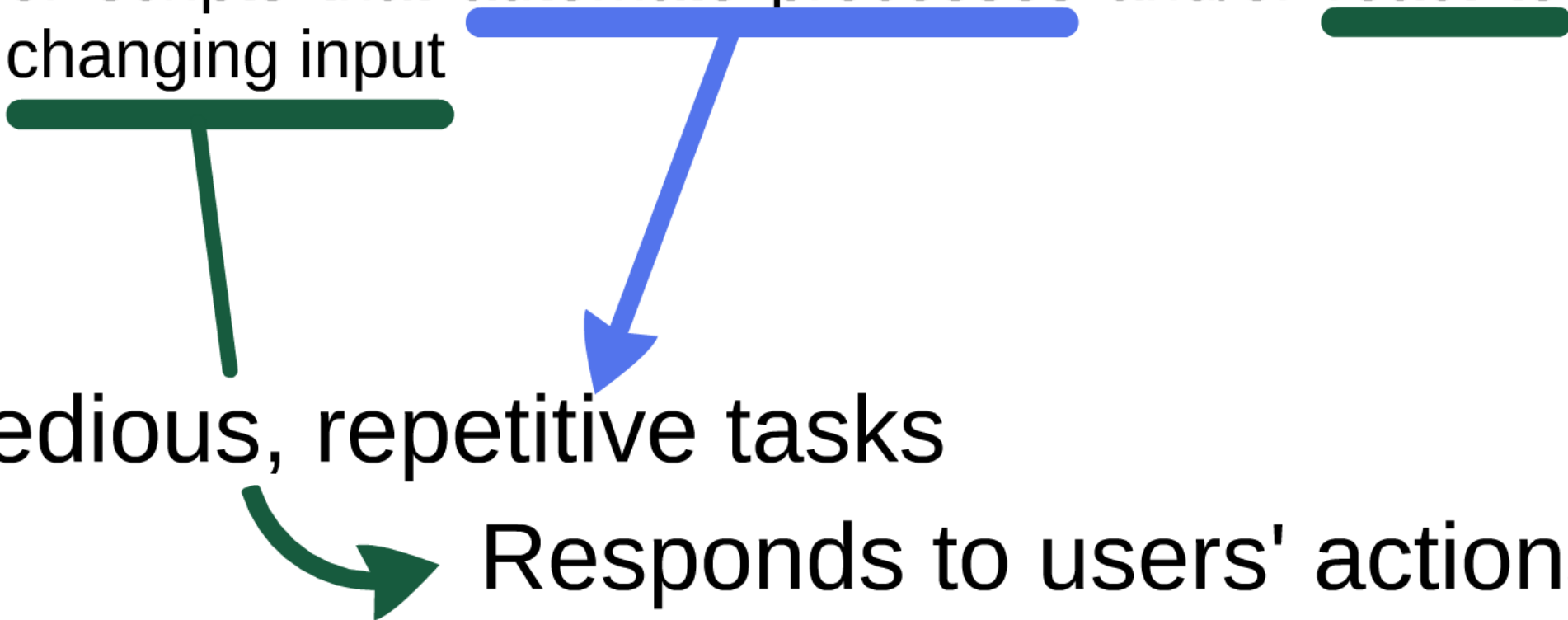


JS



What is programming?

The purpose of programming is to create programs or scripts that automate processes and/or react to changing input



```
graph TD; A[automate processes] --- B[Responds to users' actions]; C[changing input] --- D[Tedious, repetitive tasks]; D --> B
```

Tedious, repetitive tasks

Responds to users' actions

Write it in code, which can be run on a computer

Fundamental aspects

There are aspects which are fundamental to any programming task.

- Ability to output information
- Storage of information for the use of the program
- Being able to perform operations on data
- Ability to read information.

Higher level language

Written in English words. Why?

Like CSS there's syntax that you have to observe, or else it's not going to work.

Also like CSS, there's a pre-built library of functions that the language can use

Obviously there are several steps that have to be taken between having code, and a working program

With specialised software, and an interpretable language, you don't have to worry about this though

Not all languages are equal

Many programming languages have been designed with particular goals in mind.

Sometimes they have been developed to solve problems with existing languages, or introduce features that are unavailable.

Nonetheless, many of the basics are the same between different languages

Generally made by programmers **cough** not always

Easier for programmers to learn

Why reinvent the wheel?

Outputting information

One of the key aspects of any language is the ability to output information, particularly in the form of text

Two purposes: for ***programmers*** (debugging)
users

"Text" includes textual representations of numbers and other types of character #) < ! @ * ;

...remember that HTML is saved as text

It's important that output isn't necessarily fixed, but can be used to output data that has been changed in the course of the program.

Memory

```
5 + 6;    33*1231;
```

```
console.log("The calculation was completed");
```

The key is being able to store information
...specifically, in the computer's memory...
and then have the script use that information

```
some_place_in_memory ← 33*1231;
```

```
console.log("The calculation was completed");
```

```
console.log("The value was "+ some_place_in_memory);
```

```
The calculation was completed  
The value was 40623
```

Variables

The way to store information in memory is through the use of something called variables

Variables can store all sorts of information

Some languages demand that specific types of variable are made to deal with different types of data.

variables designed for whole numbers

variables designed for decimal numbers

variables designed for textual characters

Other languages "don't care" and provide a "one-size-fits-all" container to store any type of data

Variable usage

In order to use variables you must name them.

The actual allocation of the variable, and its data, to memory is handled by back-end processes we never have to look at.

To create a variable you can simply declare it.

```
var thisIsMyVariable = 76; ←
```

```
var THISISANOTHER_VARIABLE = 21;
```

```
var third_value = 228 * 3;
```

```
console.log("The value's " + third_value);
```

The value's 228

Variable usage II

Provided that a script knows about a particular variable, it can be used in all sorts of circumstances.

```
var t = 5;
```

```
t + 5;      console.log("Hello "+ t);      Hello 5
```

```
t = t + 5;  console.log("Hello "+ t);      Hello 10
```

```
console.log("Hello "+ (t+5));
```

Hello 15

```
t = t * t;
```

```
console.log("Hello "+ t);
```

Hello 100



what?

```
console.log("Hello "+ forgotToMakeThis);
```

Uncaught ReferenceError: forgotToMakeThis is not defined

```
var z = 18;  
console.log(z);
```



18

```
var g = "Hello ";  
console.log(g+z);
```

Hello 18

```
g = z;  
console.log(g);
```



Give the thing on the left the value of the thing on the right and save it for later use.

18

Variable operations

The script has a pre-built understanding of simple operations

| | |
|----------------|---|
| addition | + |
| multiplication | * |
| division | / |
| subtraction | - |
| modulus | % |

You can also use brackets as you would in maths

$(12+8)*5$

You can also compare values

Yes, = is already taken... as such we use `===`

Value comparison

When you compare two values the answer is going to be either true or false.

Is eight greater than five?

Is eleven equal to eleven?

TRUE

Is 7 less than 3?

FALSE

greater than >

less than <

equal to ===

```
console.log(11===11);
```

true

```
console.log(10<2);
```

false

Logic of the machine

These comparisons (called logical operators) can be used to great effect.


You can store these true and false values as you would any other value.

```
var wut;  
wut = (5>2)  
console.log(wut);  
true
```

But far more useful is to make conditions based off of logical operators.


To make a condition you use the special word "if"

```
var p = 2;  
if (5 > 2)  
    p = 15;  
console.log(p);
```



15

```
var p = 2;  
if (5 < 2)  
    p = 15;  
console.log(p);
```



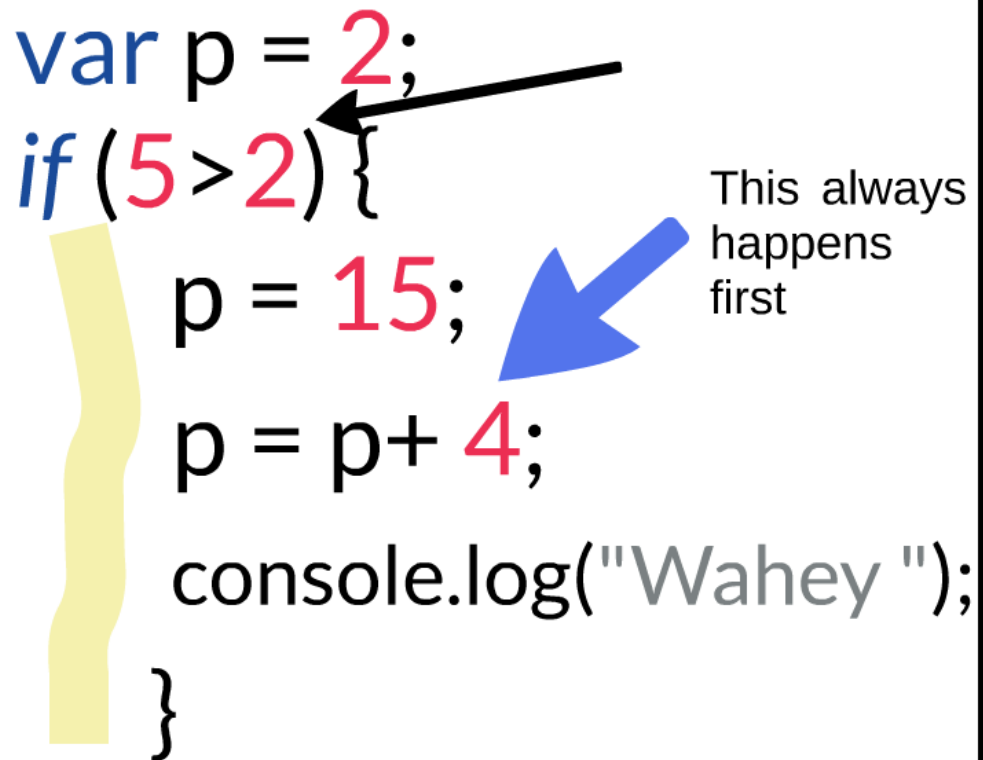
2

```
var p = 2;  
if (p > 2)  
    p = 15;  
console.log(p);
```

2

Power of if

```
var p = 2;  
if (5 > 2) {  
  p = 15;  
  p = p + 4;  
  console.log("Wahey ");  
}
```



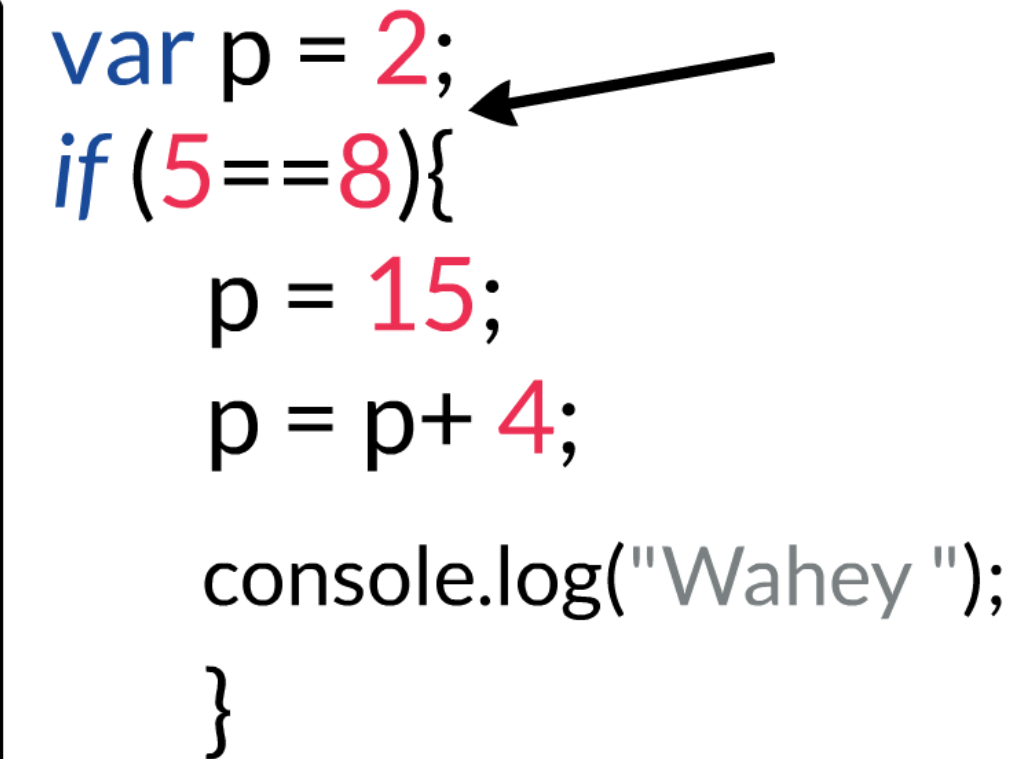
This always happens first

console.log(p);

Wahey

19

```
var p = 2;  
if (5 == 8) {  
  p = 15;  
  p = p + 4;  
  console.log("Wahey ");  
}
```



console.log(p);

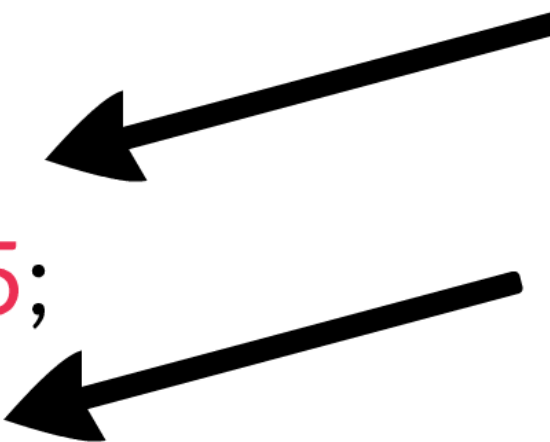
2

More options! Or *else*...

What if you have some sort of condition, but you want something else to happen if that condition isn't met?

You can use the other special word "else"

```
var p = 2;  
if (p > 2)  
    p = 15;  
else  
    p = 9;  
console.log(p);
```



The diagram consists of two black arrows. The first arrow originates from the right side of the 'if' keyword and points diagonally down and to the left towards the code block 'p = 15;'. The second arrow originates from the right side of the 'else' keyword and points diagonally down and to the left towards the code block 'p = 9;'. This visually links the control flow keywords to their respective execution paths.

Else... if...

"else" and "if" can be combined to create additional conditions.


```
if (20 > 100)  
    console.log("Huge");
```



```
else if (20 > 50)  
    console.log("Medium");
```



```
else if (20 > 10)  
    console.log("Small");
```



```
else if (20 > 5)  
    console.log("Tiny");
```



Final example

```
var m = 23;  
var f = 4;  
if (m === f)  
    console.log("They're the same");  
else if (m < f)  
    console.log("f is bigger");  
else  
    console.log("m must be bigger");
```

m must be bigger

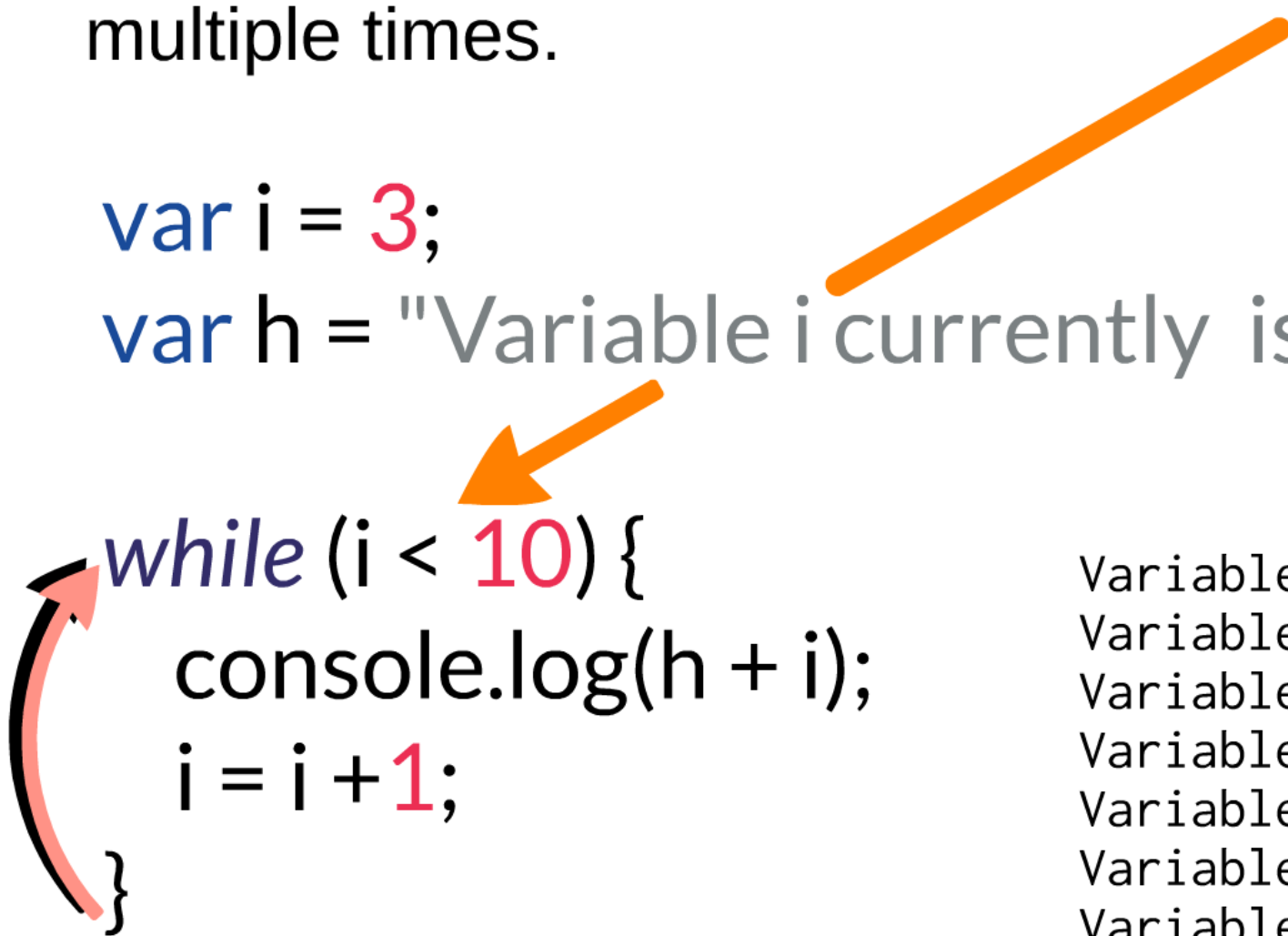
Doing things for a while

A loop makes the script repeat the same action multiple times.

```
var i = 3;
```

```
var h = "Variable i currently is "
```

```
while (i < 10) {  
    console.log(h + i);  
    i = i + 1;  
}
```



| |
|---------------------------|
| Variable i currently is 3 |
| Variable i currently is 4 |
| Variable i currently is 5 |
| Variable i currently is 6 |
| Variable i currently is 7 |
| Variable i currently is 8 |
| Variable i currently is 9 |

Any questions?

Variables

Assignment

Operations

Logic

Loops

Next day: deep dive into JavaScript