JavaScript Recap

JavaScript Recap

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

JavaScript Recap

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

JavaScript Recap

But the *syntax* in JavaScript is different.

Ruby vs. JavaScript

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

Ruby vs. JavaScript

First, Ruby and JavaScript run in different environments.

Ruby vs. JavaScript

We have greater control of where our Ruby runs.

Ruby vs. JavaScript

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

Ruby vs. JavaScript

JavaScript runs on browsers.

Ruby vs. JavaScript

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

Ruby vs. JavaScript

And there are *so* many browsers and versions.

Ruby vs. JavaScript

Some greater than others...

Ruby vs. JavaScript

In JavaScript functions are the star of the show.

Ruby vs. JavaScript

In Ruby your programs are composed of classes and objects.

Ruby vs. JavaScript

In JavaScript functions are of greater importance.

Ruby vs. JavaScript

There is no `end` keyword.

Ruby vs. JavaScript

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

Ruby vs. JavaScript

Instructions need to end with a semicolon `;`

Ruby vs. JavaScript

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

Ruby vs. JavaScript

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

Ruby vs. JavaScript

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...');
}
```

Functions

To define a function you use the `function` keyword.

Functions

To define a function you use the `function` keyword.

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

eat('pizza');
```

Functions

Named functions are also values that can be passed around.

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

console.log(eat);
```

Functions

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

Loops

You can use a `for` to loop in your program.

Loops

You can use a `for` to loop in your program.

```
var i = 0;

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

Loops

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

Loops

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

Loops

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

```
var foods = [ 'pizza', 'cookies', 'bread' ];

foods.forEach(function (food) {
    console.log(food);
});
```

Loops

We've also got `.map`.

```
var foods = [ 'pizza', 'cookies', 'bread' ];

var capsFoods = foods.map(function (food) {
    return food.toUpperCase();
});

console.log(capsFoods);
```

Loops

And `.reduce`.

```
var foods = [ 'pizza', 'cookies', 'bread' ];

var msg = foods.reduce(function (pre, food) {
    return pre + ' AND ' + food;
});

console.log(msg);
```

Loops

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

```
var foods = [ 'pizza', 'cookies', 'bread' ];

var best = foods.filter(function (food) {
    return food !== 'bread';
});

console.log(best);
```

Objects

Generic objects in JavaScript are like hashes in Ruby.

Objects

Generic objects in JavaScript are like hashes in Ruby.

```
var obj = {
  food: 'pizza',
  amount: 9999
};

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

Objects

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

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

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

Objects

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
```