Arrays and Loops

Programming for Engineers
Winter 2015

Andreas Zeller, Saarland University

Querying Sensors

```
int ledPin = 13;      // The LED
int buttonPin = 8;    // The button


void loop() {
    buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH)
        digitalWrite(ledPin, HIGH);
    else
        digitalWrite(ledPin, LOW);
}
```

Assignment

- The assignment

  name = value

  causes the variable name to have the new value
- As the program continues, every access to the variable name produces this value (until the next assignment)

Blinking with Millis

```
int ledPin = 13;      // Pin LED
int buttonPin = 8;    // Pin button

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop() {
    int ms = millis() % 1000;
    if (ms < 500)
        digitalWrite(ledPin, LOW);
    if (ms > 500)
        digitalWrite(ledPin, HIGH);
}
```

Debouncing with Millis

```
unsigned long previousPush = 0;

void loop() {
  if (millis() - previousPush >= 50) {
    if (!pushed && digitalRead(buttonPin) == HIGH) {
      previousPush = millis();
      ledStatus = !ledStatus;
      pushed = 1;
    }
    else if (pushed && digitalRead(buttonPin) == LOW){
      pushed = 0;
      previousPush = millis();
    }
  }
 // blinking...
}
```

Datatypes

- millis() has the type "unsigned long" – integer values in $[0 \ldots 2^{32}-1]$
- Usual integer numbers ("int") are in $[-2^{n-1} \ldots 2^{n-1}-1]$
- n is (depending on the device) 16 or 32
- 2^{15} Milliseconds = 32767 ms = 32,7 s
  2^{32} Milliseconds = 1193 h = 49,7 days

Overflow

- When we try to store a value that is too large for a datatype, there is an overflow
- Only the last (binary) digits are stored
- This can lead to arbitrary values

Boeing 787: US-Luftfahrtbehörde warnt vor Stromausfall im Dreamliner

REUTERS

Boeing 787: Programmierter Stromausfall nach 248 Tagen

Die US-Luftsicherheitsbehörde warnt: In Boeings Dreamliner kann im Flug der Strom ausfallen, das Flugzeug unkontrollierbar werden. Schuld ist ein Computerfehler.

Die US-Luftfahrtbehörde FAA (Federal Aviation Administration) hat eine Warnung für den Boeing-Langstreckenjet 787 Dreamliner herausgegeben. Demnach kann ein Softwareproblem dazu führen, dass in dem Flugzeug der Strom ausfällt, sodass die Piloten die Kontrolle über den Flieger verlieren würden.

Am Freitag hat die Behörde eine sogenannte Flugtauglichkeitsdirektive veröffentlicht, in der es

Boeing 787: US-Luftfahrtbehörde warnt vor Stromausfall im Dreamliner

- Power generator failure after 248 days of continuous operation
- 248 days = 2^{31} /100 seconds → int-overflow on time measuring
- If all four generators affected: catastrophe
- Workaround: restarting every 247 days at the latest

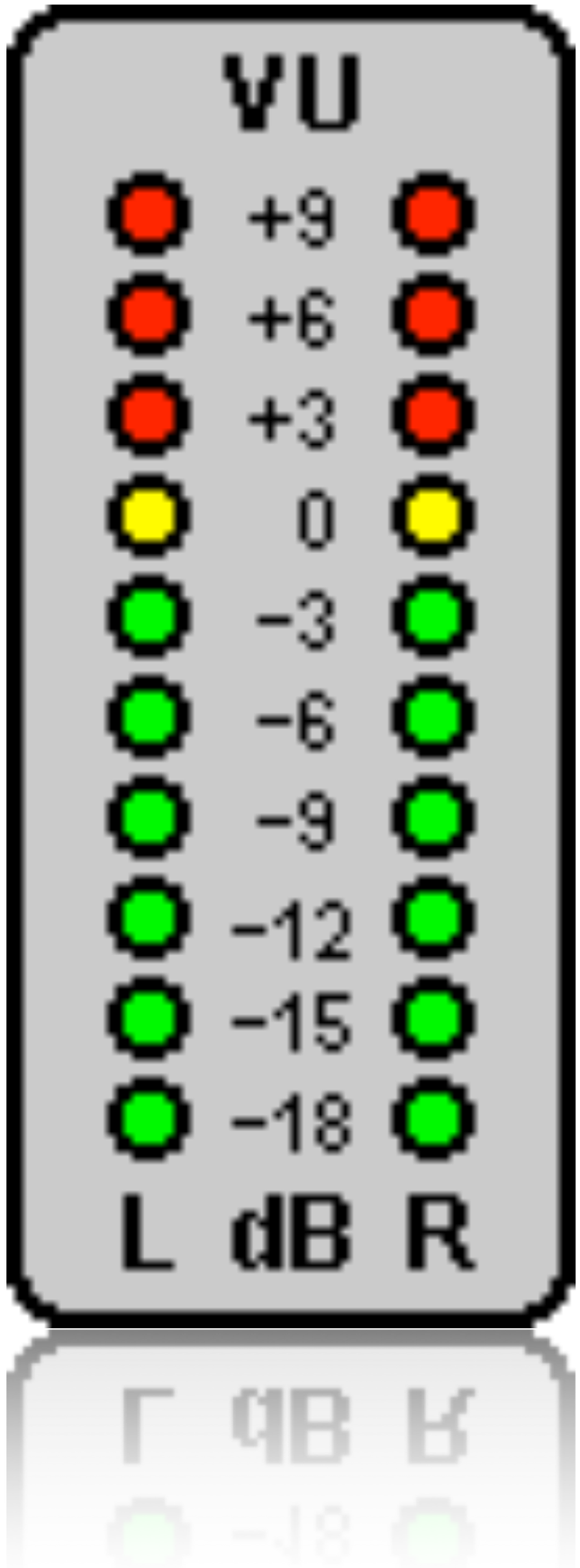
Boeing 787: Programmierter Stromausfall nach 248 Tagen

Today's Topics

- Arrays
- Loops
- Heartbleed

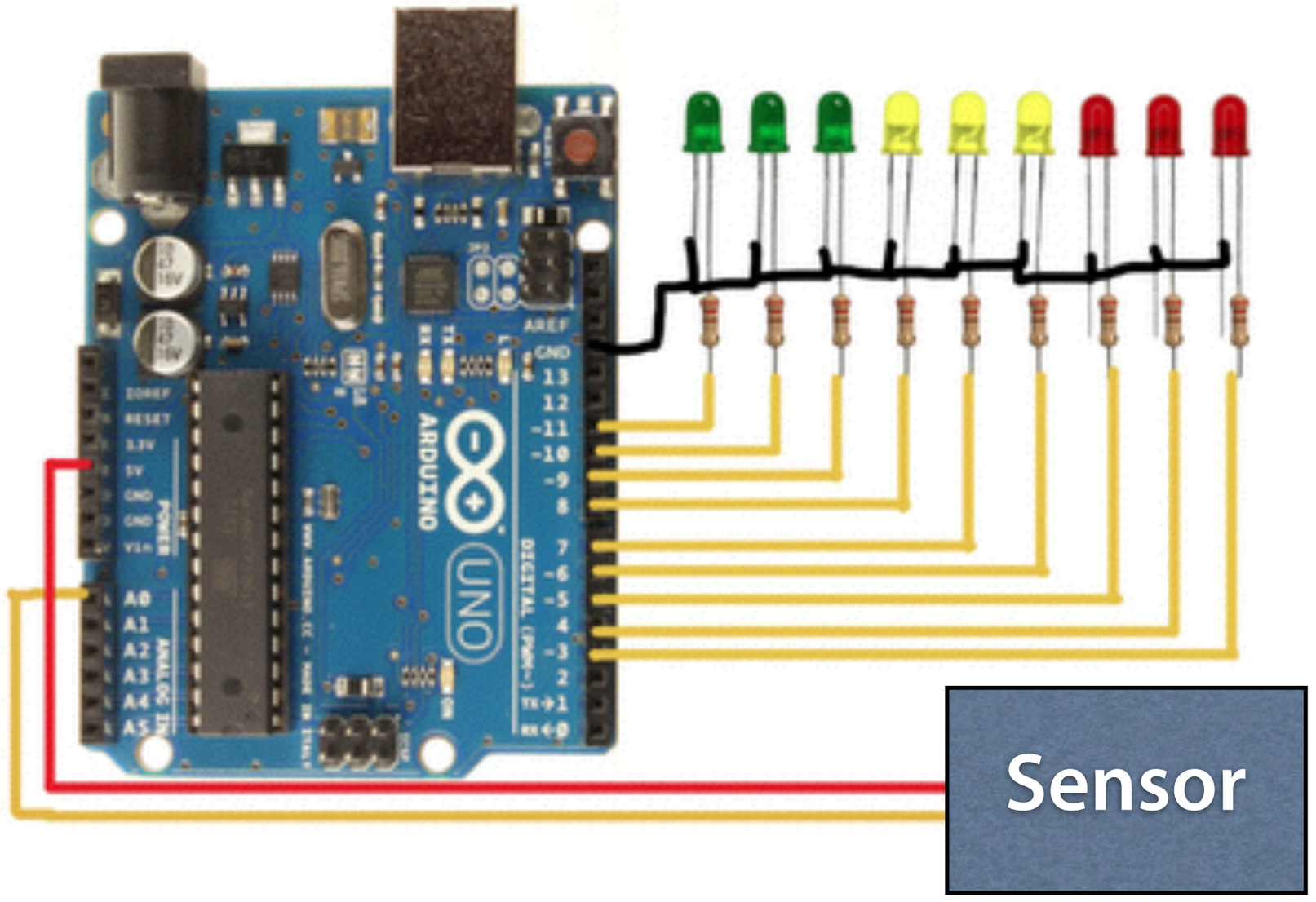
Level

LED-Chain shall display the level



Plan

- We connect a sensor – e.g. a photoresistor
- We measure its value
- We connect 7 LEDs
- We make 0 - 7 LEDs turn on depending on the level


ARDUINO
UNO
Sensor

Querying Sensors

- We already know digitalRead() which reads data digitally
- New: analogRead() reads data analogously

```
analogRead(pin_number)
```

returns a value of 0…1023,
depending on the voltage (0…5V)

Querying Sensors

```
int inputPin = 0;    // The input pin

void setup() {
     Serial.begin(9600);
}

void loop() {
     int level = analogRead(inputPin);

     Serial.println(level);
     delay(100);
}
```

Outputting Sensor Value

- 0…7 LEDs must be lit depending on the analogue input

Outputting Sensor Value

```
int inputPin = 0;    // The input pin
int led1 = 13;    // The LED pins
int led2 = 12;
int led3 = 11;
// etc. for 4 more LEDs

void setup() {
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    // etc.
}
```

```
int inputPin = 0;    // The input pin

int led1 = 13;    // The LED pins
int led2 = 12;
int led3 = 11;
// etc. for 4 more LEDs

void setup() {
     pinMode(led1, OUTPUT);
     pinMode(led2, OUTPUT);
     // etc.
}

void loop() {
     int level = analogRead(inputPin);

     if (level < 100)
          digitalWrite(led1, LOW);
     else
          digitalWrite(led1, HIGH);
```

```
void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);

    if (level < 200)
        digitalWrite(led2, LOW);
    else
        digitalWrite(led2, HIGH);

    if (level < 300)
        digitalWrite(led3, LOW);
    else
        digitalWrite(led3, HIGH);

    // etc. for 4 more LEDs
}
```

```
int inputPin = 0;   // The input pin

int led1 = 13;   // The LED pins
int led2 = 12;
int led3 = 11;
// etc. for 4 more LEDs

void setup() {
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    // etc.
}

void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);

    if (level < 200)
        digitalWrite(led2, LOW);
    else
        digitalWrite(led2, HIGH);

    if (level < 300)
        digitalWrite(led3, LOW);
    else
        digitalWrite(led3, HIGH);

    // etc. for 4 more LEDs
}
```

All variables are processed identically

Goal

We need a way to process multiple variables in the same way

Arrays

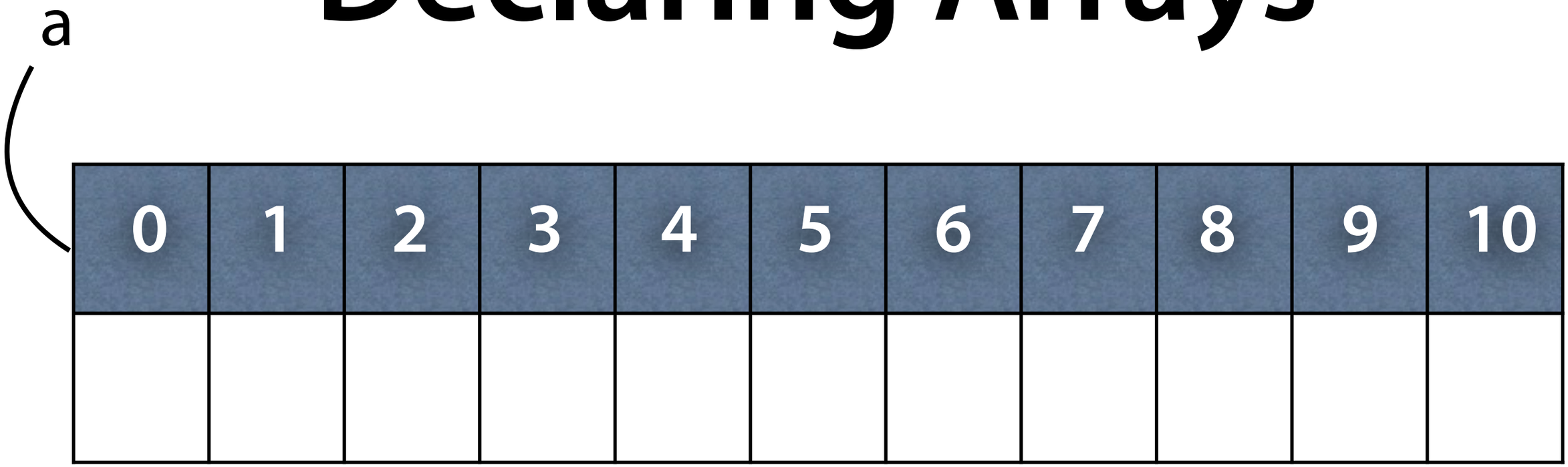
- An array unites multiple variable to a single entity
- Each element (each variable) is accessed by a number

Arrays

0	1	2	3	4	5	6	7	8	9	10

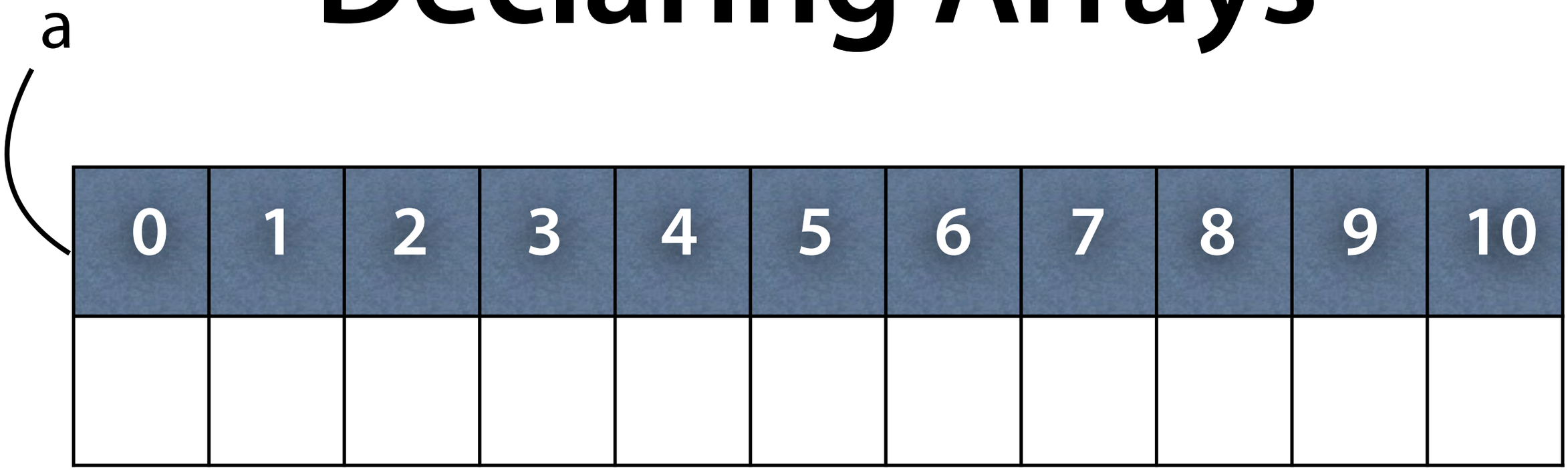
- The notation name[index] refers to the element with the number index
- The first element has the number 0

Declaring Arrays




```
int a[11];     // 11 elements
```

Declaring Arrays



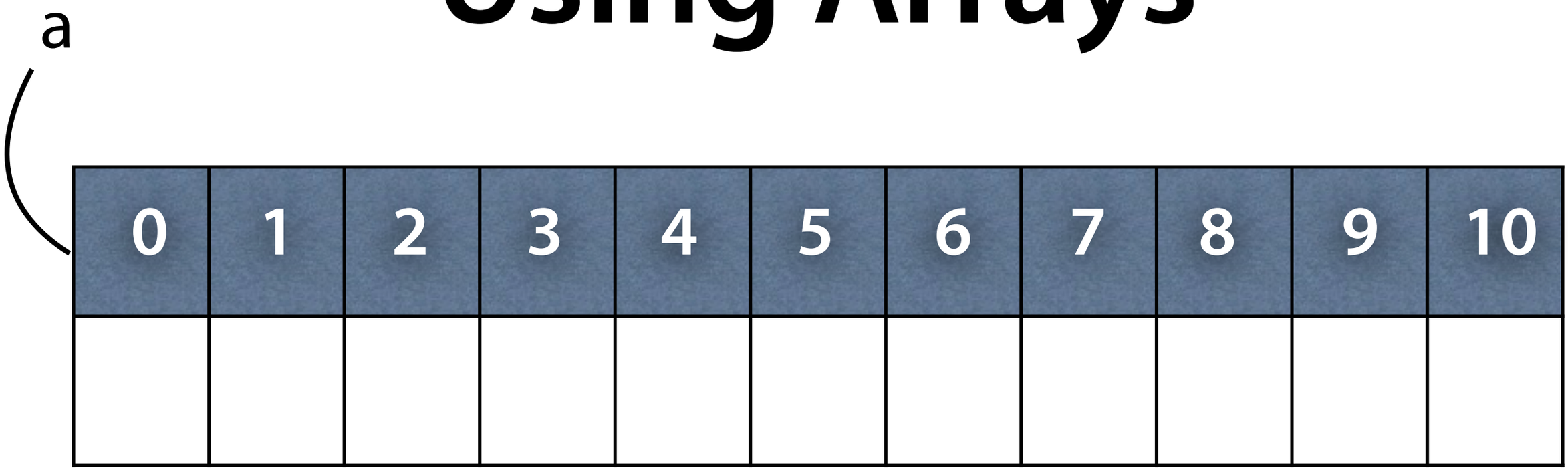

```
int a[11];     // 11 elements
```

Type of the elements

Name of the array

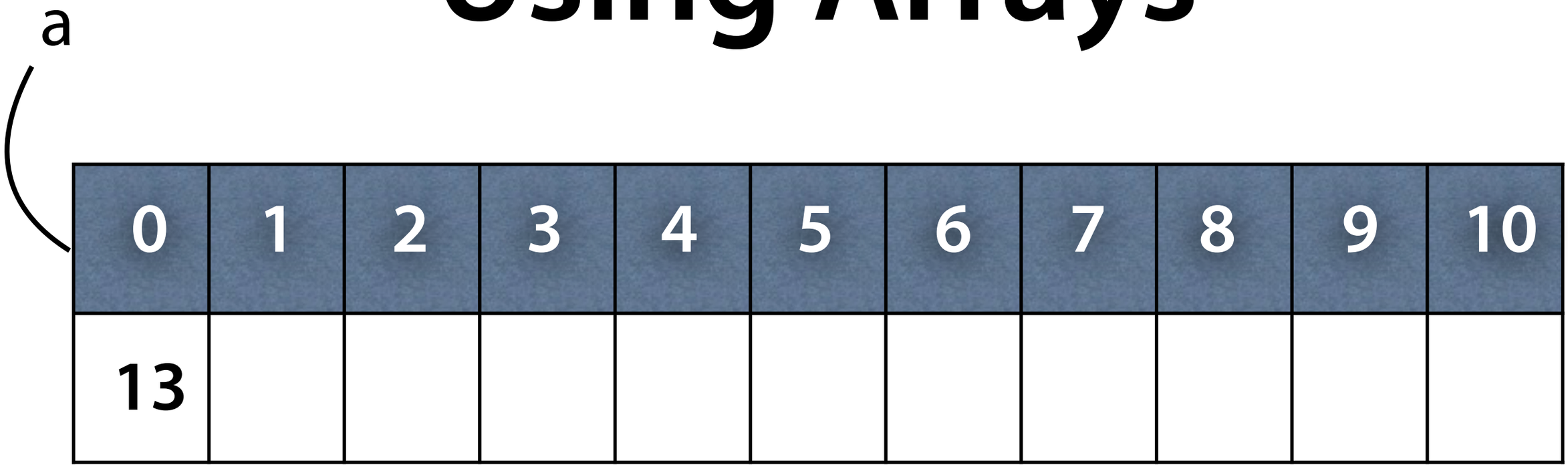
Size of the array

Using Arrays



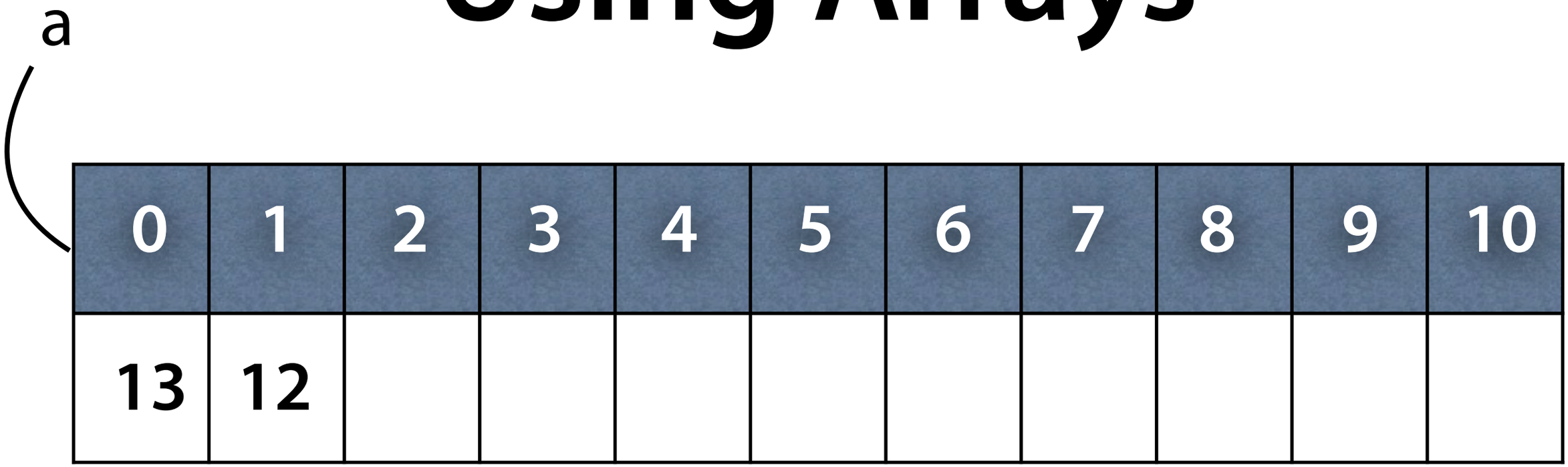

```
int a[11];     // 11 elements

void setup() {
    a[0] = 13;



}
```

Using Arrays




```
int a[11];     // 11 elements

void setup() {
    a[0] = 13;



}
```

Using Arrays




```
int a[11];     // 11 elements

void setup() {
    a[0] = 13;
    a[1] = 12;


}
```

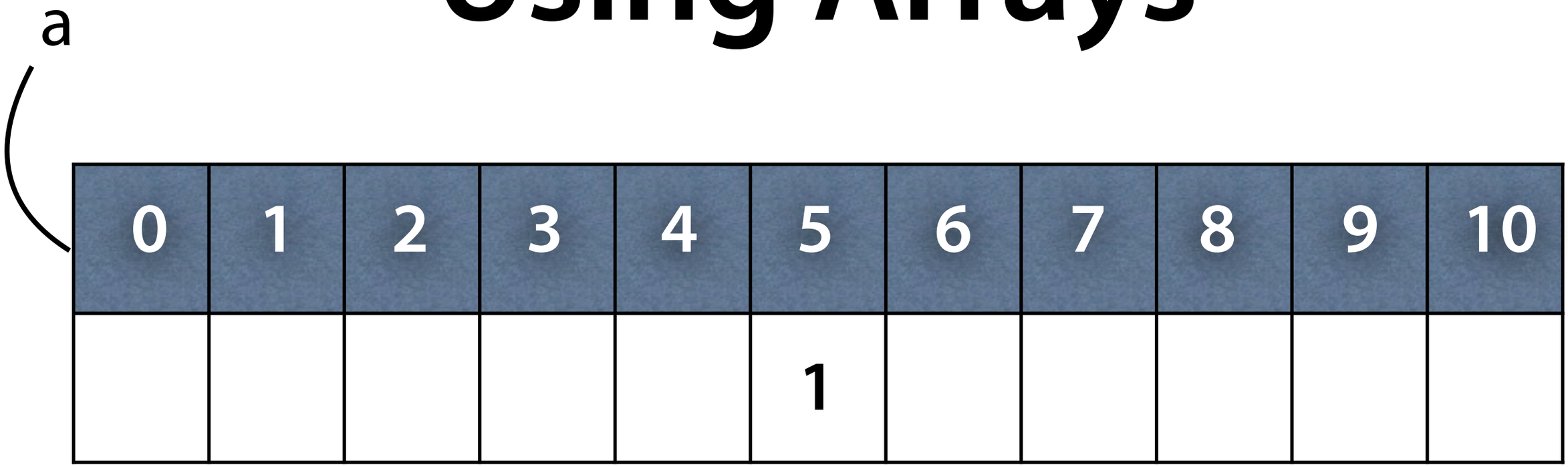
Using Arrays

a

0	1	2	3	4	5	6	7	8	9	10
13	12	25								

```
int a[11];     // 11 elements

void setup() {
    a[0] = 13;
    a[1] = 12;
    a[2] = a[0] + a[1];

}
```

Using Arrays




```
int a[11];     // 11 elements

void setup() {
    int i = 5;
    a[i] = 1;


}
```

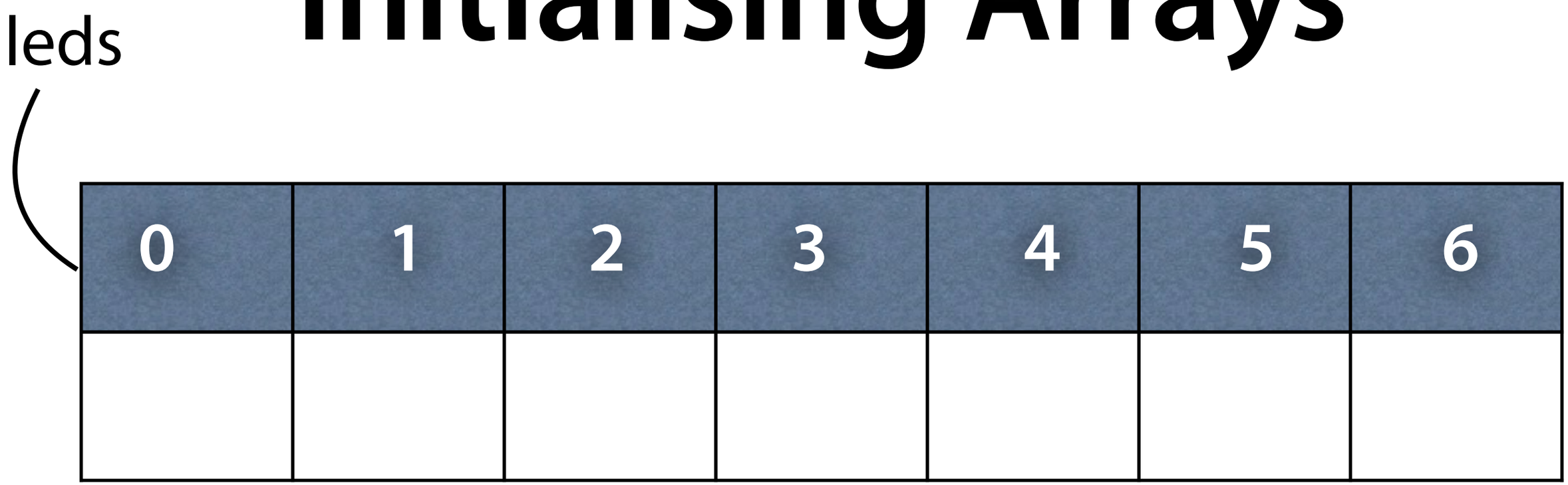
Using Arrays

a

0	1	2	3	4	5	6	7	8	9	10
	2				1					

```
int a[11];     // 11 elements

void setup() {
    int i = 5;
    a[i] = 1;
    a[a[i]] = 2;

}
```

Initialising Arrays



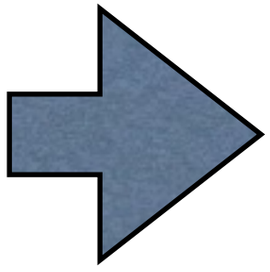

```
int leds[7];

void setup() {
    leds[0] = 13;
    leds[1] = 12;
    leds[2] = 11;
    leds[3] = 10;
    leds[4] = 9;
    leds[5] = 8;
    leds[6] = 7;
}
```

Initialising Arrays

leds

0	1	2	3	4	5	6
13	12	11	10	9	8	7

```
int leds[7];

void setup() {
    leds[0] = 13;
    leds[1] = 12;
    leds[2] = 11;
    leds[3] = 10;
    leds[4] = 9;
    leds[5] = 8;
    leds[6] = 7;
}
```



```
int leds[] =
    { 13, 12, 11, 10, 9, 8, 7 };

void setup() {
    // already initialised
}
```

Array Overflow


```
int a[11];     // 11 elements

void setup() {
    int x = a[11];
}
```

What happens here?

Undefined Behaviour

- When a program accesses an array outside its range, the behaviour is undefined (= anything can happen)
- More on this later

Beware the range on every array access!

Loops

- We still need a way process all elements equally

```
void setup() {
    pinMode(leds[0], OUTPUT);
    pinMode(leds[1], OUTPUT);
    pinMode(leds[2], OUTPUT);
    pinMode(leds[3], OUTPUT);
    pinMode(leds[4], OUTPUT);
    pinMode(leds[5], OUTPUT);
    pinMode(leds[6], OUTPUT);
}
```

What does this look like with 100 LEDs?

While-Loops

- A piece of a program can be repeated any number of times using a loop:

```
while (condition) {
    Instructions…;
}
```

- The instructions are repeated as long as the condition holds

While-Loops

```
i = 1;
while (i < 5) {
    Serial.println(i);
    i = i + 1;
}
Serial.println("END");
```

Executed Instructions

```
i = 1;
Serial.println(i);
i = i + 1; // 2
Serial.println(i);
i = i + 1; // 3
Serial.println(i);
i = i + 1; // 4
Serial.println(i);
i = i + 1; // 5
Serial.println("END");
```

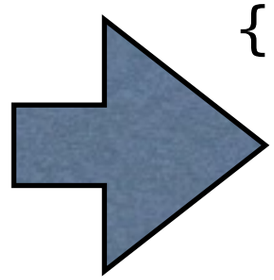
Setup with a Loop

- The setup()-code can now look like this:

```
void setup() {
     int i = 0;
     while (i < 7) {
          pinMode(leds[i], OUTPUT);
          i = i + 1;
     }
}
```

Loop with a Loop

```
void loop() {
    int level = analogRead(inputPin);

    if (level < 100)
        digitalWrite(led1, LOW);
    else
        digitalWrite(led1, HIGH);

    if (level < 200)
        digitalWrite(led2, LOW);
    else
        digitalWrite(led2, HIGH);

    if (level < 300)
        digitalWrite(led3, LOW);
    else
        digitalWrite(led3, HIGH);

    // etc. for 4 more LEDs
}
```



```
void loop() {
    int level = analogRead(inputPin);

    int i = 0;
    while (i < 7)
    {
        if (level < 100 * (i + 1))
            digitalWrite(leds[i], LOW);
        else
            digitalWrite(leds[i], HIGH);
        i = i + 1;
    }
}
```

Infinite Loop

- setup() and loop() get called by the system in an infinite loop:

```
void main() {
   setup();
   while (1) {
      loop();
   }
}
```

Intricacies

- Instead of *x = x + y* you can write *x += y* (similarly *x *= y*, *x /= y*, *x -= y*)

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i += 1;
    }
}
```

Increment

- Instead of *x* = *x* + 1 or *x* += 1
  you can write ++*x* or *x*++
  (similarly –-*x* or *x*-- for *x* –= 1*)*

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i++;
    }
}
```

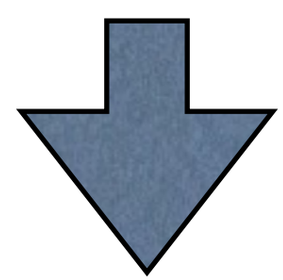
Pre- and Post Increment

- ++*x* first increments x and then returns the value
- *x*++ first returns the value of x and then increments

```
int x = 0;
int y = 0;

y = x++;
// x = 1, y = 0

y = ++x;
// x = 2, y = 2

y = x--;
// x = 1, y = 2
```

Inline Increment

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i++;
    }
}
```



```
void setup() {
    int i = 0;
    while (i < 7)
        pinMode(leds[i++], OUTPUT);
}
```

For-Loops

- If the number of repetitions is known, we often use for-loops:

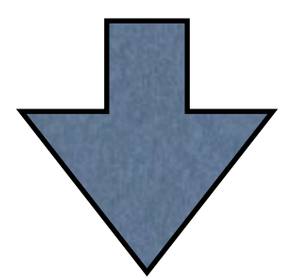
```
for (Initialisation; Condition; Increment) {
    Instructions…;
}
```

is the same as

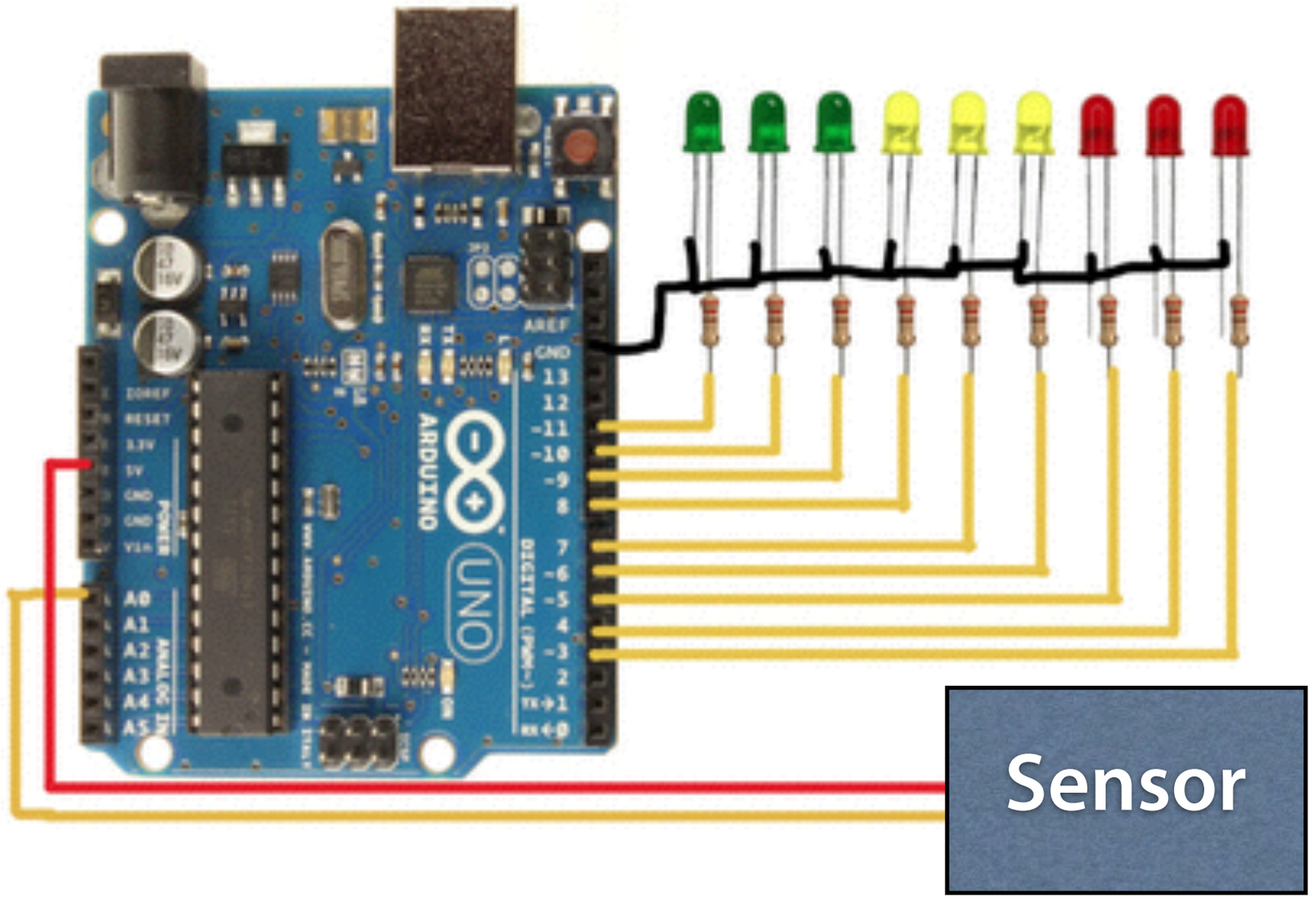
```
Initialisation;
while (Condition) {
    Instructions…;
    Increment;
}
```

Setup with For-Loop

```
void setup() {
    int i = 0;
    while (i < 7) {
        pinMode(leds[i], OUTPUT);
        i++;
    }
}
```



```
void setup() {
    for (int i = 0; i < 7; i++)
        pinMode(leds[i], OUTPUT);
}
```


ARDUINO
UNO
Sensor

Displaying the Level

```
void range(int d) {
  for (int i = 0; i < 7; i++)
    if (i < d)
      digitalWrite(leds[i], HIGH);
    else
      digitalWrite(leds[i], LOW);
}


void loop() {
    int level = analogRead(inputPin);
    range(level / 20);
}
```

Displaying a Dot

```
void dot(int d) {
  for (int i = 0; i < 7; i++)
    if (i == d)
      digitalWrite(leds[i], HIGH);
    else
      digitalWrite(leds[i], LOW);
}
```

```
void dot() {
    int level = analogRead(inputPin);
    dot(level / 100);
}
```

Chase

```
void dot(int d) {
  for (int i = 0; i < 7; i++)
    if (i == d)
      digitalWrite(leds[i], HIGH);
    else
      digitalWrite(leds[i], LOW);
}


void loop() {
    for (int i = 0; i < 7; i++) {
        dot(i);
        delay(20);
    }
}
```

Visibility

- Every variable is only defined inside its surrounding braces {…} (a block)

```
void loop() {
  for (int i = 0; i < 7; i++) {
    int n = i + 1;
    dot(n);
  }
  // n and i are no longer visible here
}
```

Visibility

- Every variable must have a distinct name inside a block
- Different blocks may use the same names for "their" variables

```
void dot(int d) {
  for (int i = 0; i < 7; i++) {
    ...
  }
}
```

Visibility

- Every variable must have a distinct name inside a block
- Different blocks may use the same names for "their" variables

```
void dot(int d) {
  for (int i = 0; i < 7; i++) {
    ...
  }
}
```

```
void loop() {
  for (int i = 0; i < 7; i++) {
    dot(i);
  }
}
```

Two different is

Visibility

- Accesses refer to the innermost name

```
int i = 1;
void loop() {
  Serial.println(i);
  for (int i = 2; i < 10; i++) {
    if (i < 10)
    {
      int i = 3;
      Serial.println(i);
    }
  }
}
```

Global Variables

- Variables that are not declared inside a block are called global variables
- Global Variables are risky, because you can use them unintentionally

Global Variables

- Variables that are not declared inside a block are called global variables
- Global Variables are risky, because you can use them unintentionally

```
float pi = 3.141526535;

void loop() {
  int py = 25;
  pi += 1;  // should be py
}
```

Heartbleed

OpenSSL-Sicherheitslücke: Warum "Heartbleed" Millionen Web-Nutzer gefährdet

Von Ole Reißmann

IT-Experten schlagen Alarm: Eine schwere Sicherheitslücke macht viele eigentlich besonders gesicherte Webseiten anfällig für Angriffe. Login-Daten und sensible Informationen sind in Gefahr, Nutzer sollten vorsichtshalber ihre Passwörter ändern.

Mittwoch, 09.04.2014 – 14:07 Uhr

Drucken | Senden | Merken

Nutzungsrechte | Feedback

Kommentieren | 65 Kommentare

Zur Startseite

Twittern 93 Empfehlen 511 g+1

THEMA
Heartbleed-Sicherheitslücke

Computersicherheit

Kryptografie

Internet

Heartbleed: So warnen IT-Experten vor dem schweren OpenSSL-Fehler

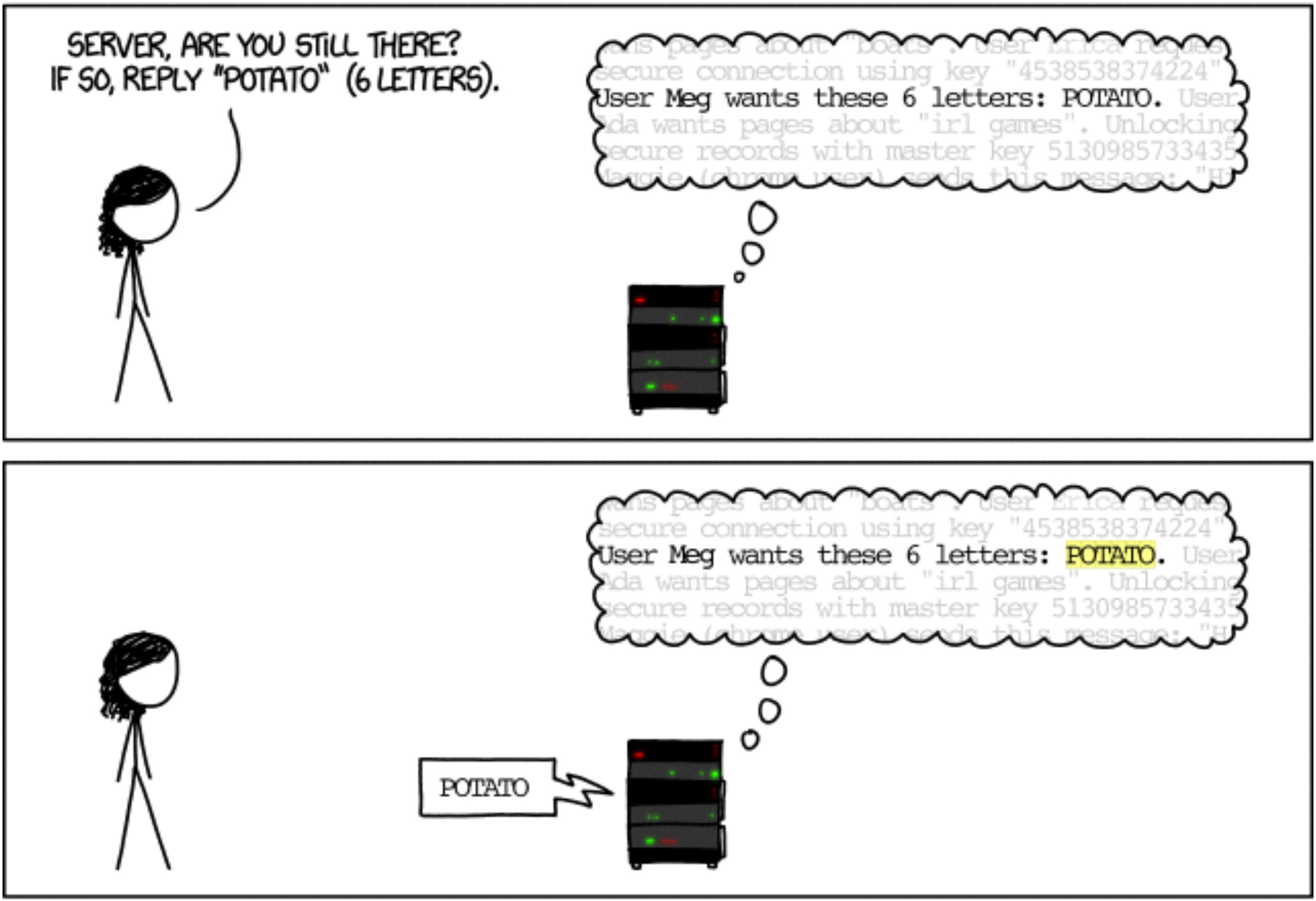
Was ist Heartbleed?

Heartbleed ist eine schwere Sicherheitslücke, die zwei Drittel des Webs betreffen könnte. Es geht um Webserver, die eigentlich eine gesicherte SSL-Verbindung anbieten, das erkennt man an dem "https" links oben in der Adresszeile. Solche Verbindungen werden immer dann eingesetzt

Heartbeat Protocol

- The heartbeat function is used to check whether a server on the Internet is still running
- One sends a string –
- and gets its back as an answer.

HOW THE HEARTBLEED BUG WORKS:



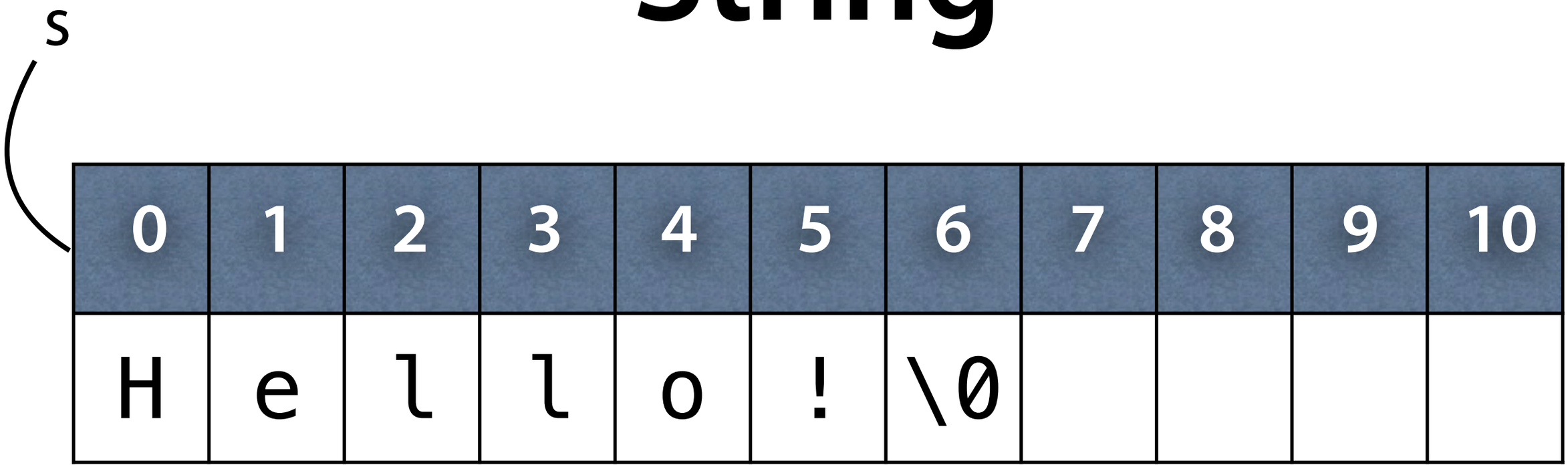
Characters in C

- A single character in C is written enclosed between two single quotes:

```
char c = 'a';
Serial.println(c);
```

- The most important use is as an array of characters (a string)
- Strings end with a special “null character”, written as ‘\0’

String




```
char s[] = { 'H', 'e', 'l', 'l', 'o', '!', '\0'};
```

or shorter

```
char s[] = "Hello!";
```

What is s[0]?

Reading Characters

- The function Serial.parseInt() returns a number from the serial interface:

```
int n = Serial.parseInt();
```

- With Serial.readBytes(buffer, n) one can read n characters into the array buffer:

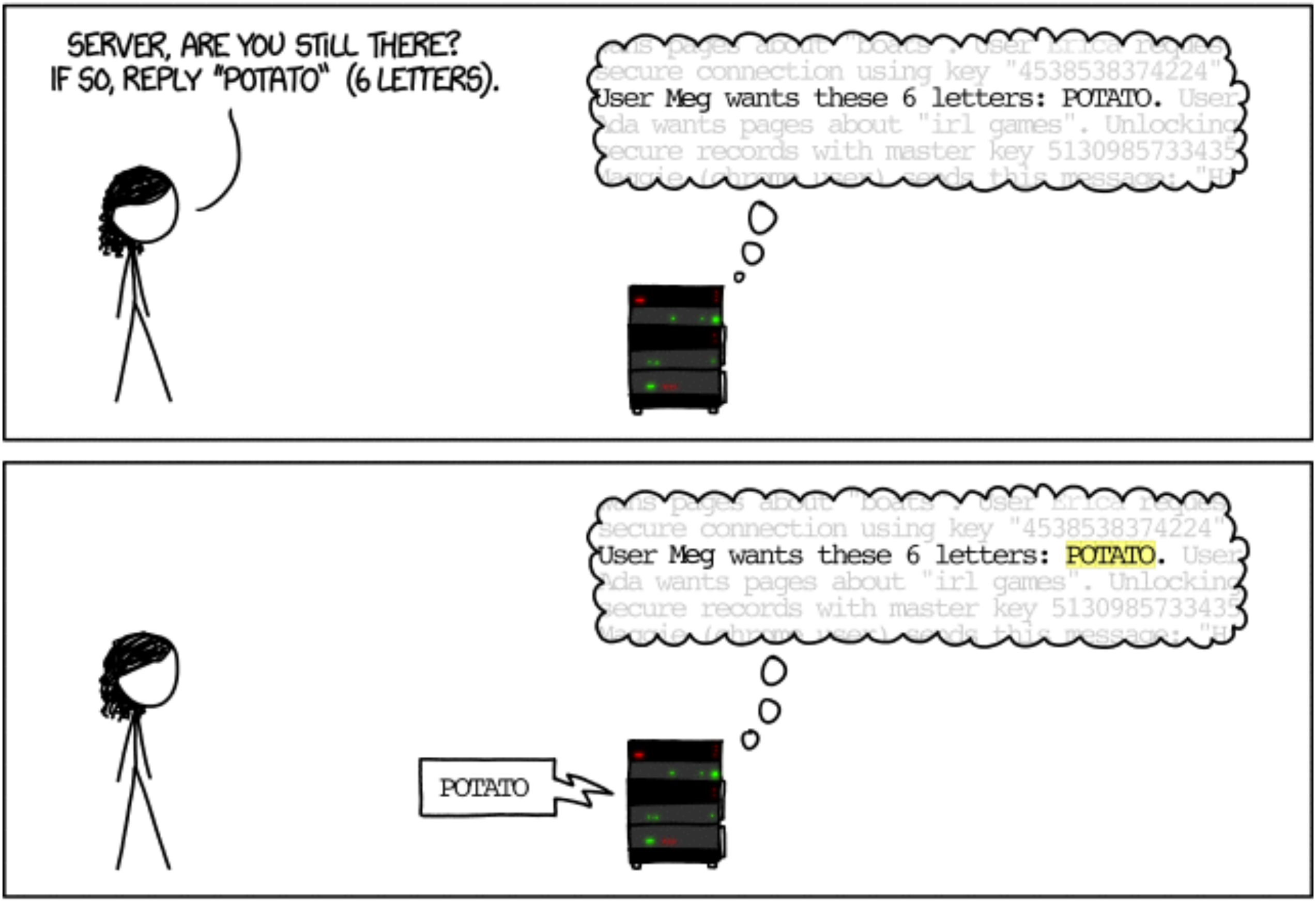
```
char buffer[20];
Serial.readBytes(buffer, n);
```

Heartbeat

```
void setup() {
  Serial.begin(9600);

  char secret[7] = "Joshua";      // secret data
  char buffer[10];                 // buffer

  while (1) { // infinite loop
     if (Serial.available() > 0) { // data available
      int n = Serial.parseInt();   // number of chars

      Serial.readBytes(buffer, n); // read characters

      for (int i = 0; i < n; i++)  // print characters
         Serial.print(buffer[i]);
       Serial.println();
     }
  }
}
```

Beware the range on every array access!

HOW THE HEARTBLEED BUG WORKS:



SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).

HMM...

SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).

User Meg wants these 500 letters: HAT.

User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHeRaSt". User

HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHeRaSt". User Meg requests pages

Heartbeat

```
void setup() {
  Serial.begin(9600);

  char secret[7] = "Joshua";      // secret data
  char buffer[10];                 // buffer

  while (1) { // infinite loop
     if (Serial.available() > 0) { // data available
      int n = Serial.parseInt();   // number of chars

      Serial.readBytes(buffer, n); // read characters

      for (int i = 0; i < n; i++)  // print characters
         Serial.print(buffer[i]);
       Serial.println();
     }
  }
}
```

Heartbleed

```
void setup() {
  Serial.begin(9600);

  char secret[7] = "Joshua";      // secret data
  char buffer[10];                 // buffer

  while (1) { // infinite loop
    if (Serial.available() > 0) { // data available
     int n = Serial.parseInt();   // number of chars

     Serial.readBytes(buffer, n); // read characters

     for (int i = 0; i < n; i++)  // print characters
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```

Correction

```
void setup() {
  Serial.begin(9600);

  char secret[7] = "Joshua";      // secret data
  char buffer[10];                 // buffer

  while (1) { // infinite loop
      if (Serial.available() > 0) { // data available
       int n = Serial.parseInt();   // number of chars
       if (n > 10) { n = 10; }      // avoid overflow
       Serial.readBytes(buffer, n); // read characters

       for (int i = 0; i < n; i++)  // print characters
          Serial.print(buffer[i]);
        Serial.println();
      }
   }
}
```

Beware the range
on every array access!

Don't confuse
= and ==

Avoid Overflows!

space.com

Outlook

- More on Strings
- Interacting with an LCD-Display

Initialising Arrays

leds

0	1	2	3	4	5	6
13	12	11	10	9	8	7

```
int leds[7];

void setup() {
   leds[0] = 13;
   leds[1] = 12;
   leds[2] = 11;
   leds[3] = 10;
   leds[4] = 9;
   leds[5] = 8;
   leds[6] = 7;
}
```

```
int leds[] =
    { 13, 12, 11, 10, 9, 8, 7 };

void setup() {
   // already initialised
}
```

While-Loops

```
i = 1;
while (i < 5) {
   Serial.println(i);
   i = i + 1;
}
Serial.println("END");
```

Executed Instructions

```
i = 1;
Serial.println(i);
i = i + 1; // 2
Serial.println(i);
i = i + 1; // 3
Serial.println(i);
i = i + 1; // 4
Serial.println(i);
i = i + 1; // 5
Serial.println("END");
```

String

s

0	1	2	3	4	5	6	7	8	9	10
H	e	l	l	o	!	\0				

```
char s[] = { 'H', 'e', 'l', 'l', 'o', '!', '\0'};
```

or shorter

```
char s[] = "Hello!";
```

Heartbeat

```
void setup() {
  Serial.begin(9600);

  char secret[7] = "Joshua";     // secret data
  char buffer[10];                // buffer

  while (1) { // infinite loop
    if (Serial.available() > 0) { // data available
     int n = Serial.parseInt();   // number of chars

     Serial.readBytes(buffer, n); // read characters

     for (int i = 0; i < n; i++)  // print characters
        Serial.print(buffer[i]);
      Serial.println();
    }
  }
}
```