

The Code Liberation Foundation Lecture 5: JavaScript and Phaser I

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 Code Liberation Foundation Lecture 5: JavaScript and Phaser I

The Origins of HTML and JavaScript



What is HTML?

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

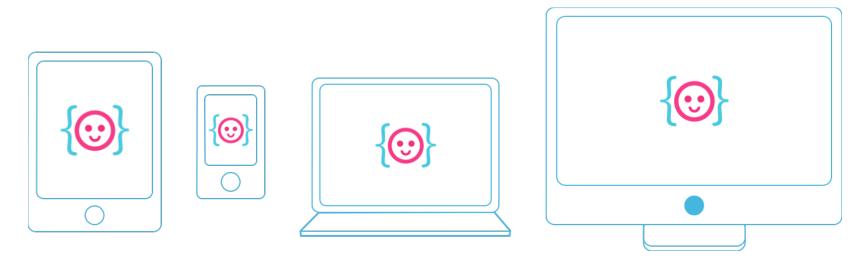
as it enables its members to insert spaces among canonical (i.e. predetermined) practices in which to develop non-canonical views – that is, ones richer and more flexible and subject to constant change. Within these spaces, there develops and is preserved a situated knowledge which becomes a collective 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.



The Code Liberation Foundation Lecture 5: JavaScript and Phaser I

Preparing to code in JavaScript



Create a new folder

Put the folder anywhere you'd like.

00		🚞 mygar	ne		M _M
		- * -		Q	
FAVORITES Macintosh HD Cosktop All My Files AirDrop Coogle Drive Dropbox Applications Documents Movies Movies Music Pictures	Image: Textmate Image: Textmate Image: Textmate Image: Textmate Textmate	 .DS_Store cgi-bin etc htdocs logs uninstall XAMPP Control xamppfiles 		 bierkraft0131220 bierkraft0131220 clients codelib Compra crazy dd.webapp edu_code favicon.ico gn-code goboiano img index.php mygame Phaser-game-1 TuT-email TuT-wordpress vdr-code 	11



Create a new file in your text editor Name it **index.html**.

	TextMate	File	Edit	View	Navigate	Text
		Ne	w w Tab		第N て第N	
			w File w Fold	er	ት የዚህ የ	
			en		80	
			en Fav en Rec	orites ent	압 ፡፡፡ ►	
		Clo			жw	
			ose Win		ዮ₩W	
				Tabs		
				er Tabs		
		Clo				
		Stie	cky			
		Sav	/e		жs	
		Sav	/e As		ዮ∺S	
		Sav	/e All		۲жS	
		Pag	ge Setu	p		



Create another file in your text editor

Name it **scripts.js**.

TextMate	File	Edit	View	Navigate	Text
	New New Tab			¥N ۲.₩N	
	New File New Folder			ጉ∺N	
		en en Eav	orites	第0 企業O	
		en Rec		•	
	Clo	se		жw	
		se Win		ዕ <mark></mark> ₩	
			Tabs		
			er Tabs s to the		
	Stic	:ky			
	Sav	/e		жs	
		e As		<u>ት</u> ዙS	
	Sav	e All		۲жS	
	Pag	ge Setu	p		



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:



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

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++;
  }
};</pre>
```

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());



The Code Liberation Foundation Lecture 5: JavaScript and Phaser I

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.





What is Phaser?

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





The Code Liberation Foundation Lecture 5: JavaScript and Phaser I

Setting up Phaser



Turn on your web server

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

_	(Welcome	Mana	age Servers	Applic	ation log 🔶	
_	Server			Status			
	MySQL Database		opped				Start
	ProFTPD		opped				Stop
0	Apache Web Serve	r Ru	inning				
							Restart
							Configure
		C 1 1		C: A!!			
		Start A		Stop All	Res	start All	



Find the XAMMP folder

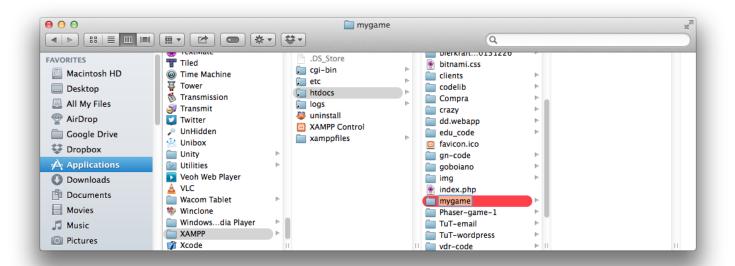
Open your XAMPP folder, then find htdocs





Create a new folder

Put the folder inside htdocs.





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.

🛒 T	extMate	File	Edit	View	Navigate	Tex
		Ne Ne	w w Tab		<mark>೫N</mark> ℃೫N	
			w File w Folde	er	☆೫N ^೫N	
		Op	en en Fav en Rec	orites ent	光O 公光O ▶	
		Clo	ose Win ose All ose Oth			
		Stie	cky			





Download the latest version of Phaser

C • This repos	itory v Search or type a command (5) Explore Gis	Blog Help 🧧 cattsmall +- 🗙	F
photonstorm /	phaser	ⓓ Watch - 362 ★ Star 4,412 ₽ Fork 1,2	202
	d fast 2D game framework for making HTML5 games		
browsers, supporting C	anvas and WebGL rendering. http://phaser.io	<> Code	_
: 1,649 commits	2 branches 📎 26 releases	62 contributors ① Issues 3:	2
		1: 1: Pull Requests	2
1 P branch: master	▼ phaser / •	ii Wiki	
Phaser 2.0.4 zero hour upo	late.		
photonstorm authored 6	days ago	latest commit e0083dcb26 🔂 😽 Pulse	
build	Phaser 2.0.4 zero hour update.	6 days ago	
docs	Docs update.	6 days ago	
examples	Removed the examples build script from the Gruntfile (fix #5	2) 2 months ago ^[J] Network	
in filters	Fixed the vectors used in the BlurX and BlurY filters (thanks	enickryall 25 days ago	
plugins	Color.HSVColorWheel will return an array with 360 color object		ê.
resources	New Phaser Project Template specifically for requireJS in the	resour 7 days ago You can clone with HTTPS, SS	SH,
src	P2 fixed creation of RevoluteConstraint by passing maxForce		
tasks	Phaser.ArrayList is a new iterative object, similar in principal)
.gitignore	Spanish translation of the tutorials	5 months ago	
Jshintrc	Unsuppress warnings for things that don't appear in the code	ase. a month ago	
.travis.yml	Fixed Travis file naming (thanks Windows)	a month ago	

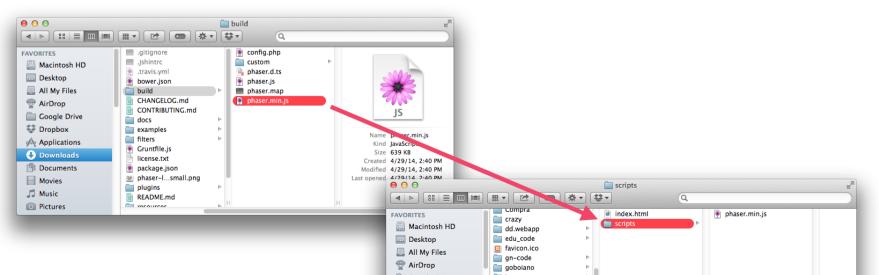
github.com/photonstorm/phaser

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.

TextMate	File	Edit	View	Navigate	Tex
	Ne			жN	
	Ne	w Tab		Σ₩N	
		w File		☆≋N	
	Ne	w Fold	er	^	
		en		жo	
				ዕ <mark></mark> װר	
		en Rec	ent	•	
	Clo			¥W www	
		ose Win	Tabs	ひ第ひ W第プへ	
			er Tabs		
	Clo	ose Tab	s to the	Right	
	Sti	cky			
	Sav	/e		жs	
		/e As		ሰ ዙS	
		/e All		۲Ħ۲	
	Pac	ae Setu	n		

1			



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



The Code Liberation Foundation Lecture 5: JavaScript and Phaser I

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');



Adding interactivity

You can add interactivity to your game using a variety of **input** types.

- this.input.mouse.x finds the X location of the mouse
- var cursors = game.input.keyboard.createCursorKeys() creates an object that contains hotkeys for the **up**, **left**, **right**, and **down** arrows.
 - cursors.left.isDown checks if the left key is down.



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!

The Code Liberation Foundation Lecture 5: JavaScript and Phaser I



Thanks! Questions?

@cattsmall
catt@codeliberation.org