

Let's do a brief recap about the JavaScript we learned in the pre-work.



We will see a lot of parallels with what we learned in Ruby.



But the *syntax* in JavaScript is different.



Let's look at some general differences between JavaScript and Ruby.



First, Ruby and JavaScript run in different environments.



We have greater control of where our Ruby runs.



It's either going to be on our computer or a server we set up.



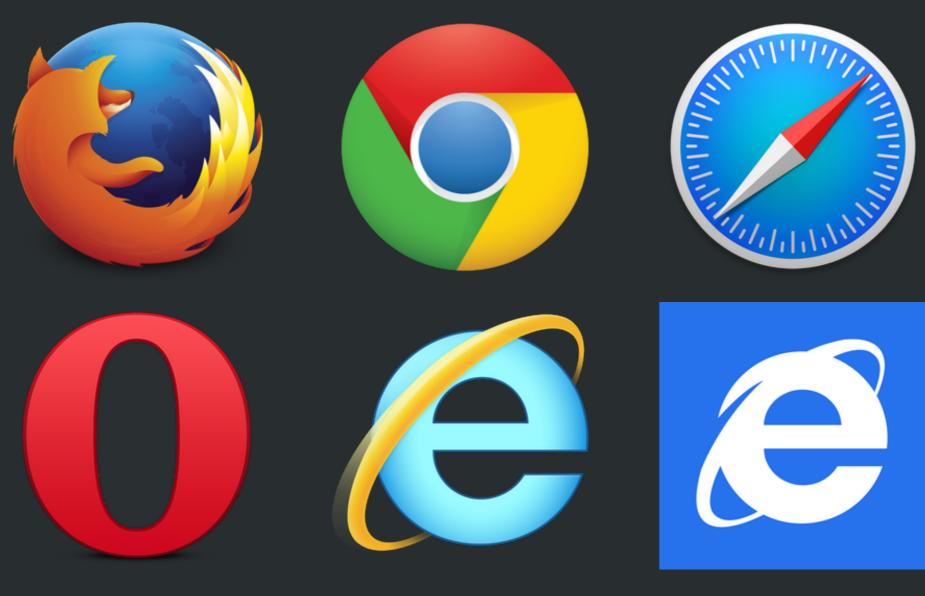
JavaScript runs on browsers.



That can be any browser your users access your Web application with.



And there are *so* many browsers and versions.





Ruby vs. JavaScript Some greater than others...





In JavaScript functions are the star of the show.



In Ruby your programs are composed of classes and objects.



In JavaScript functions are of greater importance.



There is no end keyword.



You open and close sections of code with curly braces { }.



Instructions need to end with a semicolon;



Although often if you forget the semicolon it will work anyway.



Be that as it may, try your best to not leave semicolons out.



Parentheses () are not optional.



Now let's get into the syntax.



Printing

To print things we use the console.log function.



Comments

Use // for comments.



Variables

Use the var keyword to declare variables.



Variables

They only need to be declared once.



Variables

After that you can just use them.



Strings

Double quotes " " and single quotes ' ' are exactly the same in JavaScript.



Conditions

Conditions pretty much work the same in JavaScript.



Conditions

The only difference is that you should use === for equal comparisons.



if.else

The if..else blocks in JavaScript use curly braces instead of end.



if.else

The if..else blocks in JavaScript use curly braces instead of end.

```
if (food === 'pizza') {
    console.log('Oh dear lord, pizza.');
} else if (food === 'cookies') {
    console.log('Mmmm cookies.');
} else {
    console.log('What the hell...');
}
```



To define a function you use the function keyword.



To define a function you use the function keyword.

```
function eat (food) {
    console.log('Eating some ' + food);
}
eat('pizza');
```



Named functions are also values that can be passed around.

```
function eat (food) {
    console.log('Eating some ' + food);
}
console.log(eat);
```



But we'll talk more about that in a future session.



You can use a for to loop in your program.



You can use a for to loop in your program.

```
var i = 0;

for (i = 1; i <= 42; i += 1) {
    console.log(i);
}</pre>
```



There are also array methods for looping similar to those in Ruby's Enumerable.



Except that JavaScript's versions use functions instead of blocks.



Here's the basic .forEach (instead of Ruby's .each).

```
var foods = [ 'pizza', 'cookies', 'bread' ];
foods.forEach(function (food) {
    console.log(food);
});
```



We've also got . map.

```
var foods = [ 'pizza', 'cookies', 'bread' ];
var capsFoods = foods.map(function (food) {
    return food.toUpperCase();
});
console.log(capsFoods);
```



And .reduce.

```
var foods = [ 'pizza', 'cookies', 'bread' ];
var msg = foods.reduce(function (pre, food) {
    return pre + ' AND ' + food;
});
console.log(msg);
```



And .filter (instead of Ruby's . select).

```
var foods = [ 'pizza', 'cookies', 'bread' ];
var best = foods.filter(function (food) {
    return food !== 'bread';
});
console.log(best);
```



Generic objects in JavaScript are like hashes in Ruby.



Generic objects in JavaScript are like hashes in Ruby.

```
var obj = {
  food: 'pizza',
  amount: 9999
};
console.log(obj['food']);
```



But using dot syntax is preferred (as if the key was a method).

```
console.log(obj['food']);
```

```
console.log(obj.food);
```



In another session we will see that generic objects and instances are really the same thing.



Documentation

There is no official JavaScript documentation.



Documentation

But Mozilla has your back.

developer.mozilla.org/en-US/docs/Web/JavaScript



Exercise

Write a function that receives a string of numbers separated by colons. Have the function turn that string into an array of numbers and calculate their average.

```
var numbers = '80:70:90:100';
averageColon(numbers);
//=> 85
```

