Events!

```
// Get an element
var button = document.querySelector("#background-button");

// Define the event handler function
function changeBackground() {
    button.style.backgroundColor = "red";
}

// Bind the event handler to the event
button.addEventListener("click", changeBackground);
```

Browser Events

- Megalist of events
- Mouse events
  - click, mousedown, mouseup, mousemove, mouseenter, ...
- Keyboard events
  - keydown, keypress, keyup
- View events
  - resize, scroll
- Sensor events
  - orientationchange, devicemotion, etc.
- Etc.

Function Scope

LOCAL VARIABLE SCOPE

```
function greetPerson(greeting, person) {
    var message = greeting + ", " + person + ".";
    document.write("<h1>" + message + "</h1>");
}

greetPerson("Good morning", "Mr. President");
console.log(message); // ERROR
console.log(greeting); // ERROR
```

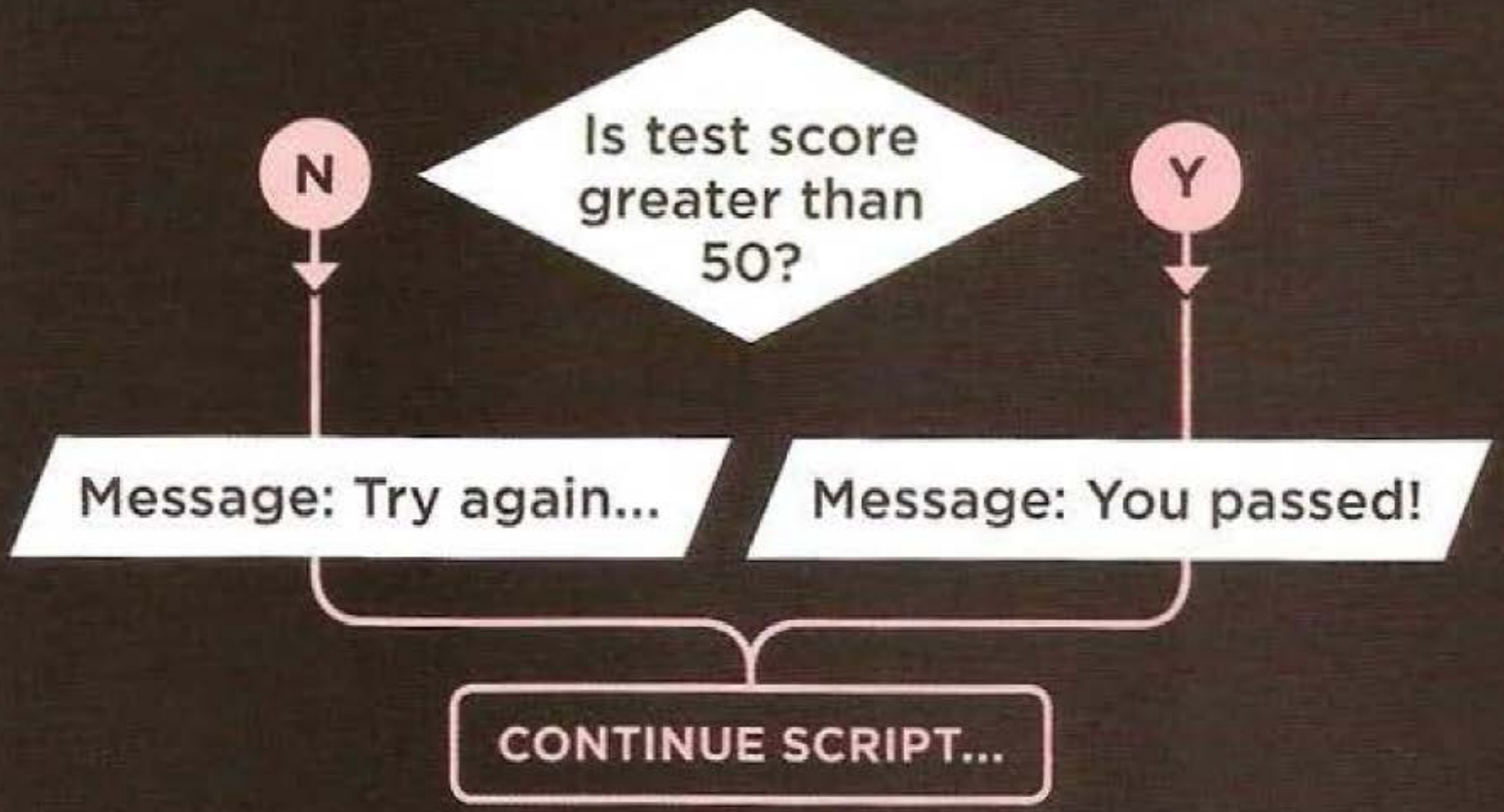
GLOBAL SCOPE

```
var greetingHeader = document.getElementById("greeting");

function greetPerson(greeting, person) {
    var message = greeting + ", " + person + ".";
    greetingHeader.textContent = message;
}

greetPerson("Good morning", "Mr. President");
console.log(greetingHeader); // NO ERROR
console.log(message); // ERROR
console.log(greeting); // ERROR
```

Making Decisions

Conditionals

“if” Statement



From JavaScript & jQuery

```
if (score >= 50) {
    console.log("You passed!");
}
```

CONDITION

```
if (score >= 50) {
    console.log("You passed!");
}
```

RESULT

Comparison Operators

```
>
>=
<
<=
=== (or ==)
!== (or !=)
```

“If…else if…else” Statement

```
var score = 90;

// "If...else if...else" statement starting

if (score >= 90) {
    console.log("You aced it!");
}
else if (score >= 50) {
    console.log("You passed!");
}
else {
    console.log("You failed!");
}

// If statement over, the script will continue
```

```
var score = 90;

// "If...else if...else" statement starting

if (score >= 90) {
    console.log("You aced it!");
}
else if (score >= 50) {
    console.log("You passed!");
}
else {
    console.log("You failed!");
}

// If statement over, the script will continue
```

IF STATEMENT

IF CLAUSE

ELSE IF CLAUSE

ELSE CLAUSE

Booleans

```
var isRaining = true;
```

BOOLEAN VARIABLE

```
if (isRaining) {
    console.log("Bring an umbrella");
}
```

```
var score = Number(prompt("What was the score?"));
var hasPassedTest = (score >= 50);
```

BOOLEAN VARIABLE

```
if (hasPassedTest) {
    console.log("You passed, ace.");
}
```

Nested “if” Statement

OUTER

INNER

```
if (score1 >= 50) {

    if (score2 >= 50) {
        console.log("Congrats - you passed both tests.");
    }

}
```

```
if (score1 >= 50) {
    if (score2 >= 50) {
        if (score3 >= 50) {
            if (score4 >= 50) {
                if (score5 >= 50) {
                    console.log("Congrats - you passed all the tests.");
                }
            }
        }
    }
}
```

Logical Operators

&&

LOGICAL AND

This operator tests more than one condition.

`((2 < 5) && (3 >= 2))`
returns `true`

If both expressions evaluate to `true` then the expression returns `true`. If just one of these returns `false`, then the expression will return `false`.

```
 true && true  returns true
 true && false returns false
false && true  returns false
false && false returns false
```

||

LOGICAL OR

This operator tests at least one condition.

`((2 < 5) || (2 < 1))`
returns `true`

If either expression evaluates to `true`, then the expression returns `true`. If both return `false`, then the expression will return `false`.

```
 true || true  returns true
 true || false returns true
false || true  returns true
false || false returns false
```

!

LOGICAL NOT

This operator takes a single Boolean value and inverts it.

`!(2 < 1)`
returns `true`

This reverses the state of an expression. If it was `false` (without the `!` before it) it would return `true`. If the statement was `true`, it would return `false`.

```
!true  returns false
!false returns true
```

From JavaScript & jQuery

COMPOUND CONDITION

CONDITION 1 CONDITION 2

```
if ((score1 >= 50) && (score2 >= 50)) {
```

AND

COMPOUND CONDITION

CONDITION 1 CONDITION 2

```
if ((score1 >= 50) || (score2 >= 50)) {
```

OR

&&
Logical AND

||
Logical OR

!
Logical NOT

a	b	a && b	a | | b	!a
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

```
(...)          // Grouping
! ...          // Logical NOT
... * ...      // Multiplication
... / ...      // Division
... + ...      // Addition
... - ...      // Subtraction
... < ...      // Less than
... <= ...     // Less than or equal to
... > ...      // Greater than
... >= ...     // Greater than or equal to
... === ...    // Strict equality
... !== ...    // Strict inequality
... && ...     // Logical AND
... || ...     // Logical OR
... = ...      // Assignment
```

HTML5 Canvas

Canvas

- A container for graphics
- Bitmap graphics (as opposed to SVG)
- HTML5 Canvas Resources:
  - Dive Into HTML5 – Canvas
  - MDN Canvas Tutorial
- Using canvas directly is tedious
  - Fabric.js
  - CreateJS

p5.js

Processing, Reinterpreted for JavaScript

What is p5?

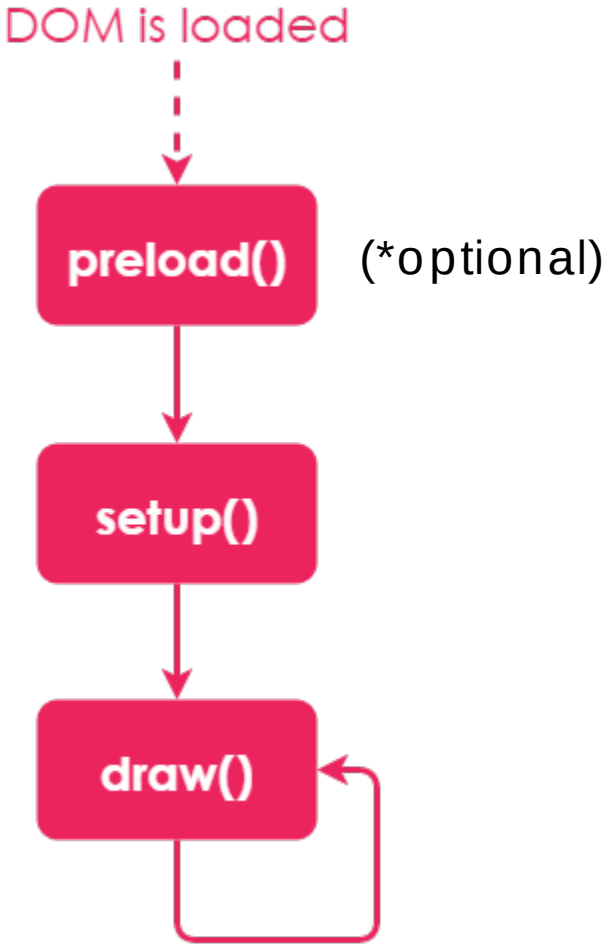
- Creative coding for the web
- It provides:
  - An easy way to use HTML5 Canvas
  - Makes life easier for: mouse tracking, colors, events, images, math, shapes, typography, sounds, etc.
  - Extra libraries
- Resources
  - Tutorials
  - Examples
  - Reference

What is p5?

It is just JavaScript code

(that other people wrote)

p5 is a framework

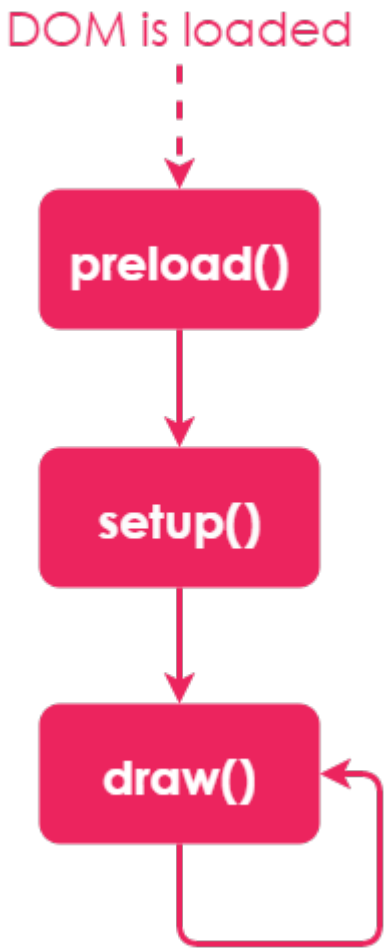


(*optional)

p5 is a framework

p5 hooks into three main functions that you write:

- **preload()** – optional, runs once and is used for loading sounds, images, etc.
- **setup()** – runs after preload(), runs once and is used for initialization tasks like creating a canvas
- **draw()** – runs after setup(), runs 60 times a second and is used for updating the screen (and running logic)



Useful Built-in Variables

These variables are created and updated by p5 behind the scenes:

- width, height
- windowWidth, windowHeight
- keyIsPressed, key, keyCode
- mouseX, mouseY
- mouseIsPressed, mouseButton

Reference

(keep this tab open)