



# JavaScript and Phaser

Jump into the world of code!



## Today we'll learn:

- The origins of HTML and JavaScript
- How to set up an HTML and JavaScript page
- JavaScript 101 - the basics
- How to set up Phaser
- The basics of Phaser



# The Origins of HTML and JavaScript



# What is HTML?

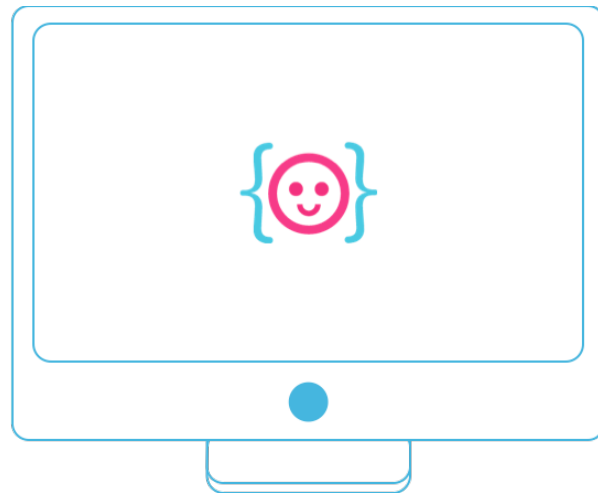
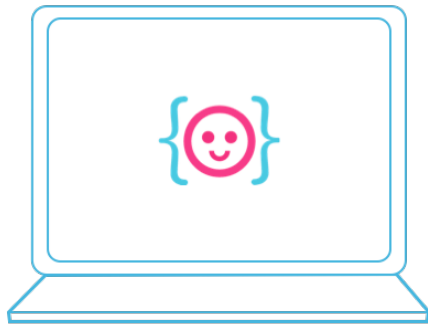
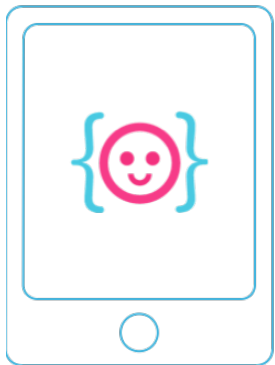
HTML is a **Markup language** (code-based annotation system)

as it enables its members to insert spaces among canonical (i.e. predetermined) practices in which to develop non-canonical views – that is, ones richer and more flexible and subject to constant change. Within these spaces, there develops and is preserved a situated knowledge which becomes a collective asset and the source of idiosyncratic power. Brown and Duguid's contribution has given rise to a set of studies, still relatively little developed, that seek to understand innovating as a



# What is HTML5?

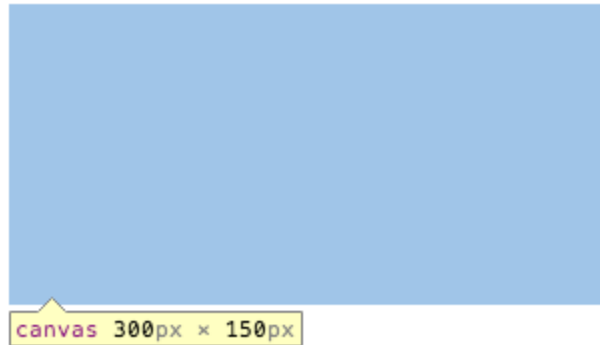
The 5th version of HTML, which was last updated in 1997. It was designed for all of our modern devices.





# What is <canvas>?

An HTML5 tag that allows you to draw things in your browser using JavaScript.



It's a box that can make anything happen!



# What is JavaScript?

- A 19-year-old programming language that is mainly used on the web.
- Allows dynamic interaction and effects to happen based on conditions and events.



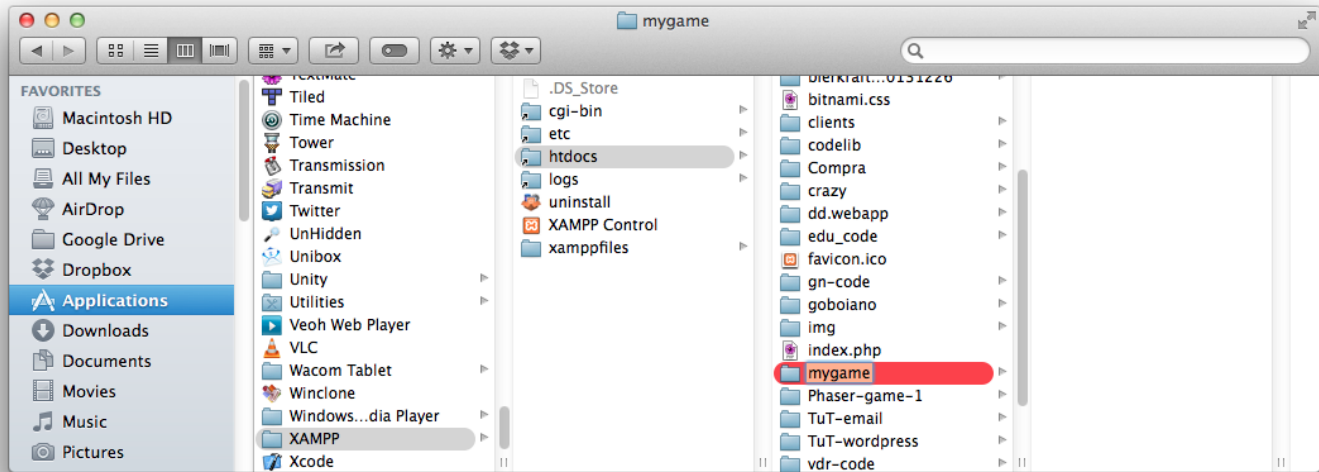
# Preparing to code in JavaScript





# Create a new folder

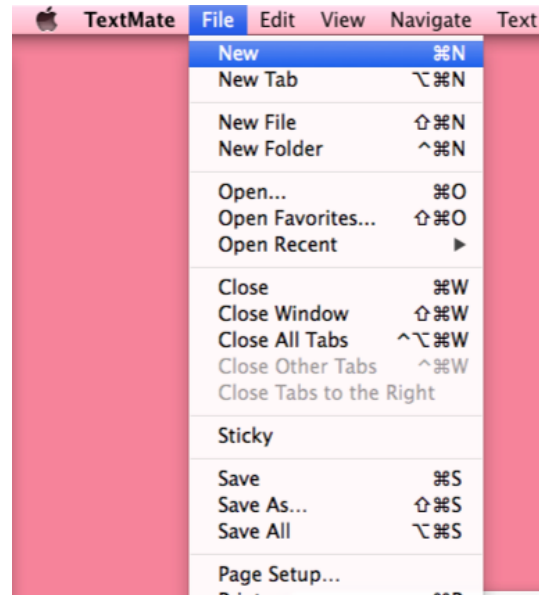
Put the folder anywhere you'd like.





# Create a new file in your text editor

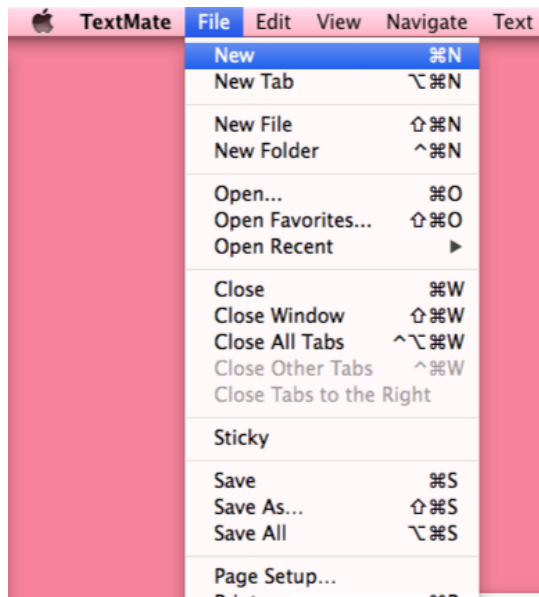
Name it **index.html**.





# Create another file in your text editor

Name it **scripts.js**.





# Create a simple HTML5 layout

The bare minimum to get your HTML page working:

```
<!doctype html>
<html>
  <head>
    <title>My HTML Page</title>
  </head>
  <body></body>
</html>
```



## Link your JavaScript file

Use the **<script>** tag to tell HTML to listen to your JavaScript file.

```
<body>
```

```
    <script src="scripts.js"></script>
```

```
</body>
```



# Basic concepts of JavaScript



# Structure of JavaScript

- Each command requires ; at the end.
- What happens inside {} stays inside. This rule is called the **scope**.
- You can use “ ” or ‘ ’. JavaScript doesn’t care, as long as you’re consistent!
- JavaScript can be finicky about spacing.



# Comments

Use comments to leave notes in your code or troubleshoot issues.

- one-line comment: `// your comment`
- multi-line comment: `/* your comment */`





# The Console

The console allows you to test your code.

Type:

```
console.log("Hello, world!");
```

View **Element Inspector's** console in your browser.



# Variables

Useful for storing data that may change or be referenced throughout the course of your game.

For example, your score:

```
var score = 1;  
score = 2;
```



# Types of Variables

Unlike some other languages that require you to state the variable data type, JavaScript does not. Keep track of your variable's type!

```
var message = "Hello, World!";  
message = 1; nope!  
message = true; nope!  
message = "Welcome to the internet!"; yeah!
```



# Types of Variables

## Numbers

```
var score = 1;
```

## Strings

```
var message = "Hello, World!";
```

## Booleans

```
var isAlive = true;
```



# Functions

A group of code that performs a specific task.

```
var doMath = function (variable) {  
    variable += 1;  
    console.log(variable);  
};  
  
doMath(score);
```



# Functions

You can declare a function in two ways:

```
var doMath = function(variable) {  
  ;  
};
```

```
function doMath(variable) {  
  ;  
};
```



# Conditional Statements

Perform a task if something is true or false.

```
var testScore = function (variable) {  
    if (variable !== 10) {  
        variable += 1;  
        console.log("Not 10 yet! You're at " + variable);  
    }  
};  
testScore(score);
```



# For Loops

Actions that happen until the set condition is false.

```
var addNumbers = function (variable) {  
    for(i=0; i < 10; i++) {  
        variable++;  
    }  
};  
  
addNumbers(score);
```





# Object Variables

Stores sets of data in one variable.

```
var Catt = {  
    height: 164,  
    age: 24,  
    occupation: "Product Designer"  
}
```



# Object Variables

Uses **dot notation** to reference and/or define properties.

```
Catt.hairColor = "dark brown";  
Catt.hasPets = true;
```



# Array Variables

Stores sets of data in numbered list form.

```
var inventory = [ "sword", "potion" ];
```

Access and modify array items with **brackets**.

```
inventory[2] = "crescent moon wand";
```



# Math

- Addition: +
- Subtraction: -
- Multiplication: \*
- Division: /

```
console.log( score + 1 );
```



## Complicated Math

Use parentheses to do complicated formulas without having to remember how PEMDAS works.

```
score = ((51/43) + (61*48)) - 5;
```



# Math's Random Function

The **Math.random()** function allows you to find a random number between 0 and 1.

```
score = Math.random();
```



## Random numbers above 1

Want a random number over 1? Multiply it by 100.

```
score = Math.random() * 100;
```



## Random Integers

If you don't want a decimal number, use the **Math.floor()** function, which rounds down to the nearest integer.

```
score = Math.floor(Math.random()) ;
```





# Phaser



# What is a framework?

- Frameworks help to reduce the amount of time spent reinventing the wheel.
- They come with a large set of tools to help you accomplish tasks faster.



# What is Phaser?

Phaser is an open source JavaScript framework made for HTML5 game developers by HTML5 game developers.

WEBGL & CANVAS

PRELOADER

PHYSICS

SPRITES

GROUPS

ANIMATION

PARTICLES

CAMERA



INPUT

SOUND

TILEMAPS

DEVICE SCALING

PLUGIN SYSTEM

MOBILE BROWSER

DEVELOPER SUPPORT

BATTLE TESTED



## What is Phaser?

Phaser requires a server to run for security reasons. Local servers allow you create this experience without an internet connection.



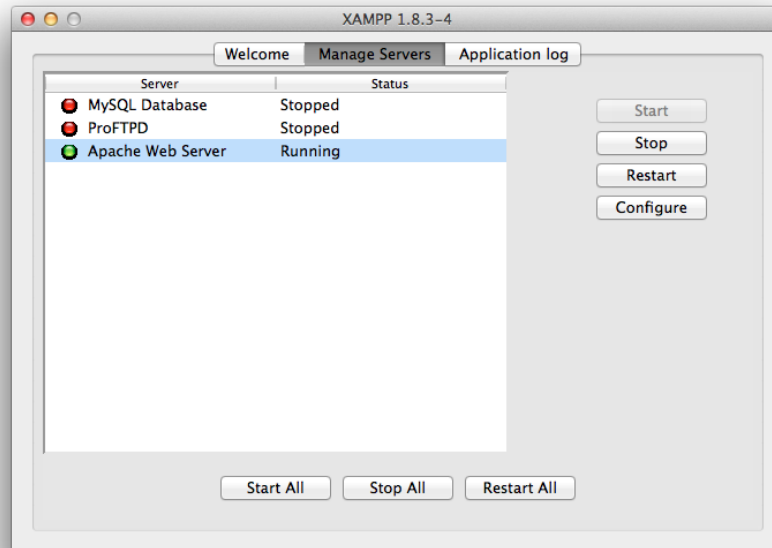


# Setting up Phaser



# Turn on your web server

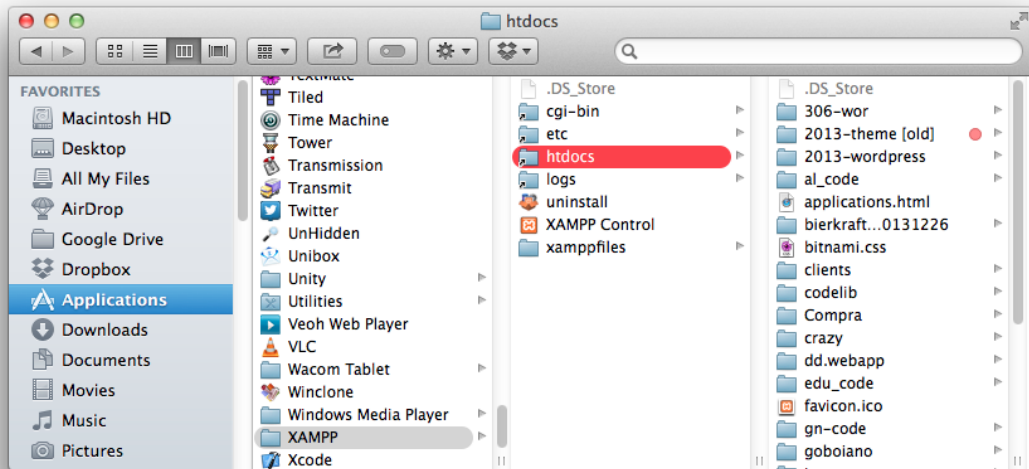
Open XAMPP and start the **Apache Web Server**.





# Find the XAMPP folder

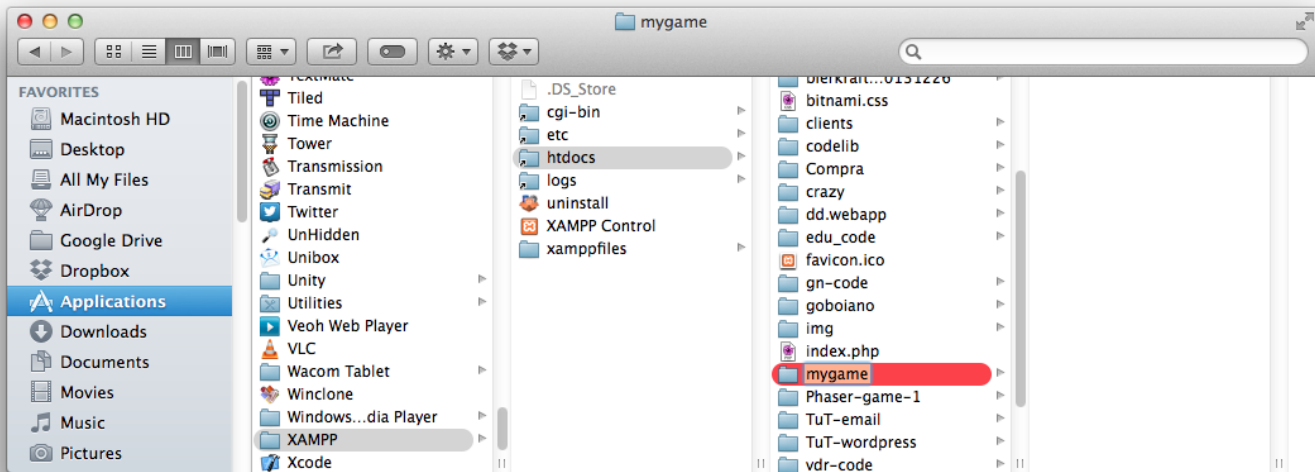
Open your XAMPP folder, then find htdocs





# Create a new folder

Put the folder anywhere you'd like.

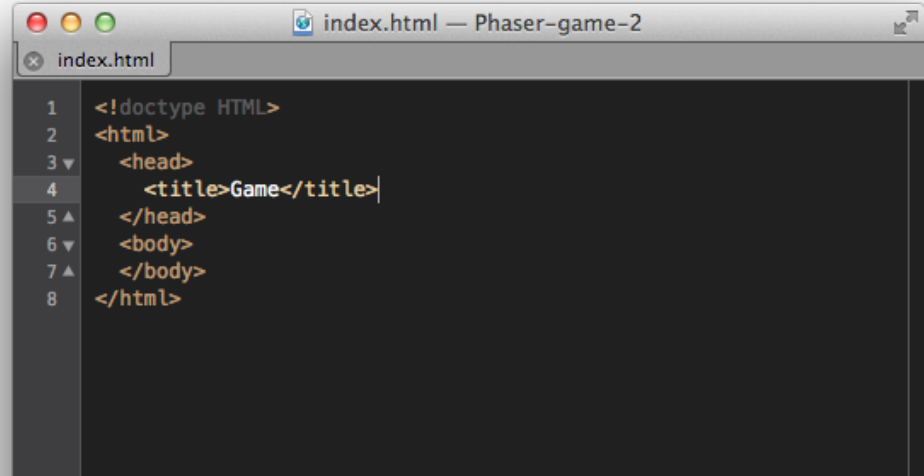
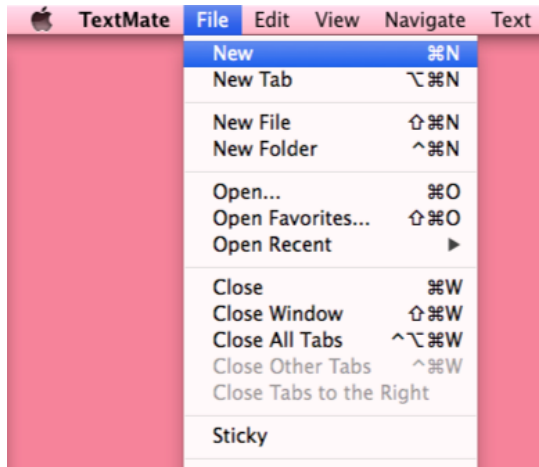






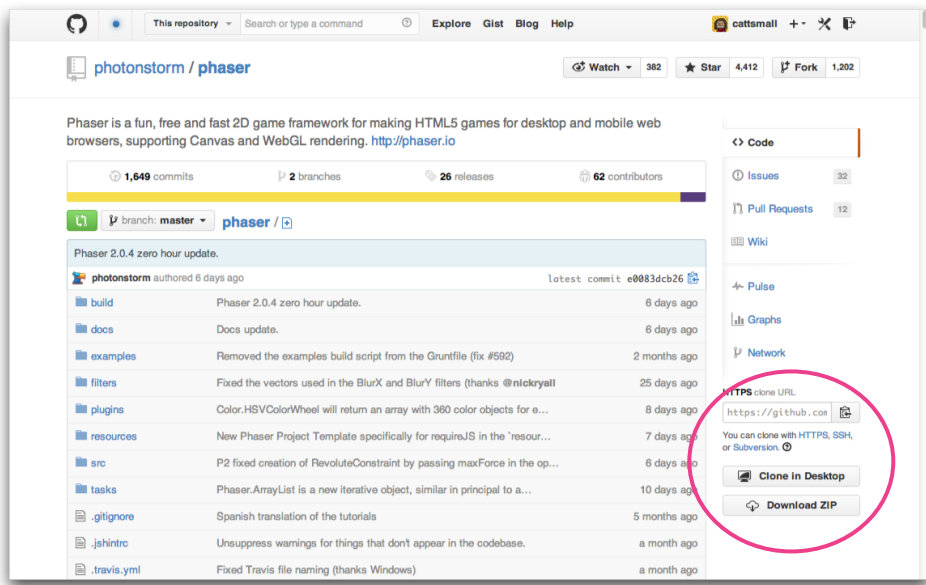
# Create a new file in your text editor

Save the new file as **index.html** in your folder. Use the same HTML5 code as before.





# Download the latest version of Phaser



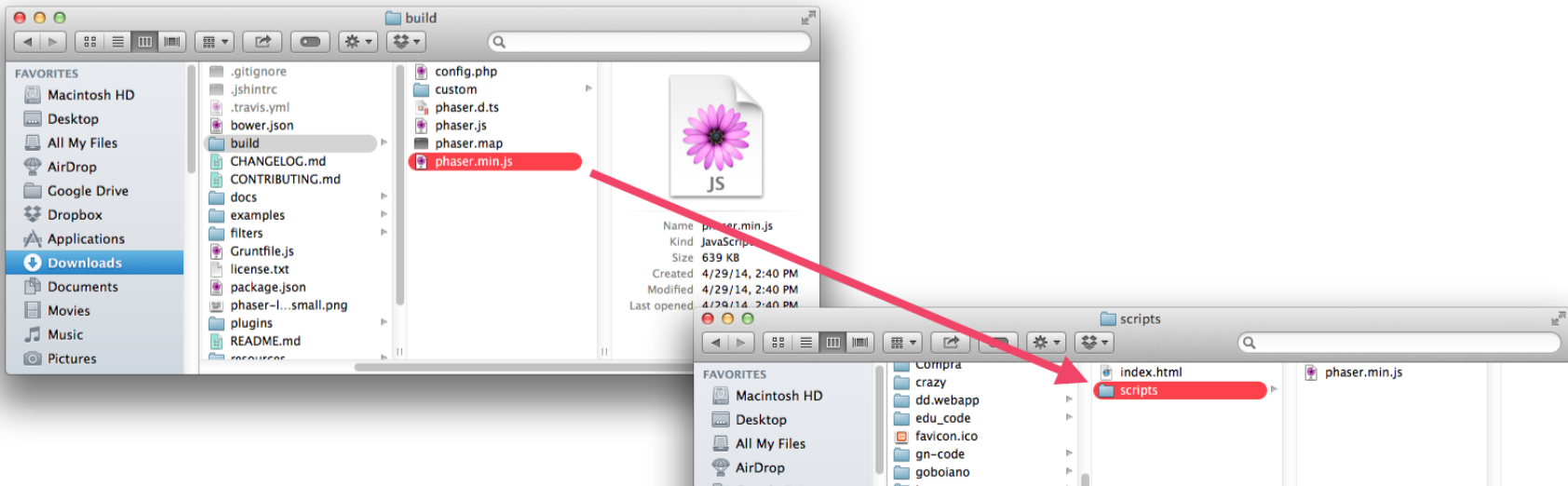
[github.com/photonstorm/phaser](https://github.com/photonstorm/phaser)

Press "Download ZIP"



# Move phaser.min.js

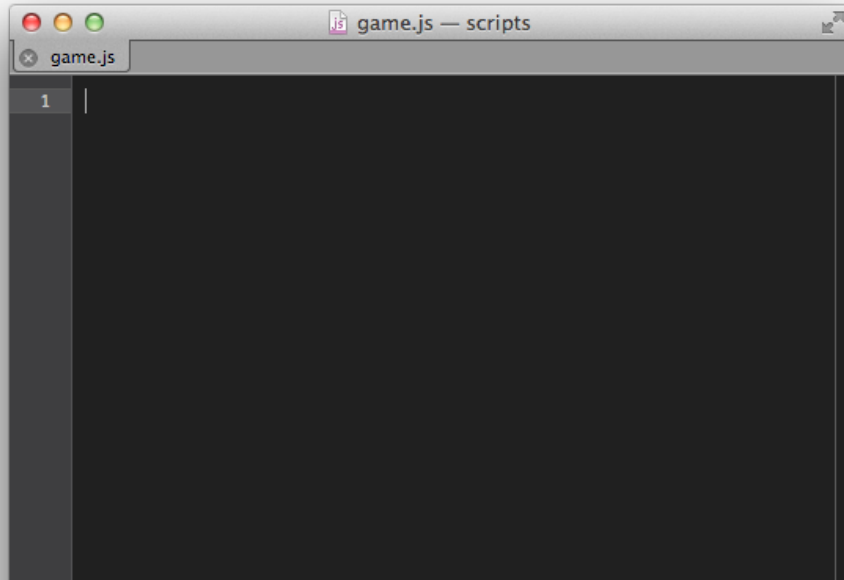
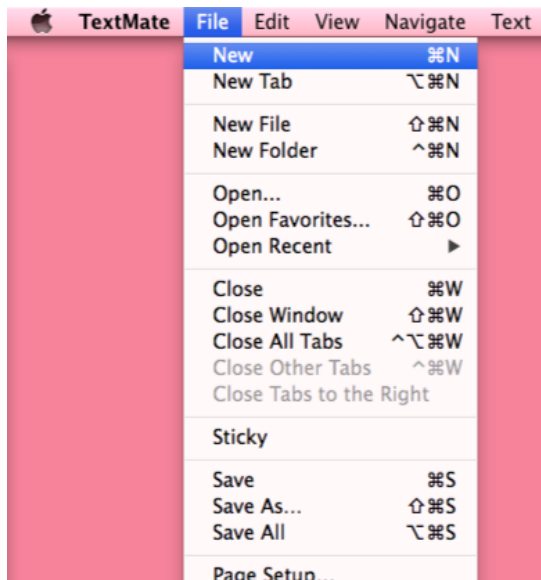
Move **phaser.min.js** from **build** to a new folder called “scripts” in your game’s directory.





# Create a file for your game's code

Save your new file as **game.js** in your **scripts** folder.





## Download placeholder art

In case you don't have your own art to work with, you can use some we found:

<http://tinyurl.com/CLF-html5-art-2014>

Art by Robert Podgórski



# Basic Phaser Concepts



# Game

An object that contains properties related to gameplay including window width, window height, and graphic rendering settings.

```
var game = new Phaser.Game(640, 960, Phaser.AUTO);
```



# States

Phaser allows your game to have different states. Some example of possible states are intro screens, gameplay levels, and win/lose screens.

```
game.state.add('GamePlay', myGame.GamePlay);  
game.state.start('GamePlay');
```





# Prototypes

Object functions need prototypes to run. In this case, prototypes outline the variables and functions within the scope of each state.

```
myGame.GamePlay.prototype = {  
  }
```



## Preloading

Phaser needs to know what images to prepare before the game can be displayed. This phase is called **the preload()** function.

```
function preload() {  
}
```



# Images

There are several types of images in Phaser:

- **image** - static, no animation
- **spritesheet** - sprite with animation
- **tilemap** - environment objects

```
this.load.image(background, 'img/background.png');
```



## Images

Sprites require widths and heights since they might have multiple animation frames. The last two numbers are the sprite's width and height.

```
this.load.spritesheet('player', 'img/player.png', 32, 64);
```



## Creating the Game

Once the preload function is complete, Phaser needs you to tell it how the game will start.

```
function create() {  
}
```



# Creating the Game

The **create()** function lets you set up variables, objects, and the look of your game.

```
function create() {  
    myGame.score = 0;  
}
```



## Updating the Game

Unlike `preload` and `create`, which only run once each, the **`update()`** function runs every millisecond.

```
function update() {  
}
```



## Updating the Game

`update()` is where your player is told to move, the score is updated, etc.

```
function update() {  
    myGame.score += 1;  
}
```





## Drawing objects

You can draw interactive objects onscreen using Phaser's **add** function.

```
myGame.character = this.add.sprite(x, y, 'charName');
```



## Animating objects

You can animate objects by adding to its **animations** property and choosing the frames that should be shown in order.

```
myGame.character.animations.add('animationName', [0, 1, 2]);
```



## Animating objects

To trigger an animation, use the play command. Name the animation you want to play, enter a **framerate**, and say whether the animation should loop (true) or not (**false**).

```
myGame.character.animations.play('animationName', 30,  
false);
```



# Physics

Phaser has a set of systems called **Physics** that allows you to easily check when objects touch.

```
this.physics.enable(player, Phaser.Physics.ARCADE);
```



# Physics

Phaser has 3 types of physics:

- **Arcade** - only checks if rectangles overlap.  
Quickest to load.
- **Ninja** - checks for slopes and rotation (curves)
- **P2** - allows you to make a full-fledged physics game with angles and swinging like Angry Birds



## Checking collision

Using Phaser's physics, you can trigger a function when two objects (or groups of objects) overlap:

```
this.physics.arcade.overlap(player, enemy, playerDies);
```



# Groups

Have an object you want to repeat onscreen and give the same properties? Make a group.

```
myGame.myGroup = this.add.group();
```



## Using Groups

Use a **for loop** or timer function to instantiate objects and add them to a group.

```
var groupItem = myGroup.create(x, y, 'spriteToUse');
```

or

```
myGame.myGroup.add(groupItem);
```





## To do:

- Make a small interactive game in Phaser.
- If possible, use your game art from last week to make the game even more awesome!



# Thanks! Questions?

@cattsmall  
catt@code liberation.org