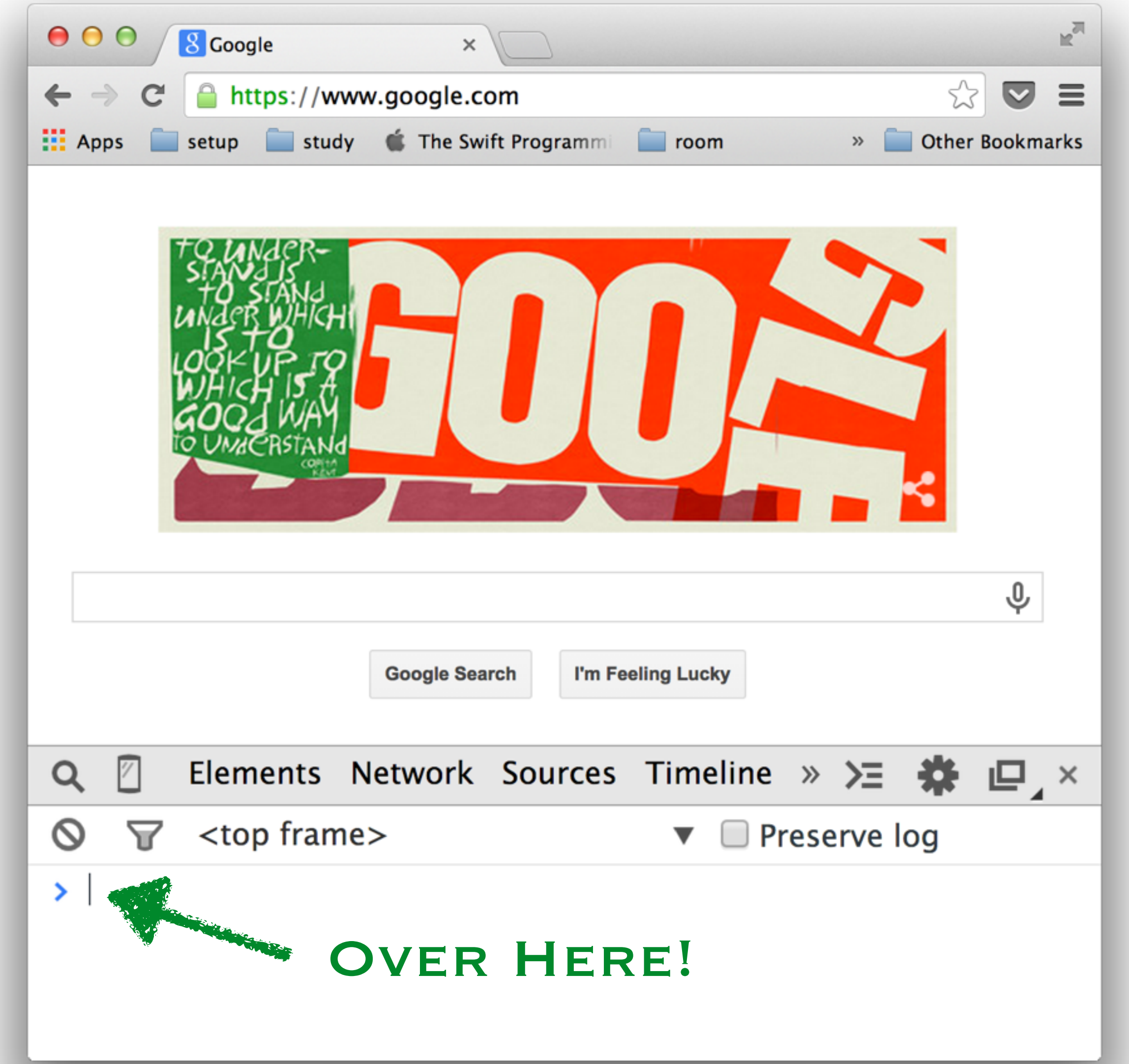
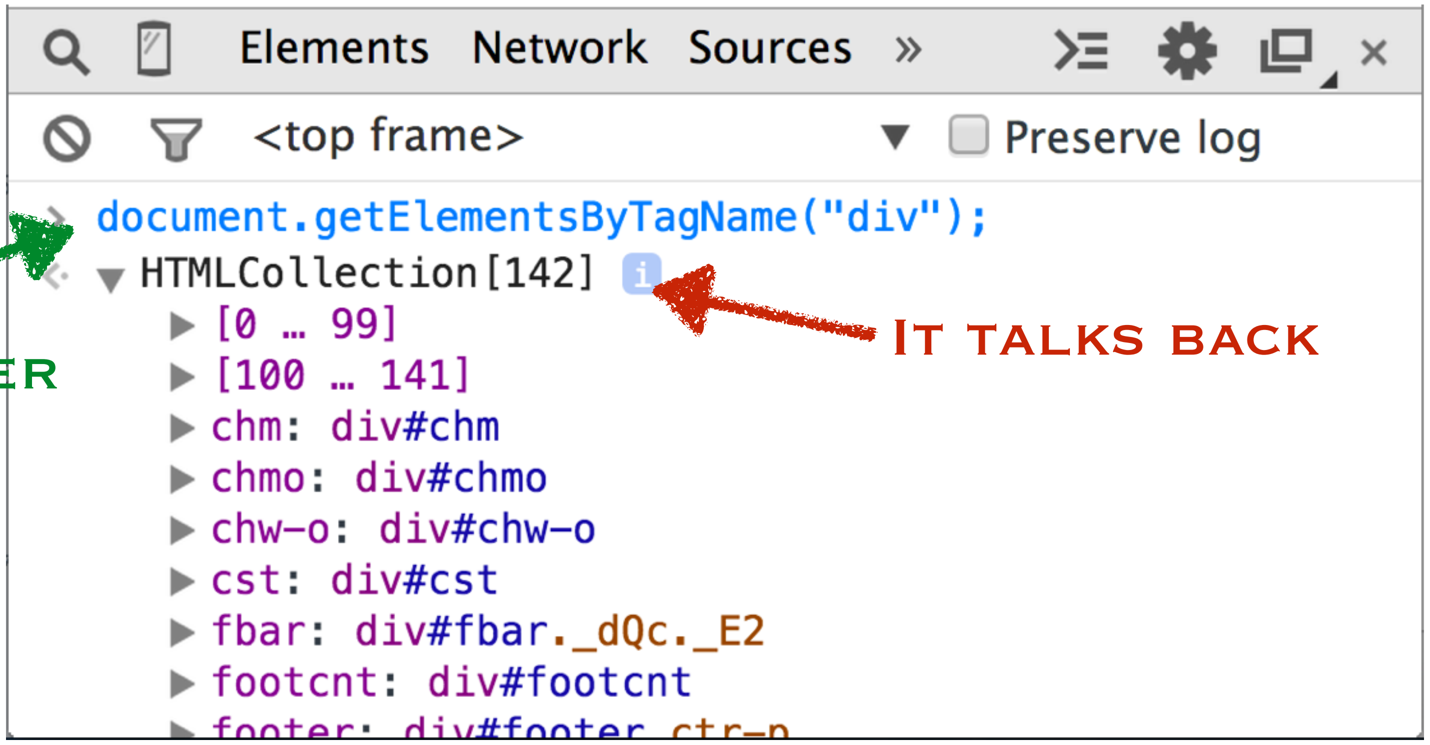
Javascript Basics

The Browser Console

For talking to the browser about the current page.



The browser speaks javascript.



```
> document.getElementsByTagName("div");
< ▼ HTMLCollection[142]
    ▶ [0 … 99]
    ▶ [100 … 141]
    ▶ chm: div#chm
    ▶ chmo: div#chmo
    ▶ chw-o: div#chw-o
    ▶ cst: div#cst
    ▶ fbar: div#fbar._dQc._E2
    ▶ footcnt: div#footcnt
    ▶ footer: div#footer.ctr-p
```

Talk to the Browser

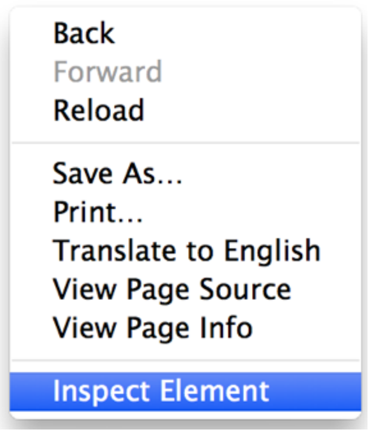
It talks back

Opening the Console

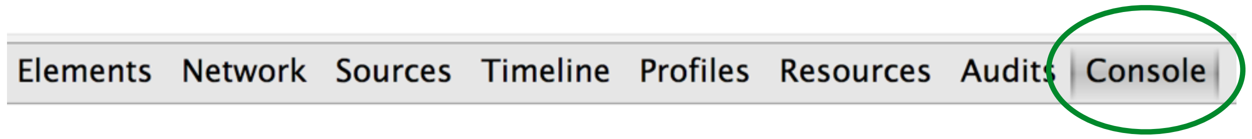
Context Menu

Right click anywhere on the page.

Select *Inspect Element*.



Click *Console*.



Opening the Console

Short Cut

Toggle the console open/closed with

command + option + j (mac)

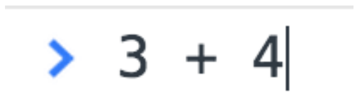
control + shift + j (windows/linux)

REPL

Read. Evaluate. Print. Loop.

The Browser Console is a REPL.

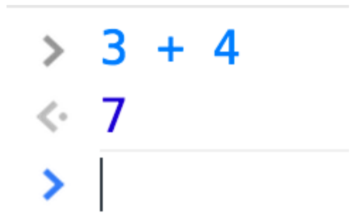
READ Type an expression into the console.



```
> 3 + 4
```

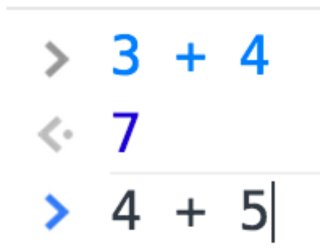
EVALUATE Press Return/Enter.

PRINT The result of the expression is printed.



```
> 3 + 4
<· 7
> 
```

LOOP Start again at READ.



```
> 3 + 4
<· 7
> 4 + 5
```

Try Me: Numbers

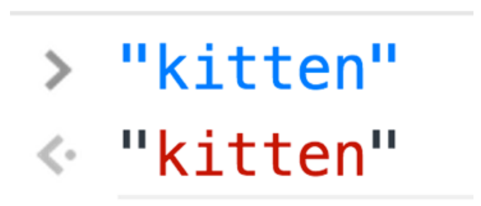
```
> 3 + 10 * 4

> 3 + (10 * 4)

> (3 + 10) * 4
```

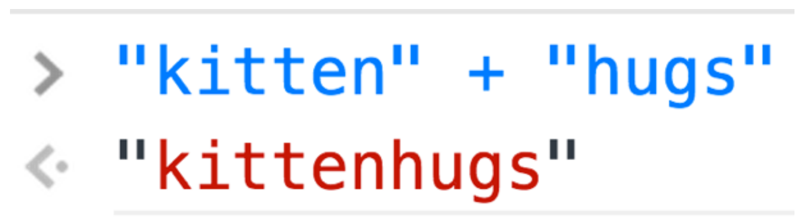
Try Me: Strings

strings are used to store text



```
> "kitten"
<· "kitten"
```

+ means something to strings, too

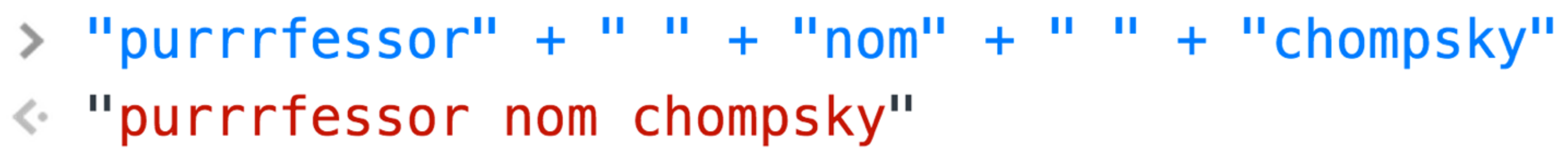


```
> "kitten" + "hugs"
<· "kittenhugs"
```

now try

```
> "purrrfessor" + " " + "nom" + " " + "chompsky"
```

Try Me: Strings



```
> "purrrfessor" + " " + "nom" + " " + "chompsky"
<· "purrrfessor nom chompsky"
```

Purrrf. Chompsky

Primitives

The simplest forms of data.

Primitive Types

number	7, 3.14, NaN, Infinity
string	“seven”, ‘seven’
boolean	true, false
null	- -
undefined	- -

typeof

Returns a string representing the type of whatever is on the right.

```
> typeof 1

> typeof "a"

> typeof true
```

Variable Declaration

```
var PI;
var x, y;
var arbitraryVariableName;
```

"var" is a javascript keyword.

Try Me

```
> ninjaTurtles;

> var ninjaTurtles;
```

Naming Variables

Variable names can have all letters, digits (but not as the first character), $ and _.

Try Me

```
> var mo$;

> var octopus_n_boots;

> var no spaces;

> var no-hyphens;

> var redFishBlueFish;
```

Naming Conventions

How we've all agreed to name variables, to make it easier to read each other's code.

Almost Everything

lowerCamelCase

```
> var currentSpock;
```

Constants

SCREAMING_SNAKE_CASE

```
> var SPOCK;

> var EVIL_SPOCK;
```

(constants are a convention)

Keywords

The vocabulary of the javascript language.

Reserved Words

Words reserved for later versions of Javascript.

abstract boolean break byte case catch char class
const continue debugger default delete do double else
enum export extends false final finally float for function
goto if implements import in instanceof int interface long
native new null package private **protected** public return
short static super switch synchronized this throw throws
transient true try typeof var void volatile while with

Assignment

Giving a value to a variable.

```
> var pi = 3.14159;
> pi = 3.14159265358979323846;
```

Initialization

A variable is initialized the first time it is assigned a value.

```
> var pi;
> pi = 3.14159; // initializing PI here
> pi = 3.14159265358979323846;
```

Try Me

```
> kitties;

> var kitties;

> kitties;

> kitties = "yes, please";

> kitties
```

So, what does undefined mean?

undefined

The value of an un-initialized variable.

Declare and Initialize

(at the same time)

```
var PI = 3.14159;
var height = 5, width = 2;
```

Try Me

```
> var numKitties = 5, numKittiesIWant = numKitties + 1;

> numKitties;

> numKittiesIWant;
```

null

The empty value (like nil in ruby).

Collecting Values

Array

List of values.

Initialize

Most of the time, arrays are created with *array literals.*

```
> var somePrimitives = ["string", 1, true, undefined, null];
```

Index

A reference to an item in a list. Javascript arrays are indexed with square brackets surrounding an integer.

```
> ninjaTurtles[0]; // the first element in list ninjaTurtles
```

Indexing starts at 0!!!

Altering

You can change the values in an array using an index.

```
> var ourCats =
    ["chompsky","buster","shadow","smudge","soot"];
  ourCats;

> ourCats[4] = "joe"; ourCats;
```

Objects

Unordered collection.

Property

The items in an Object collection. Each property has a name and a value. Property names can be any string.

Initializing

There are *object literals* similar to the *array literals* we saw earlier.

```
var starTrek = {
   goodGuys: "federation",
   badGuys: "romulans"
}
```

Property Names   Property Values

Indexing

Just like an array -- the index is surrounded by square brackets.

```
> starTrek["goodGuys"];
> starTrek["scientific accuracy"];
```

Indexing non-existent properties returns undefined.

Altering Properties

Treat the indexed property like a variable.

```
> starTrek["goodGuys"] = ["federation", "klingons"];
```

Adding Properties

Pretend like you're altering a property that's already there.

```
> starTrek["latest-planet-to-join"] = "klingons";
```

Dot Notation

Special syntax for accessing object properties.

```
> starTrek.goodGuys;
```

This won't work for all property names.

Try Me

```
> var starTrek = {};

> starTrek["latest-planet-to-join"] = "klingons";
  starTrek.latest-planet-to-join;

> starTrek["misunderstood civilizations"] = ["borg"];
  starTrek.misunderstood civilizations;

> starTrek["1stRule"] = "prime directive";
  starTrek.1stRule;
```

To use dot notation, the property name has to be a valid variable name plus keywords.

Statements

Instruction or action.

Most statements must end with semi-colons.

assignment	`one = 1;`
control	`if (true) {} else {} // no semi-colon`
declaration	`var kittenCuddles;`
invocations	`meow();`

function definition	`function meow() {`
	`  console.log(“mrreeooow”);`
	`} // no semi-colon`

Semi-colon Insertion

If given code with missing semi-colons, javascript will attempt to insert them before executing the code.

This generally agreed to be a bad trait of the language.

Don't forget semi-colons!

Functions

An ordered group of statements that (ideally) perform a single task.

Declaring

Use the keyword *function*.

START A FUNCTION DECLARATION

FUNCTION NAME

```
> function stop() {
    alert("hammer time");
  }
```

FUNCTION BODY (EVERYTHING INSIDE THE {})

There's another way...we'll learn it soon.

Invocation

Execute the statements inside the function body.
Append *();* to the function name.

```
> stop();
```

return value

The value of a function invocation.

SPECIFY RETURN
VALUE WITH THE
RETURN KEYWORD

```
> function stop() {
    return "hammer time";
  }
  var returnValue = stop();
  returnValue;
```

Functions are Values

...they can be assigned to variables like
any other object.

Try Me

```
> var stop = function () {
    return "hammer time";
  };
  stop;
> stop();
```

Output

Our REPL only prints out the value of the previous statement.

What if we want to explicitly print something?

console.log()

Print arguments to the console.

Try Me

```
> console.log(["chicken", "turkey", "duck"]);
  console.log("hey, that's a turducken");
  console.log("that's", 1, "...", 2, "...", 3, "birds in", 1);
```

Why is *undefined* printed?

jQuery

A commonly used javascript library — the preferred way of responding to events, and querying/manipulating the DOM.

Follow Along

```
git clone https://github.com/mel-quark/kitten-toes.git
cd kitten-toes
subl .
```

$()

Shorthand for jQuery()

Used to (1) find elements in the page,
(2) create new elements, and
(3) map DOM elements into jQuery objects.

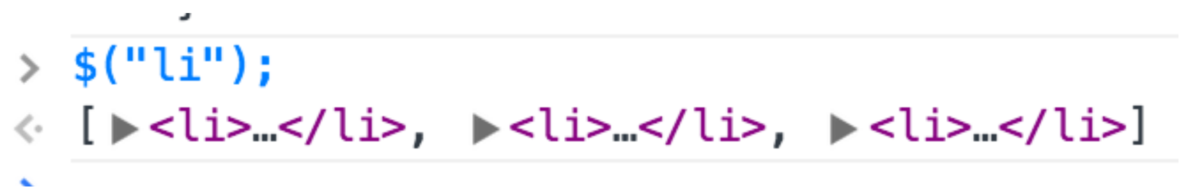
Returns a jQuery object.

(including jQuery puts this function in the global namespace)

the jQuery object

A wrapper around a list of DOM elements that provides methods for accessing and changing those elements.

$("li"); returned a jQuery object with all the li’s in the page. (try it)



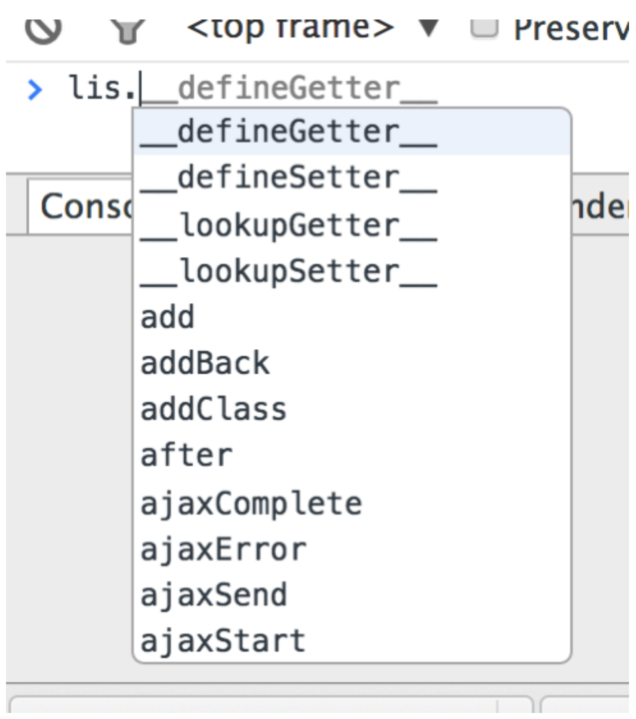
```
> $("li");
<· [▶<li>…</li>, ▶<li>…</li>, ▶<li>…</li>]
```

the jQuery API

The jQuery() function plus all methods available on jQuery objects.

Try this:
var lis = $("li");

Now type "lis." (without the quotes) to see all the jQuery object methods.



```
> lis.__defineGetter__
__defineGetter__
__defineSetter__
__lookupGetter__
__lookupSetter__
add
addBack
addClass
after
ajaxComplete
ajaxError
ajaxSend
ajaxStart
```

selectors

Strings used to query for DOM elements.

```
$("li"); // by tag name
$("#kitten-ul"); // by id
$(".mobile"); // by class name
$("ul.mobile"); // by tag and class
$("ul li"); // descendant
$("li + li"); // sibling
$("div, section"); // union
```

chaining

When a method returns the object it was called upon.

Used for aggregating several actions in a single, easy-to-read statement.

```
$("li").css("background-color", "blue")
       .append("add some text programmatically")
       .css("color", "white");
```

$variables

Convention to prepend variables that hold jQuery objects with a $.

```
$lis = $("li");
```

Events

How the browser notifies your code about user interaction and state changes.

Examples

- *load*: coordination
- *resize*: interaction with the browser
- *click*: interaction with the page
- *custom*: your javascript (or a library) triggers an event

Event Handler

A function that runs in response to an event.

on()

Register an event handler.

Usage:

```
$(dom element or css selector).on(event, handler);
```

load

An event on the *window* object.

Signifies that the page is done loading.

Use it to make sure all elements are available before performing a task.

Try Me

Open an alert dialog on page load, warning the user about the impending cuteness.

Do this in kitten-toes.js. It won’t work in the console. Why not?

Solution

```
$(window).on("load", function() {
  alert("caution: so much cuteness");
});
```

$(document).ready()

Register a handler for when the DOM is available.
This is much more common than waiting for the "load" event.

Synchronizes and creates a new scope.

Solution

```
$(document).ready(function() {
  alert("caution: so much cuteness");
});
```

it’s so common, it has a shorthand:

```
$(function() {
  alert("caution: so much cuteness");
});
```

$(this)

Inside an event handler, *this* is the element the event acted upon.

click

An event on a DOM element

Signifies the user clicked that element
with a mouse or trackpad.

Interact with the user: open a menu,
respond to a button-click, etc.

Try Me

Every time an image is clicked,
move it to the end of the <ul>.

Solution

```
$("img").click(function() {
  var $li = $(this).parent();
  var $ul = $li.parent();
  $li.detach()
     .appendTo($ul);
});
```

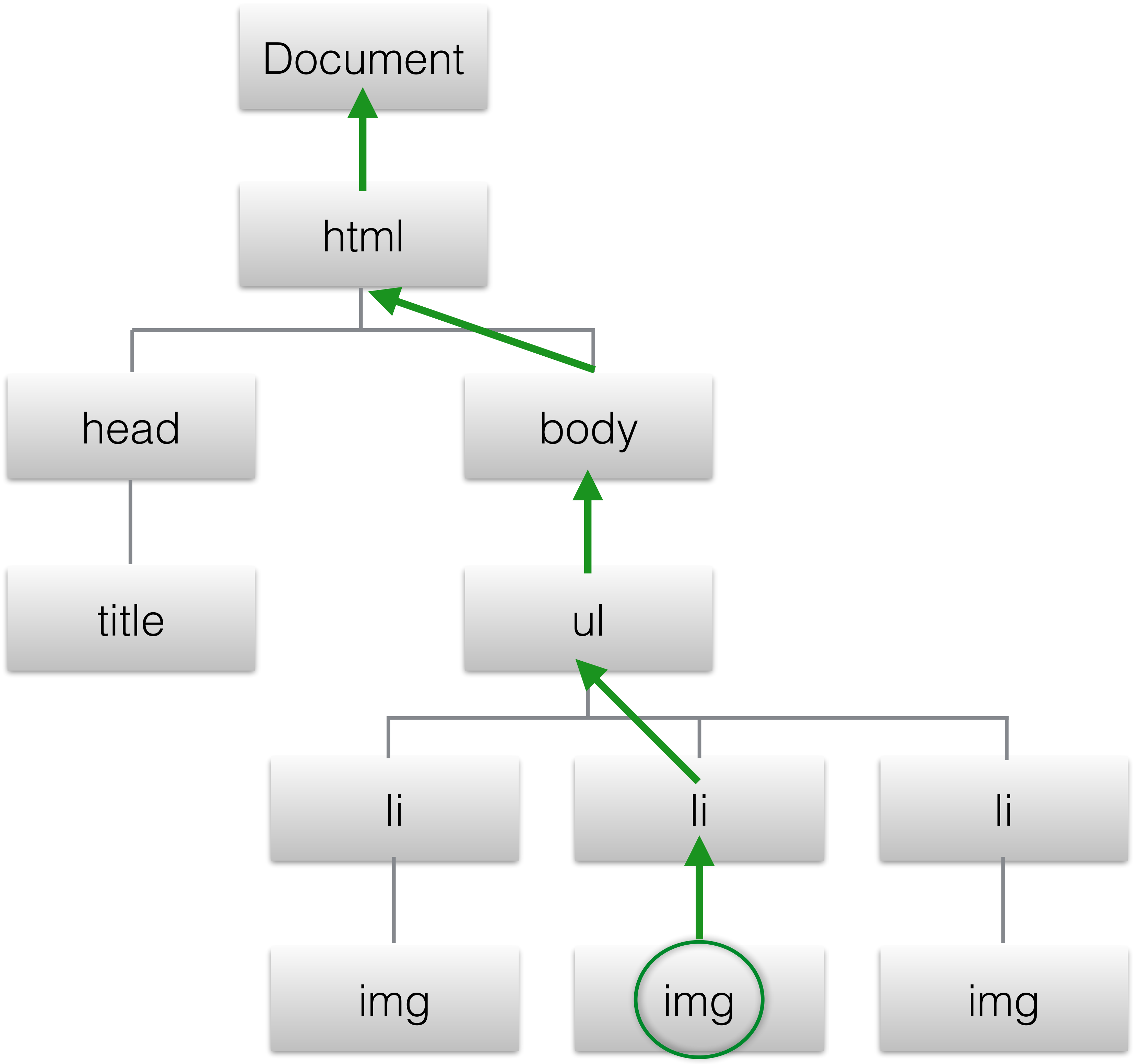
But that only works once with each image.

Event Propagation

Events affect more than the element acted upon.

Bubbling

Events go up the tree.



event.target

jQuery Docs

http://api.jquery.com

jQuery Plugins

Libraries that provide extra functionality and build on top of jQuery.

Example: http://plugins.jquery.com/?s=menu