Intro to JavaScript

The Language of the Web

Also used for plug-in scripting (Adobe/Mozilla/Chrome), game scripting, and more.

The History of JS

- **1995:** At Netscape, Brendan Eich created "LiveScript", which gets renamed to "JavaScript".
- **1996:** Microsoft releases "JScript", a port, for IE3.
- **1997:** JavaScript was standardized in the "ECMAScript" spec.
- **2005:** "AJAX" was coined, and the web 2.0 age begins.
- **2006:**jQuery 1.0 was released.
- **2010:** Node.JS was released.
- **2012:** ECMAScript Harmony spec nearly finalized.

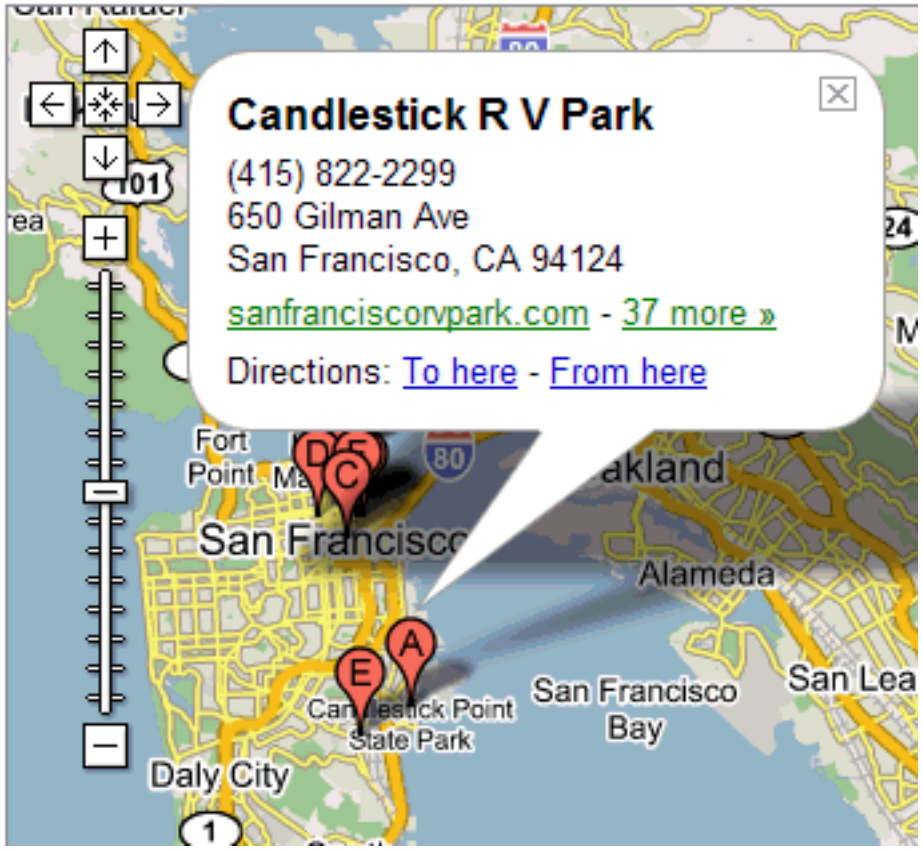
What can JS do?

Google Maps BETA

Maps Local Search Directio

candlestick park san francisco

Maps



- Image switchers and lightboxes
- Full featured web applications
- Keep track of users with cookies
- Interactive elements like tabs, sliders and accordions
- Drawing and animation
- Mind blowing awesomeness like this

Statements

Each instruction in JS is a "statement", like:

```
console.log('Hello World!');
```

Try it on repl.it:

Variables

Use variables to store values.

Declare, then initialize in 2 statements:

```
var x;
x = 5;
console.log(x);
```

Or declare and initialize in one statement:

```
var y = 2;
console.log(y);
```

Re-assign the value later:

```
var x = 5;
x = 1;
```

Primitive Data Types

- **string:** an immutable string of characters:

```
var greeting = 'Hello Kitty';
var restaurant = "Pamela's Place";
```

- **number:** whole (6, -102) or floating point (5.8737):

```
var myAge = 28;
var pi = 3.14;
```

- **boolean:** Represents logical values true or false:

```
var catsAreBest = true;
var dogsRule = false;
```

- **undefined:** Represents a value that hasn't been defined.

```
var notDefinedYet;
```

- **null:** Represents an explicitly empty value.

```
var goodPickupLines = null;
```

Variable Names

- Begin with letters, $ or _
- Only contain letters, numbers, $ and _
- Case sensitive
- Avoid reserved words
- Choose clarity and meaning

- Prefer camelCase for multipleWords (instead of under_score)
- Pick a naming convention and stick with it

  OK:

```
var numPeople, $mainHeader, _num, _Num;
```

  Not OK:

```
var 2coolForSchool, soHappy!
```

Expressions

Variables can also store the result of any "expression":

```
var x = 2 + 2;
var y = x * 3;
var name = 'Pamela';
var greeting = 'Hello ' + name;
var title = 'your highness';
var formalGreeting = greeting + ', ' + title;
```

Loose Typing

JS figures out the type based on value, and the type can change:

```
var x;
x = 2;
x = 'Hi';
```

A variable can only be of one type:

```
var y = 2 + ' cats';
console.log(typeof y);
```

Exercise Time!

Comments

Comments are human-readable text ignored by the computer:

```
// You can write single-line comments
var x = 4; // Or you can comment after a statement


/*
  Or you can write multi-line comments, if you have something very long
```

```
  to say like this gratuitously long description.
*/
```

Functions

Functions are re-usable collections of statements.

First declare the function:

```
function sayMyName() {
  console.log('Hi Pamela!');
}
```

Then call it (as many times as you want):

```
sayMyName();
```

Beware: Circular Dependancies

```
function chicken() {
    egg();
}

function egg() {
```

```
    chicken();
}


egg();
```

Arguments

Functions can accept any number of named arguments:

```
function sayMyName(name) {
  console.log('Hi, ' + name);
}

sayMyName('Pamela');
sayMyName('Testy McTesterFace');
function addNumbers(num1, num2) {
  var result = num1 + num2;
  console.log(result);
}
```

```
addNumbers(7, 21);
addNumbers(3, 10);
```

You can also pass variables:

```
var number = 10;
addNumbers(number, 2);
addNumbers(number, 4);
```

Return Values

The return keyword returns a value to whoever calls the function (and exits the function):

```
function addNumbers(num1, num2) {
  var result = num1 + num2;
  return result; // Anything after this line won't be executed
}

var sum = addNumbers(5, 2);
```

You can use function calls in expressions:

```
var biggerSum = addNumbers(2, 5) + addNumbers(3, 2);
```

You can even call functions inside function calls:

```
var hugeSum = addNumbers(addNumbers(5, 2), addNumbers(3, 7));
```

Variable Scope

JS Variables have "function scope". They are visible in the function where they're defined:

A variable with "local" scope:

```
function addNumbers(num1, num2) {
  var localResult = num1 + num2;
  console.log("The local result is: " + localResult);
}
addNumbers(5, 7);
console.log(localResult);
```

A variable with "global" scope:

```
var globalResult;
function addNumbers(num1, num2) {
  globalResult = num1 + num2;
  console.log("The global result is: " + globalResult);
}
```

```
addNumbers(5, 7);

console.log(globalResult);
```

Coding Conventions: Spacing

Use newlines between statements and use spaces to show blocks.

Bad:

```
function addNumbers(num1,num2) {return num1 + num2;}


function addNumbers(num1, num2) {

return num1 + num2;

}
```

Good:

```
function addNumbers(num1, num2) {

  return num1 + num2;

}
```

Exercise Time!

Getting Help

Google for questions or check Mozilla Developer Network andW3Schools.

Post problematic code on JSFiddle and share the link.