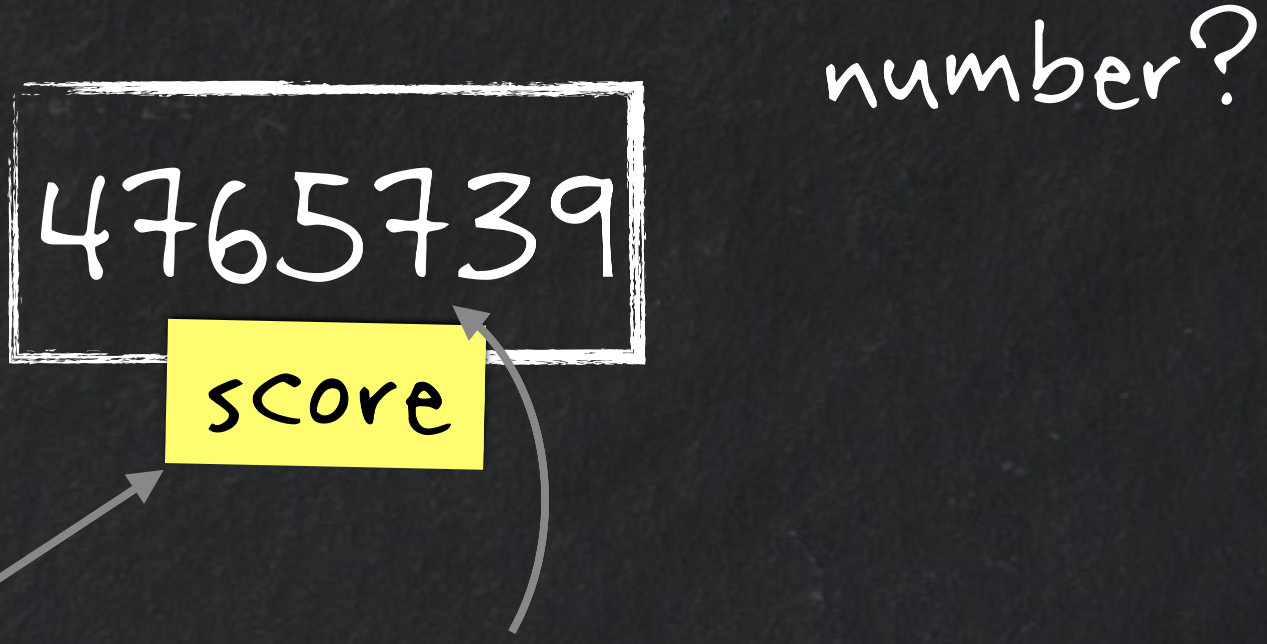
1: Name 2: Value 3: Type

3 Important Things About Variables



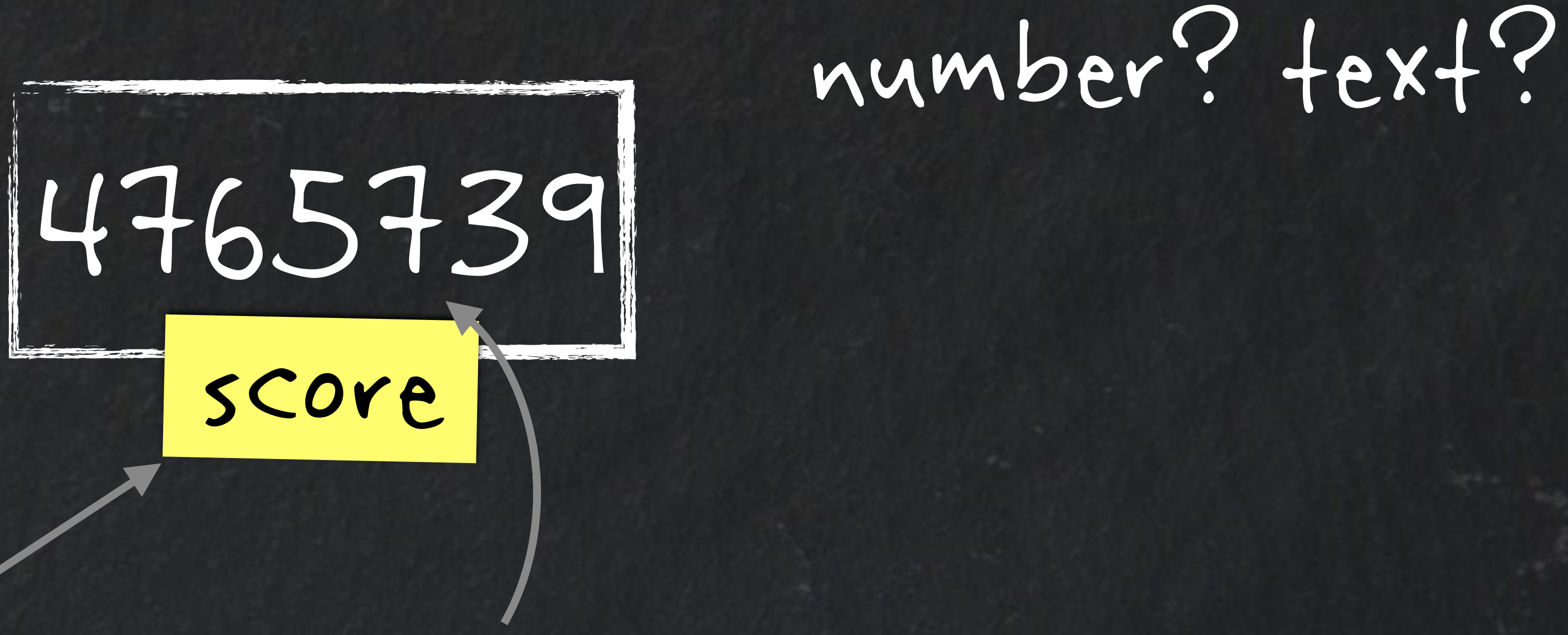
1: Name **2: Value** **3: Type**

3 Important Things About Variables



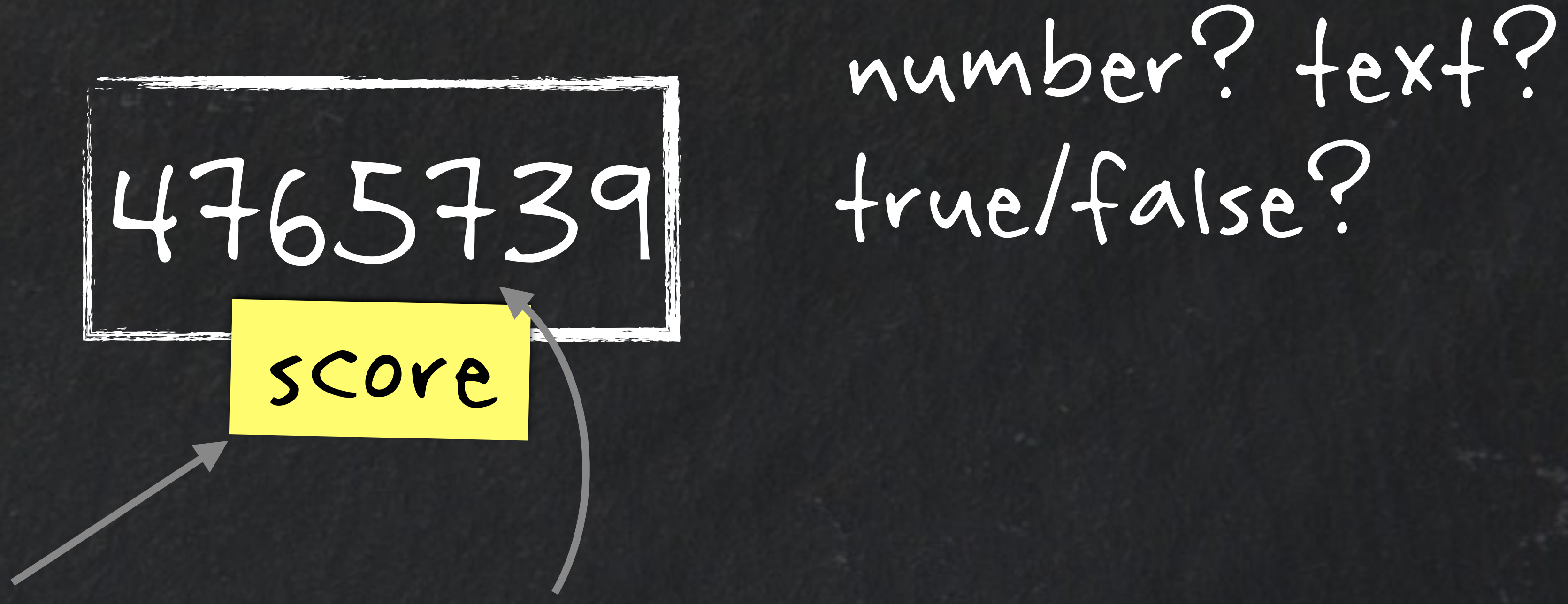
1: Name 2: Value 3: Type

3 Important Things About Variables



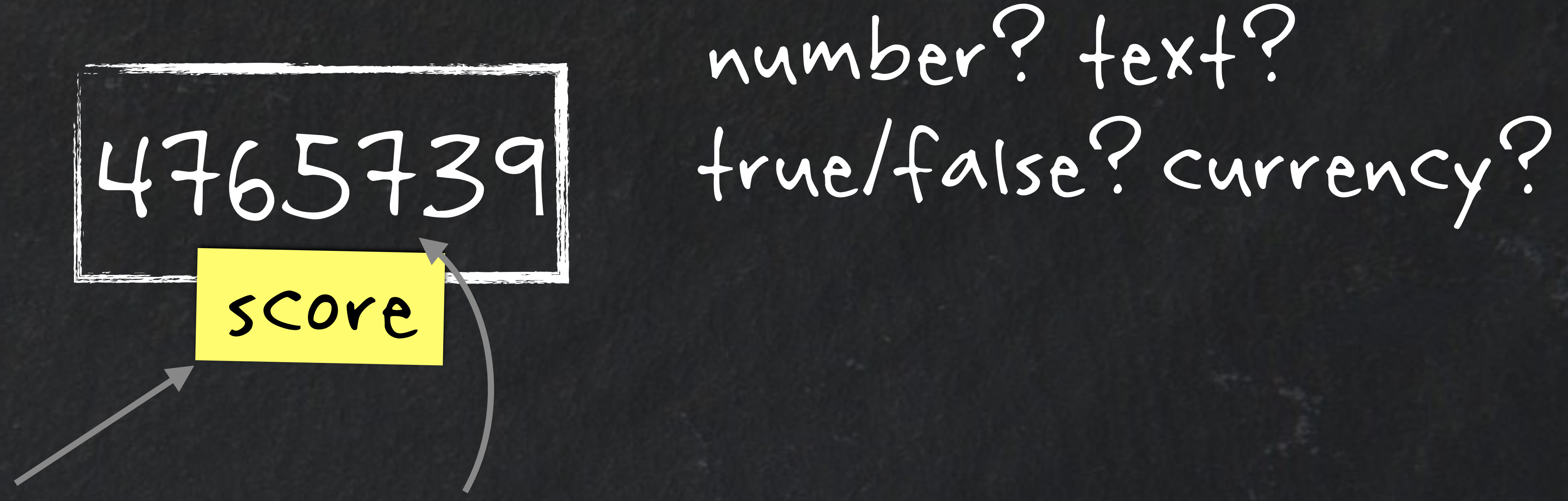
1: Name 2: Value 3: Type

3 Important Things About Variables



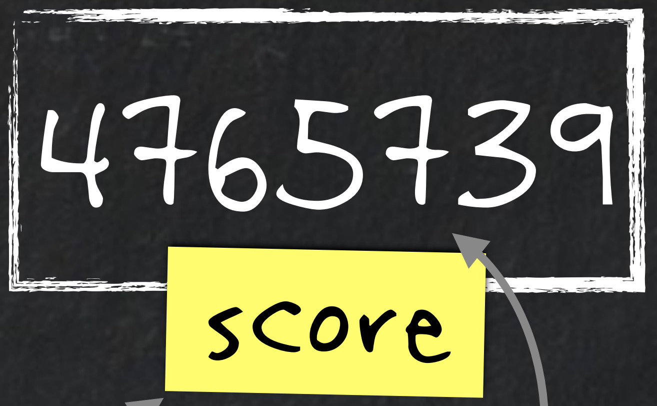
1: Name 2: Value 3: Type

3 Important Things About Variables



1: **Name** 2: **Value** 3: **Type**

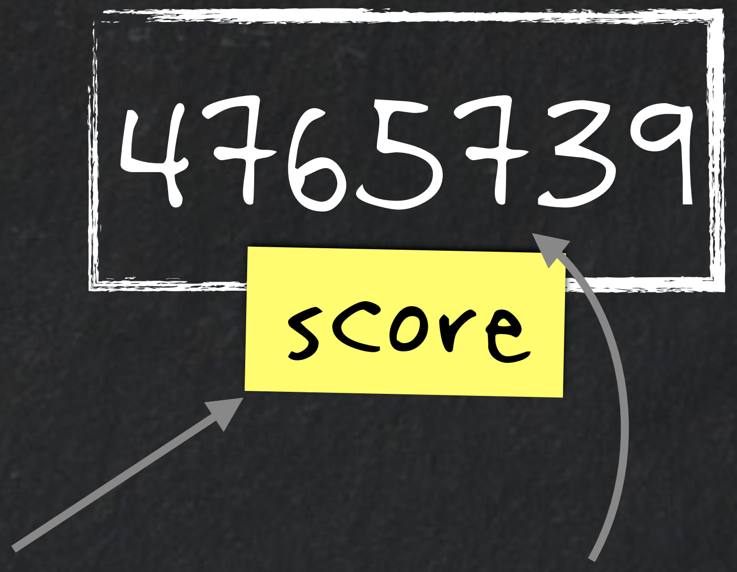
3 Important Things About Variables



number? text?
true/false? currency?
something else?

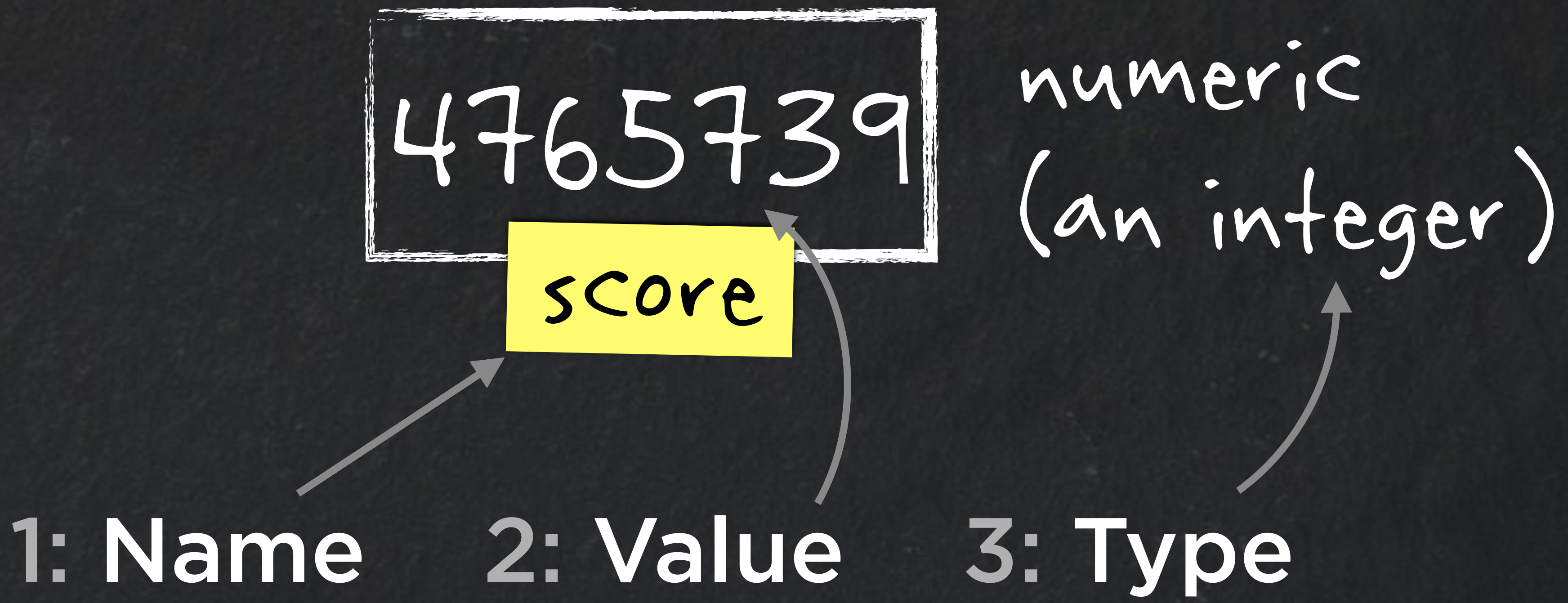
1: Name **2: Value** **3: Type**

3 Important Things About Variables



1: Name   2: Value   3: Type

3 Important Things About Variables


4765739
score
numeric
(an integer)
1: Name
2: Value
3: Type