

Coding in Javascript

Apr 2018

Instructors:

Jeremy

Introductions.

What is your name?

What would you like
to learn?

Breaks.
~10:30am

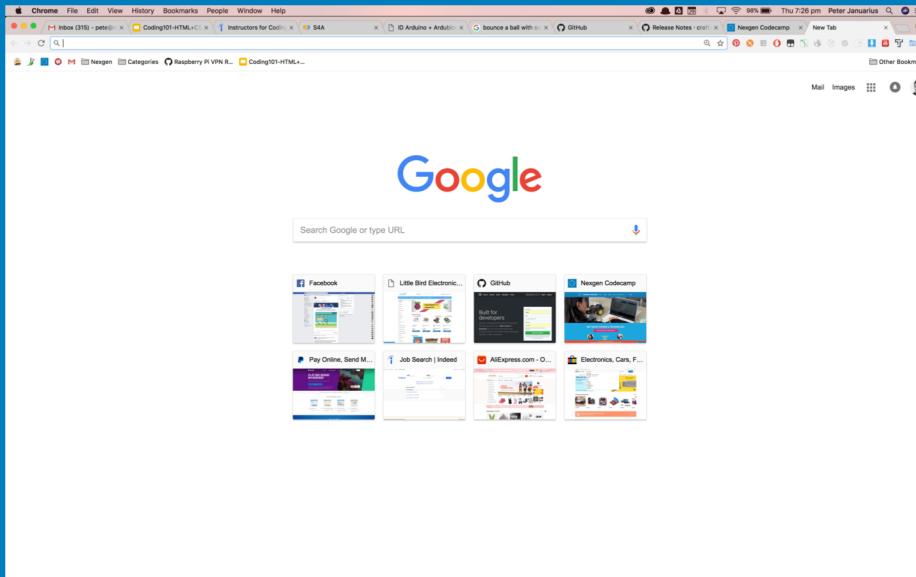
~12:30pm

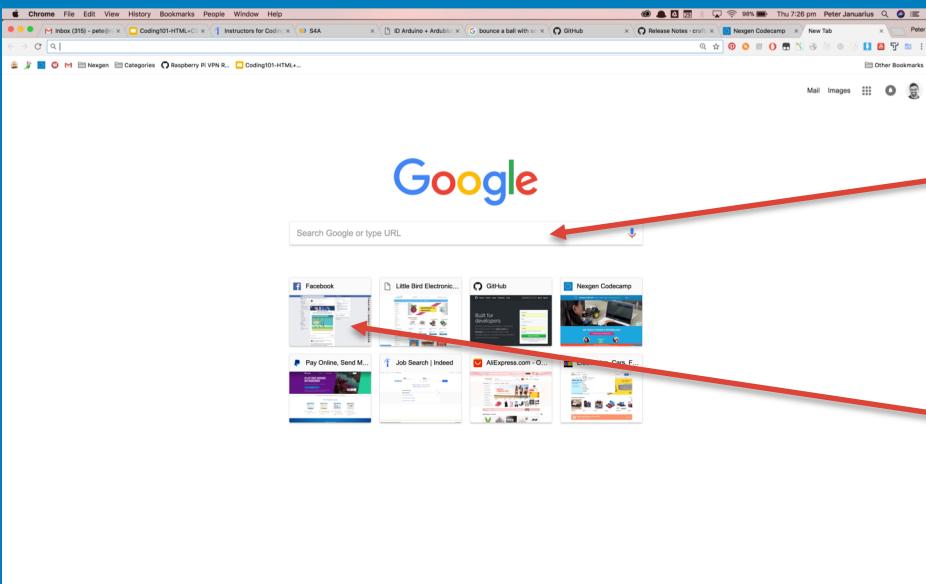
What we will be doing

- Learning Javascript
- Basic coding concepts
- Building an endless shooter
- Using our own web servers

Lesson 1 - Concepts & Game Intro

Every webpage is rendered (displayed) by your browser in HTML



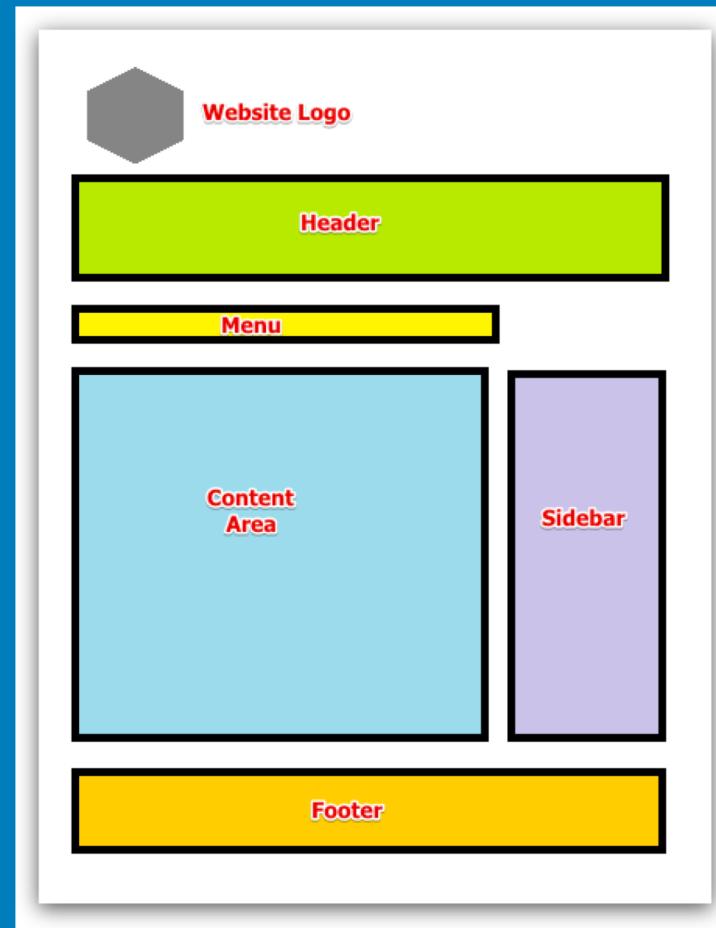


```

<!DOCTYPE html>
<html lang="en-AU">
  <head>...</head>
  <body class="default-theme des-mat" style="background: #fff;">
    <div class="mngb">...</div>
    <div id="TZA4S">
      <div class="lga">...</div>
      <form action="/search" id="f" method="get">...</form>
      <div class="mv-hide" id="most-visited">
        <div id="mv-tiles" style="width: 681px;">
          <iframe id="mv-single" src="chrome-search://most-visited/single.html?removeToo...
            <#document
              <!DOCTYPE html>
              <html>
                <!-- Copyright 2015 The Chromium Authors. All rights reserved.
                    Use of this source code is governed by a BSD-style license that can
                    be found in the LICENSE file. -->
              <head>...</head>
              <body style="--tile-title-color:#000000;"> == $0
                <div id="most-visited">
                  <div id="mv-tiles" style="opacity: 1;">
                    <a class="mv-tile" data-tid="1" href="https://www.facebook.com/" aria-l...
                    <a class="mv-tile" data-tid="2" href="http://littlebird.com.au/" aria-l...
                    Australia - In Stock | Little Bird Electronics" title="Australia - In Stock | Little Bird Electronics">...</a>
                    <a class="mv-tile" data-tid="3" href="https://github.com/" aria-l...
                    <a class="mv-tile" data-tid="4" href="https://nexgencodecamp.com.au/" aria-l...
                    <a class="mv-tile" data-tid="5" href="https://paypal.com/" aria-l...
                    Australia" title="Pay Online, Send Money or Set Up a Merchant Account">...</a>
                    <a class="mv-tile" data-tid="6" href="http://indeed.com/" aria-l...
                    <a class="mv-tile" data-tid="7" href="https://www.aliexpress.com/"...
                    Fashion, Home & Garden, Toys & Sports, Automobiles and More." title="Home & Garden, Toys & Sports, Automobiles and More.">...</a>
                    <a class="mv-tile" data-tid="8" href="http://ebay.com.au/" aria-l...
                    eBay" title="Electronics, Cars, Fashion, Collectibles, Coupons and Mo...
                  </div>
                </div>
              </body>
            </html>
          </iframe>
        </div>
      </div>
    </div>
  </body>
</html>
<script nonce="ClHzRDKiOoLDWYPMvtZyyQ==">...</script>
<script src="/xjs/_/js/k=xjs.ntp.en.bwJr3A6P0-g.0/m=spch/am=gAgSIw/exm=sx,jsa,ntp.c...
  async gapi_processed="true"></script>
</body>

```

An HTML page is essentially the structure or the look of the page:



An HTML page also contains links to stylesheets (css) and code (javascript):

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Nexgen Codecamp Code & Game Design</title>
    <link rel="stylesheet" href="./css/style.css">
</head>
<body class="background">
    <h2 class="title">Pete's Invaders</h2>

    <div id="game">
        <!-- this is the game area -->
    </div>

    <!-- Add Crafty JS -->
    <script src="./js/crafty-min.js"></script>

    <!-- Add Utility functions -->
    <script src="./js/utils/utils.js"></script>

    <!-- Add Components -->
    <script src="./js/components/Starfield.js"></script>
    <script src="./js/components/Player.js"></script>
    <script src="./js/components/EnemySimple.js"></script>
    <script src="./js/components/Bullet.js"></script>
    <script src="./js/components/Explosion.js"></script>

    <!-- Add our game code !!! -->
    <script src="./js/main-06.js"></script>
</body>
</html>
```

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive

Mac/Windows/
Linux

**System built
with Objects**

**Mouse over,
Click,
Keypress**

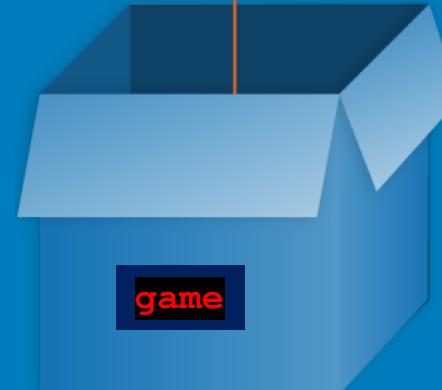
Variables (containers)

Variables are used to store data/information such as **numbers** or **characters**
Variables have **names** and **values**

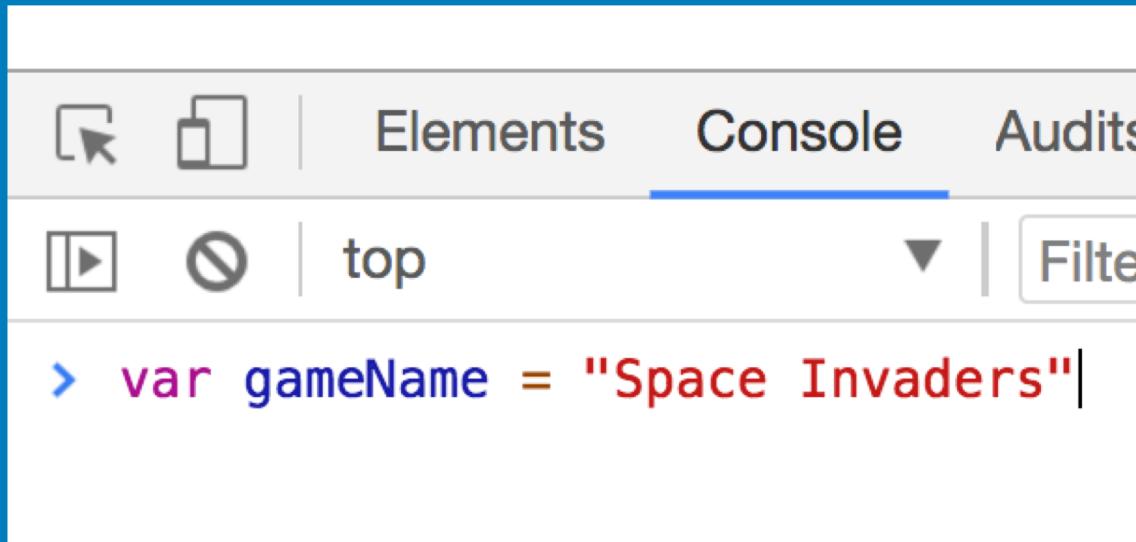
```
var score = 1000
```



```
var game = "Pacman"
```

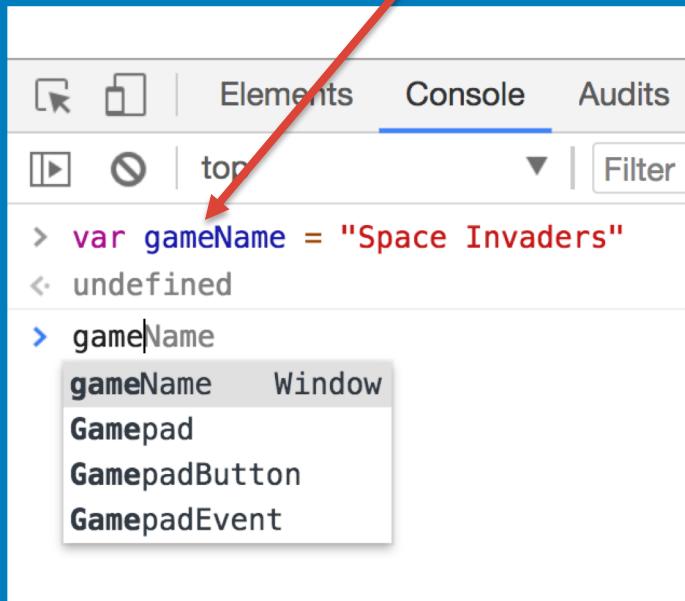


1. Open Google Chrome
2. Use Ctrl+Shift+I (or Cmd+Opt+I on Mac) to open the DevTools.
3. Click on the 'Console' tab
4. Create a variable of your choice and hit enter to add it to Chrome



The screenshot shows the Google Chrome DevTools interface with the 'Console' tab selected. The top navigation bar includes icons for selection, inspection, Elements, Console (which is underlined in blue), and Audits. Below the navigation bar, there are two large buttons: one with a play icon and another with a stop icon. To the right of these buttons is the text 'top'. Further to the right is a dropdown arrow and a 'Filter' input field. At the bottom of the console area, a blue arrow points right, followed by the code: `> var gameName = "Space Invaders";`

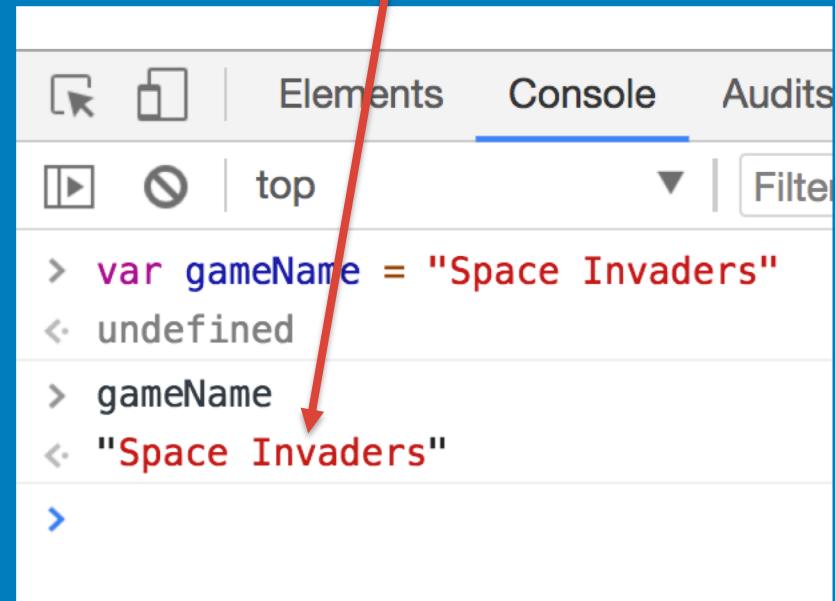
1. Type in the variable **name**. You will see that Chrome has it stored
2. Hit the Enter key. Chrome will display the variable **value**



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. In the console, the following code is entered and executed:

```
> var gameName = "Space Invaders"
<- undefined
> gameName
gameName      Window
Gamepad
GamepadButton
GamepadEvent
```

A red arrow points from the word 'name' in the first line of code to the word 'name' in the third line of output.



The screenshot shows the same DevTools interface with the 'Console' tab selected. The previous code is still present, but now the variable 'gameName' is explicitly expanded to show its value:

```
> var gameName = "Space Invaders"
<- undefined
> gameName
<- "Space Invaders"
>
```

A red arrow points from the word 'value' in the second list item of the slide text to the value 'Space Invaders' in the console output.

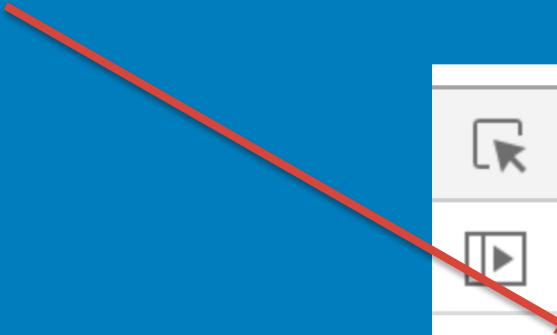
Function (actions)

The diagram shows a black rectangular box containing a function definition. Four white speech bubbles point from the text to specific parts of the code:

- Function keyword**: Points to the word "function".
- Function name**: Points to the identifier "add".
- Function parameter**: Points to the parameters "n1" and "n2".
- Function code/body**: Points to the "return" statement and the expression "n1 + n2".

```
function add ( n1, n2 ) {  
    return n1 + n2;  
}
```

Create the 'add' function in Chrome



```
Elements      Console
top
> function add(n1,n2){
      return n1+n2;
}
```

Once created, call the function, passing two numbers to add:

add(2, 3)

HEX Colors

#BE0712

#1E9C5A

#FFFD38

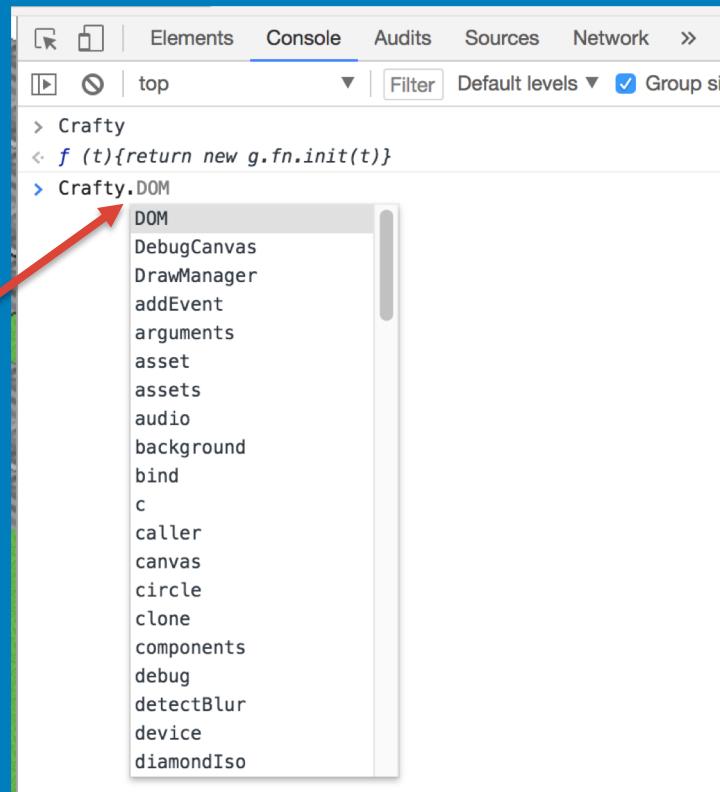
#6F359E

#**9** **7** **0** **5** **1** **5**

Red Green Blue

Introducing the 'Crafty' Object

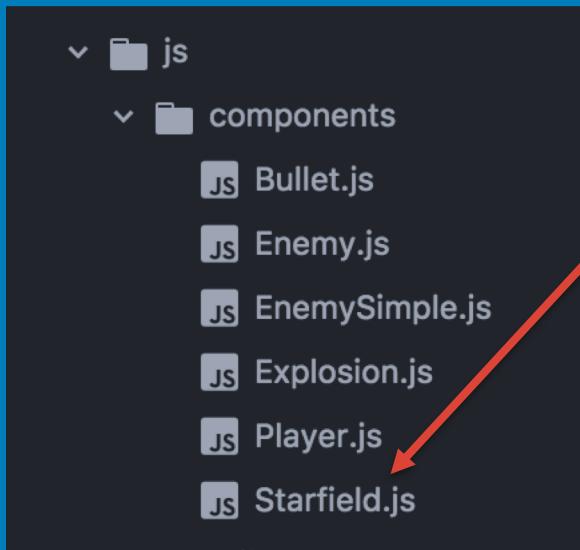
- Crafty JS is a Javascript game library (engine) that helps us to build games in Javascript
- To use it we may add the crafty.min.js file to our HTML page with a <script> tag
- Type 'Crafty' in Chrome Dev Tools, console tab and then . (dot)



Lesson 2 – Creating a Starfield

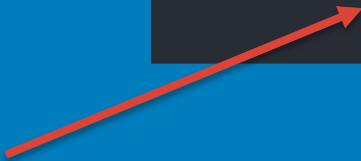
Creating a Starfield Object

The Starfield code is found in js/components/Starfield.js



We use the 'new' keyword to create new Javascript Objects:

```
// Create the Starfield  
_starField = new Starfield(game);
```



Once, created, **_starField** can now be used to create a moving star Field in the game. You can have Different numbers of stars running at different speeds

The Starfield function takes 1 parameter (game)

```
// Start the __starField  
__starField.start();
```

To call a method on an object, we use the dot notation

Looking at the code for the Starfield in Starfield.js, we can see that we could alter the velocity, size and number of stars quite easily.

```
1 function Starfield(div) {  
2     this.fps = 30;  
3     this.canvas = null;  
4     this.width = 0;  
5     this.width = 0;  
6     this.minVelocity = 5;  
7     this.maxVelocity = 40;  
8     this.stars = 200;  
9     this.maxSize = 2;  
0     this.intervalId = 0;  
1  
2     this.initialise(div);  
3 }  
4
```

Lesson 3 – Adding a Player

What is a Sprite ?

sprite

/sprɪt/ 🔊

noun

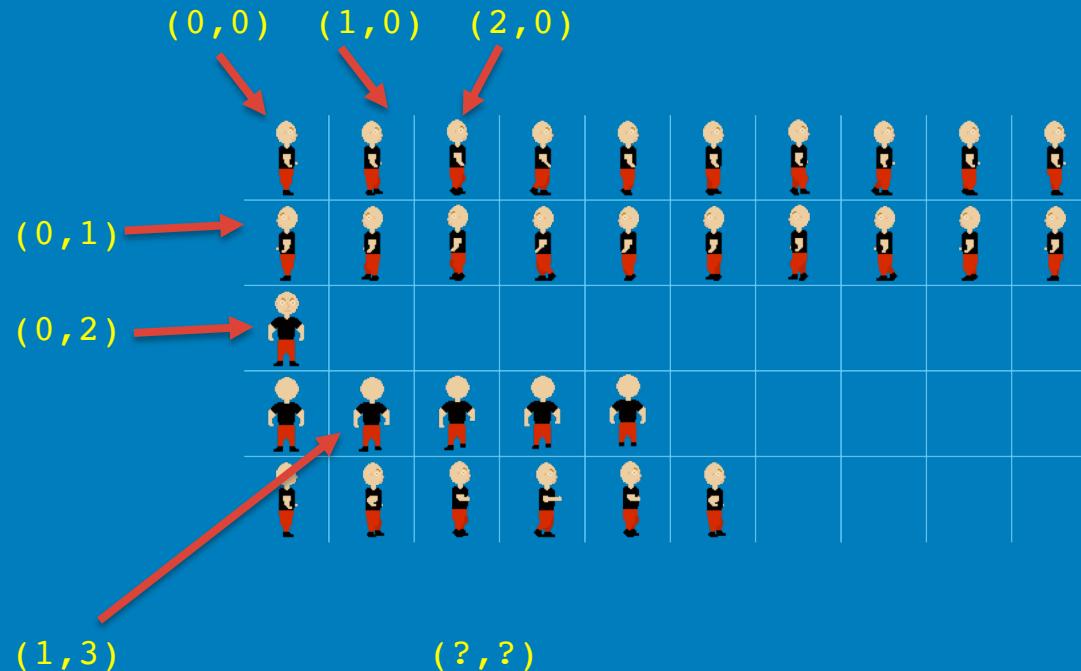
1. an elf or fairy.

synonyms: [fairy](#), [elf](#), [pixie](#), [imp](#), [brownie](#), [puck](#), [peri](#), [goblin](#), [hobgoblin](#), [kelpie](#), [leprechaun](#); [More](#)

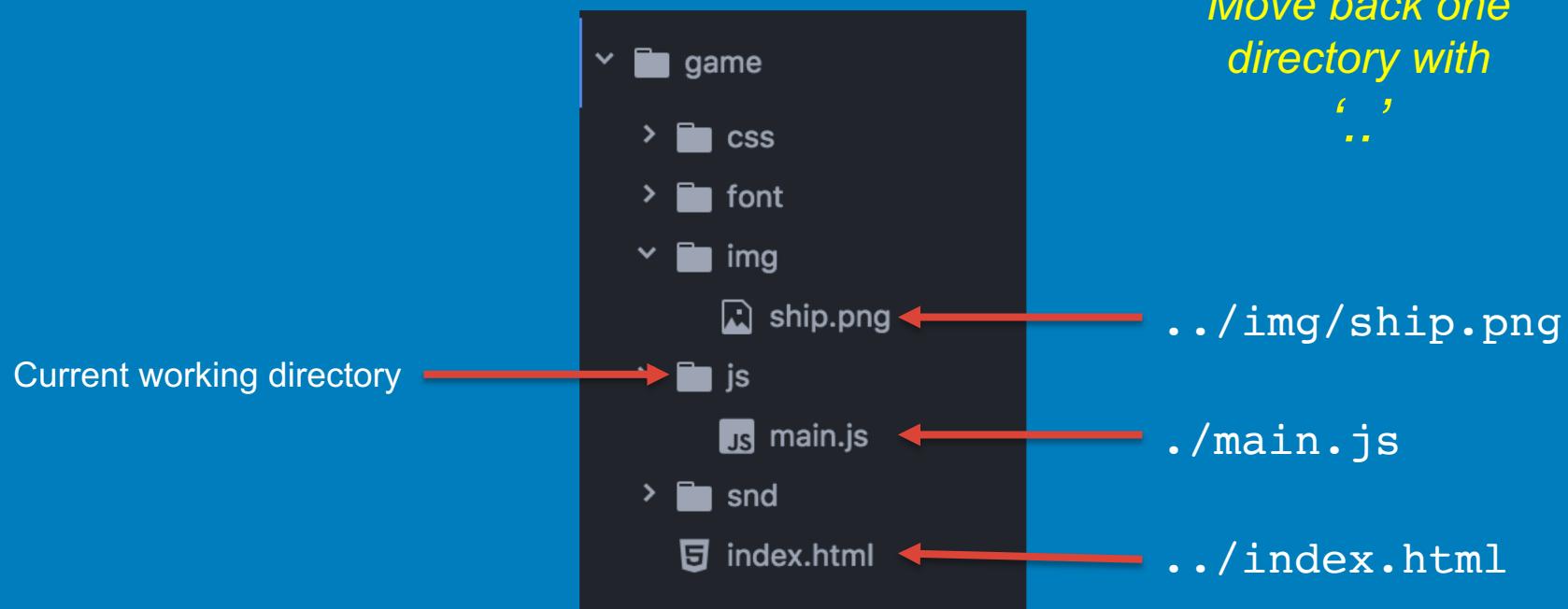
2. a computer graphic which may be moved on-screen and otherwise manipulated as a single entity.



Sprites on a Grid



Relative Directories



Create a Sprite with Crafty

```
Call sprite() method
Crafty Object →
name of sprite
/**** SPRITES ****/
Crafty.sprite("../img/ship.png", {
    ship: [0, 0, 32, 34]
});
```

path to ship image, ship.png

width, height

col/row of sprite map

Entities in Crafty

An entity is some *thing* in your game. If that sounds vague, it's because practically anything can be an entity -- the player, the enemies, a projectile, a high score counter, or a menu item.

Entities are made up of components, which you can think of as bundles of functionality. Crafty provides several built-in components, and you can also define your own with [Crafty.c\(\)](#).

Creating an Entity:

You create an entity with [Crafty.e\(\)](#). In most cases you'll pass in a list of the components you want to start with:

```
var square = Crafty.e("2D, Canvas, Color");
```

Components in Crafty

```
var __player = Crafty.e("2D, Canvas, Color, Player");
```

Position/2D component

Canvas component

Color component

Player component

The entity that you have
created in a variable called
'square'

Crafty has a lot of built-in components, but creating
your own is a good way to organize your game.

Player.js
file



```
1 // Create Player Component
2 Crafty.c("Player", {
3     required: "2D, Canvas, Twoway, SpriteAnimation, Collision, ship",
4     init: function() {
5         // Put initialisation code, eg. position
6     },
7
8
9     bindEvents: function(that) {
10        that.bind("KeyDown", function(e) {
11            // Respond to a Key pressed down
12        }).bind("EnterFrame", function(e) {
13            // Do something every frame of animation
14        }).bind('EnemyHit', function(data){
15            // Do something when you have hit an Enemy, eg. add to score
16        }).onHit('EnemySimple', function(o) {
17            // Do something when player is hit, eg. lose a life
18        });
19
20
21        fire: function() {
22            // Fire a bullet code!
23        },
24
25        talk: function() {
26            // Write to console when the component is ready
27            Crafty.log("Player 1 Ready!")
28        }
29    })
30}
```

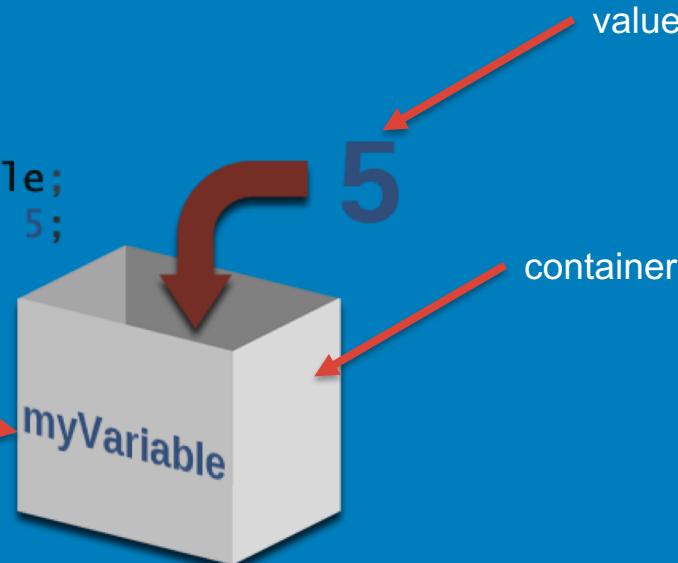
Lesson 4 – Variables & an Enemy

Variable Scope

Recap: What is a variable ?

code → `int myVariable;
myVariable = 5;`

name



Variable Scope

```
function testVariableScope(){  
    var myVar = "I am here!";  
}
```



Create a variable inside a function in Chrome Dev tools.

Type the variable name and then ENTER

```
> myVar  
✖ ►Uncaught ReferenceError: VM732:1  
myVar is not defined  
at <anonymous>:1:1  
>
```

What just happened and Why?

Variable Scope

```
> myVar = "I am here!"  
< "I am here!"
```

This time just type out the variable but not in a function

```
> myVar  
< "I am here!"
```

Again, type the variable on the Chrome Dev tools command line and press ENTER

Variable Scope

main.js

```
var anotherVar = "I am elsewhere";
```

```
function test() {  
    var myVar = "I am here";  
}
```

The **scope** of this variable is **global**. It can be accessed from anywhere inside the program.

The **scope** of this variable is **local** to the function. That is, it can only be accessed from code that sits inside the function.

Lesson 5 – Adding Component Behaviour

Create a Component

Create a Crafty component entity

```
Crafty.c("EnemySimple", {  
    required: "2D, Canvas, SpriteAnimation, Collision, Gravity bee",  
    /* This function will be called when the component is added to an entity */  
    init: function() { ...  
  
    afterInit: function(props){ ...  
  
    bindEvents: function(self) { ...  
  
    talk: function() { ...  
  
    fire: function() { ...  
})  
}
```

Give the component Gravity

Components give you power!

```
afterInit: function(props){  
    this.x = props.x;  
    this.y = props.y;  
    this.reel('EnemySimpleMove', 1000, [[0, 0], [1, 0]]);  
    this.animate('EnemySimpleMove', -1);  
    this.gravity();  
    this.gravityConst(props.speed);  
},
```

Adding the Gravity component, gives your component/entity extra functions. You can now set the gravity speed.

Lesson 6 – Multiple enemies!

Let's say I had a character thus:



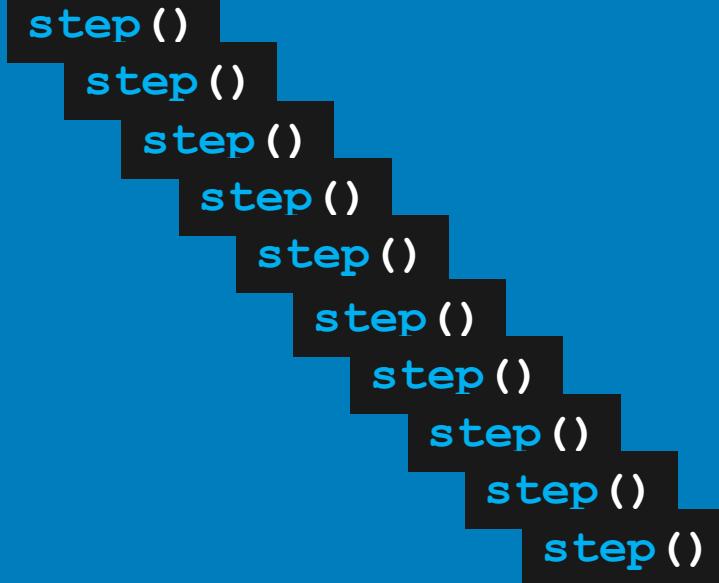
...and a function

`function step()`

... that makes the character take **one** step.

... What if you wanted him to take 10 steps?

You could call the function 10 times...?

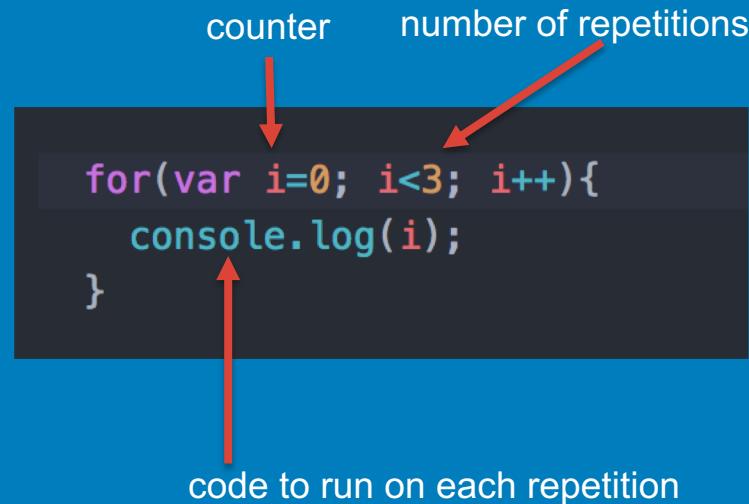


Is that a good idea?

What if you wanted to take 1000 steps?

For....Loops

Another way of repeating an instruction!



Lesson 7 – Events

Events in a Game

Level complete

Enemy is killed

Player loses a life

Game started

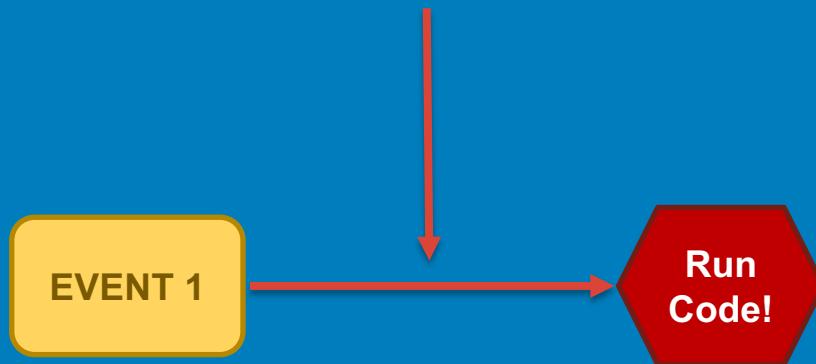
End of game

High score reached

Score updated

Events in a Game

When an EVENT occurs, we *may* want to run some Javascript code!



Events in a Game

To **setup** an event in Crafty, we must '**bind**' it

```
***** EVENTS *****
Crafty.bind('player_fired', function(player) {
    player.fire();
});
```

The name of the event

The function that is run when
the event occurs

Events in a Game

To **run** an event in Crafty, we must **trigger** it

Data that is passed into the
'player_fired' event code

```
Crafty.trigger('player_fired', obj);
```

Trigger the 'player_fired' event