

Javascript



Arnelle Balane

But first, a
history lesson...

1995

- Created by Brendan Eich in 10 days (May 1995)
 - Originally named Mocha
 - Changed to LiveScript in September
 - Final name JavaScript licensed in December
-

1996 - 2003

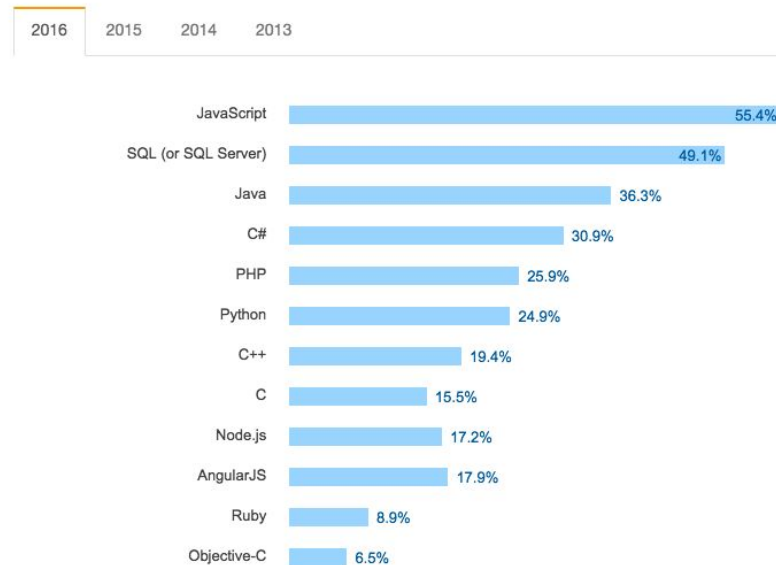
- JavaScript was taken to ECMA to create standard specification
 - ECMAScript is the standard; JavaScript is the implementation
 - Conflict begins with Microsoft refusing to cooperate
 - Brendan Eich rejoins Ecma
-

2005 - Present

- 2005: Jesse James Garrett describes Ajax in a paper
 - JavaScript renaissance period: open source libraries Prototype, jQuery, Dojo, and Mootools released
 - 2008: Conflict is resolved and both parties work to drive the language forward
-

Why Javascript?

I. Most Popular Technologies



49,397 responses

More people use JavaScript than use any other programming language. PHP appears to be falling out of favor as Node and Angular emerge.

StackOverflow Developer Survey 2016

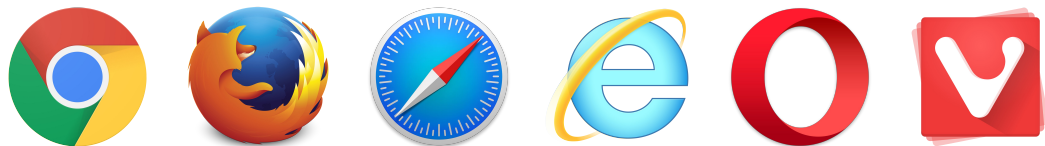
easy to get started

easy to get started

did I ask you to install anything? :p

it's everywhere!

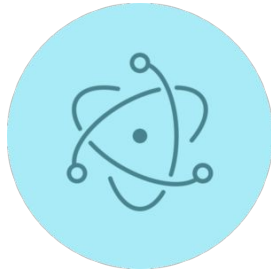
(almost, but working on it)



from browsers-side



to servers-side



to desktop applications



even operating systems!

Part I

The Language

Data Types

true false

booleans

1

2

3

4

5

1 . 2

2 . 3

3 . 4

4 . 5

numbers

' I am a string '

" I am an awesome string "

strings

[1, 2, 3, 4, 5]

arrays

```
{ a: 1, b: 2, c: 3 }
```

objects

key
↓
{ a : 1 , b : 2 , c : 3 }
↑
value

objects

```
function name( ) { }
```

functions

```
var name = 'Javascript';
```

we can store them in **variables**!

$$1 + 2 * (3 - 4) / 5$$

numbers

arithmetic operations

$$1 + 2 * (3 - 4) / 5$$
$$= 0.6$$

numbers

arithmetic operations

```
Math.pow( 2 , 2 ) ;
```

```
Math.sqrt( 4 ) ;
```

etc.

numbers
other math stuff

'Javascript is awesome!'

strings

```
'hello' + 'world'
```

strings

concatenation

```
'hello' + 'world'  
  
= 'helloworld'
```

strings
concatenation

```
'Javascript'.length
```

strings

number of characters

`'Javascript'.length`

`⇒ 10`

strings

number of characters


```
var name = 'Javascript';
```

strings

individual characters

```
var name = 'Javascript';  
name[0];
```

strings

individual characters

```
var name = 'Javascript';  
name[0];
```

strings

individual characters

```
var name = 'Javascript';  
name.substring(0, 4);
```

strings
substrings

```
var name = 'Javascript';  
name.substring(0, 4);
```

strings
substrings

bit.ly/javascript-string

strings
reference

[1, 2, 3, 4, 5]

arrays

```
[ 1, 2.3, 'string', [1] ]
```

arrays

can contain anything


```
[ 1, 2, 3 ].length
```

⇒ 3

arrays

number of items

```
var array = [ 1, 2, 3 ];
```

arrays
getting values

```
var array = [ 1, 2, 3 ];  
array[0];
```

arrays
getting values

```
var array = [ 1, 2, 3 ];  
array[0];
```

arrays
getting values

```
var array = [ 1, 2, 3 ];  
    array[0] = 4;
```

arrays
setting values

```
var array = [ 4, 2, 3 ];  
array[0] = 4;
```

arrays
setting values

```
var array = [ [1], [2] ] ;
```

arrays
nested arrays

```
var array = [ [1], [2] ];  
array[0];
```

arrays
nested arrays


```
var array = [ [1], [2] ];  
array[0];
```

arrays
nested arrays

```
var array = [ [1], [2] ];  
array[0][0];
```

arrays
nested arrays

```
var array = [ [1], [2] ];  
            array[0][0];
```

arrays
nested arrays

```
[1, 2].concat([3, 4])
```

arrays
concatenation

[1, 2, 3, 4]

arrays
concatenation

```
var stack = [ 1, 2, 3 ];
```

arrays

stack operations

```
stack.push(4);
```

arrays

stack operations

stack \Rightarrow [1, 2, 3, 4]

arrays

stack operations


```
var top = stack.pop();
```

arrays

stack operations

top \Rightarrow 4

arrays
stack operations

top \Rightarrow 4

stack \Rightarrow [1, 2, 3]

arrays

stack operations

```
var queue = [ 1, 2, 3 ];
```

arrays

queue operations

```
queue.push(4);
```

arrays

queue operations

```
queue.push(4);
```

it's not called "enqueue" ಠ_ಠ

arrays

queue operations

queue \Rightarrow [1, 2, 3, 4]

arrays
queue operations

```
var front = queue.shift();
```

arrays

queue operations


```
var front = queue.shift();
```

it's not called "dequeue" either ° (`A') °

arrays

queue operations

front \Rightarrow 1

arrays
queue operations

front \Rightarrow 1

queue \Rightarrow [2, 3, 4]

arrays
queue operations

bit.ly/javascript-array

arrays
reference

```
{ a: 1, b: 2, c: 3 }
```

objects

{

a : 1 ,

b : 2 ,

c : 3

}

objects



objects
simplest form

```
var object = {};
```

objects
simplest form


```
object.key = 'value';
```

objects

adding key-value pairs

```
object.key = 'value';
```

or

```
object['key'] = 'value';
```

objects

adding key-value pairs

object \Rightarrow { key: 'value' }

objects

adding key-value pairs

```
delete object.key;
```

objects

removing key-value pairs

object \Rightarrow {}

objects

removing key-value pairs

```
var object = {  
    key: { a: 1, b: 3 }  
};
```

objects

nested objects

```
var object = {  
    key: { a: 1, b: 3 }  
};
```

```
object.key;
```

objects

nested objects

```
var object = {  
  key: { a: 1, b: 3 }  
};
```

`object.key;`

objects

nested objects


```
var object = {  
    key: { a: 1, b: 3 }  
};
```

```
object.key.a;
```

objects

nested objects

```
var object = {  
    key: { a: 1, b: 3 }  
};
```

```
object.key.a;
```

objects

nested objects

bit.ly/javascript-object

objects
reference

`null` `undefined`

`null & undefined`

Control Structures

```
if (condition) {}  
else if (condition) {}  
else {}
```

if...else if...else

```
if (condition) {}  
else if (condition) {}  
else if (condition) {}  
else if (condition) {}  
else {}
```

if...else if...else

can have as many “else if”s as needed

conditions resolve to either

true or **false**

conditions

< <= > >=
== != === !==

conditions
equality comparisons

`condition && condition`

`condition || condition`

conditions

compound conditions

false

null

undefined

' '

0

NaN

conditions

falsy values

everything else

conditions
truthy values

```
var number = 12;  
if (number > 20) {  
    // (a) do something  
} else if (number > 10) {  
    // (b) do another thing  
} else {  
    // (c) do something else  
}
```

if...else if...else

```
var number = 12;  
if (number > 20) {  
    // (a) do something  
} else if (number > 10) {  
    // (b) do another thing  
} else {  
    // (c) do something else  
}
```

if...else if...else

```
var string = '';  
if (string) {  
    // (a) do something  
} else {  
    // (b) do something else  
}
```

if...else if...else

```
var string = '';  
if (string) {  
    // (a) do something  
} else {  
    // (b) do something else  
}
```

if...else if...else


```
var string = 'hello';  
if (string) {  
    // (a) do something  
} else {  
    // (b) do something else  
}
```

if...else if...else

```
var string = 'hello';  
if (string) {  
    // (a) do something  
} else {  
    // (b) do something else  
}
```

if...else if...else

```
switch (something) {  
    case value: /* do something */;  
    case value: /* do something */;  
    default: /* do something */;  
}
```

switch...case

```
var number = 12;  
switch (number) {  
    case 6: /* (a) */;  
    case 12: /* (b) */;  
    case 18: /* (c) */;  
    default: /* (d) */;  
}
```

switch...case

```
var number = 12;  
switch (number) {  
    case 6: /* (a) */;  
    case 12: /* (b) */;  
    case 18: /* (c) */;  
    default: /* (d) */;  
}
```

switch...case

```
var number = 12;  
switch (number) {  
    case 6: /* (a) */;  
    case 12: /* (b) */;  
    case 18: /* (c) */;  
    default: /* (d) */;  
}
```

switch...case

```
var number = 12;  
switch (number) {  
    case 6: /* (a) */;  
    case 12: /* (b) */;  
    case 18: /* (c) */;  
    default: /* (d) */;  
}
```

switch...case

```
var number = 12;  
switch (number) {  
    case 6: /* (a) */;  
    case 12: /* (b) */;  
    case 18: /* (c) */;  
    default: /* (d) */;  
}
```

"fallthrough"

switch...case


```
var number = 12;  
switch (number) {  
    case 6: /* (a) */; break;  
    case 12: /* (b) */; break;  
    case 18: /* (c) */; break;  
    default: /* (d) */;  
}
```

switch...case

```
var number = 12;
switch (number) {
    case 6: /* (a) */; break;
    case 12: /* (b) */; break;
    case 18: /* (c) */; break;
    default: /* (d) */;
}
```

switch...case

```
while (condition) {  
    // do something while  
    // condition is true  
}
```

while loop

```
var number = 0;
```

while loop

```
var number = 0;  
while (number <= 10) {}
```

while loop

```
var number = 0;  
while (number <= 10) {  
    number = number + 1;  
}
```

while loop

```
while (true) {  
    // infinite loop!  
}
```

while loop

```
while (true) {  
    // infinite loop!  
    break;  
}
```

while loop


```
for (declaration; condition; update) {}
```

for loop

```
for ( var i = 1 ; condition; update) {}
```

for loop

```
for ( var i = 1 ; i <= 10 ; update ) {}
```

for loop

```
for ( var i = 1 ; i <= 10 ; i += 1 ) {}
```

for loop

```
for ( var i = 1 ; i <= 10 ; i += 1 ) { }  
    eye = eye + one ;)
```

for loop

```
var letters = [ 'a', 'b', 'c' ];  
var length = letters.length;
```

for loop

```
var letters = [ 'a', 'b', 'c' ];  
var length = letters.length;  
for (var i = 0; i < length; i++) {}
```

for loop

```
var letters = [ 'a', 'b', 'c' ];  
var length = letters.length;  
  
for (var i = 0; i < length; i++) {  
    var letter = letters[i];  
}
```

for loop


```
var letters = [ 'a', 'b', 'c' ];  
var length = letters.length;  
  
for (var i = 0; i < length; i++) {  
    var letter = letters[i];  
    // i ⇒ 0, letter ⇒ 'a'  
}
```

for loop

```
var letters = [ 'a', 'b', 'c' ];  
var length = letters.length;  
  
for (var i = 0; i < length; i++) {  
    var letter = letters[i];  
    // i ⇒ 1, letter ⇒ 'b'  
}
```

for loop

```
var letters = [ 'a', 'b', 'c' ];  
var length = letters.length;  
  
for (var i = 0; i < length; i++) {  
    var letter = letters[i];  
    // i ⇒ 2, letter ⇒ 'c'  
}
```

for loop

```
for (key in object) {}
```

for..in loop

key
↓
{ a: 1, b: 2, c: 3 }

for..in loop

```
var object = { a: 1, b: 2, c: 3 };
```

for..in loop

```
var object = { a: 1, b: 2, c: 3 };  
  
for (var key in object) {}
```

for..in loop

```
var object = { a: 1, b: 2, c: 3 };  
  
for (var key in object) {  
    // key ⇒ 'a'  
}
```

for..in loop


```
var object = { a: 1, b: 2, c: 3 };  
  
for (var key in object) {  
    // key ⇒ 'b'  
}
```

for..in loop

```
var object = { a: 1, b: 2, c: 3 };  
  
for (var key in object) {  
    // key ⇒ 'c'  
}
```

for..in loop

Functions

```
function name() {  
    // function body  
}
```

functions

```
function name(parameter) {  
    // function body  
}
```

functions
with parameter

```
function name(para, meter) {  
    // function body  
}
```

functions

with multiple parameters

```
function add(x, y) {  
    var sum = x + y;  
    return sum;  
}
```

functions

example: “add” function

```
function add(x, y) {  
    var sum = x + y;  
    return sum;  
}
```

```
add(12, 34);
```

functions

example: invoking the “add” function


```
function add(x, y) {  
    var sum = x + y;  
    return sum;  
}
```

```
add(12, 34);    // ⇒ 46
```

functions

example: invoking the “add” function

```
function transform(name, fn) {  
    return fn(name);  
}
```

functions
as parameters

```
function transform(name, fn) {  
    return fn(name);  
}
```

functions
as parameters

```
function uppercase(name) {  
    return name.toUpperCase();  
}
```

functions
as parameters

```
function uppercase(name) {  
    return name.toUpperCase();  
}
```

```
transform('Javascript', uppercase);
```

functions
as parameters

```
function uppercase(name) {  
    return name.toUpperCase();  
}
```

```
transform('Javascript', uppercase);  
// ⇒ 'JAVASCRIPT'
```

functions
as parameters

```
function lowercase(name) {  
    return name.toLowerCase();  
}
```

```
transform('Javascript', lowercase);
```

functions
as parameters

```
function lowercase(name) {  
    return name.toLowerCase();  
}
```

```
transform('Javascript', lowercase);  
// ⇒ 'javascript'
```

functions
as parameters


```
var array = [ 1, 2, 3, 4, 5 ];  
var doubles = array.map(function(number) {  
    return number * 2;  
} );
```

functions

example: Array.map

```
var array = [ 1, 2, 3, 4, 5 ];  
var doubles = array.map(function(number) {  
    return number * 2;  
});
```

functions

example: Array.map

```
var array = [ 1, 2, 3, 4, 5 ];  
var doubles = array.map(function(number) {  
    return number * 2;  
});  
// doubles ⇒ [ 2, 4, 6, 8, 10 ]
```

functions

example: Array.map

```
function outer() {  
    return function inner() {  
        // function body  
    };  
}
```

functions

returning functions

```
function outer() {  
    return function inner() {  
        // function body  
    };  
}
```

functions

returning functions

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}  
  
var addFive = adder(5);
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
var addFive = adder(5);
```

functions

example: adder


```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
var addFive = adder(5);
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
var addFive = adder(5);
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
var addFive = adder(5);  
addFive(10);
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
var addFive = adder(5);  
addFive(10);
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
var addFive = adder(5);  
addFive(10); // ⇒ 15
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
adder(5)(10); // ⇒ 15
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
adder(5)(10); // ⇒ 15
```

functions

example: adder

```
function adder(x) {  
    return function add(y) {  
        return x + y;  
    };  
}
```

```
adder(5)(10); // ⇒ 15
```

functions

example: adder


```
var addTen = adder(10);  
var addTwenty = adder(20);
```

```
addTen(15);    // ⇒ 25  
addTwenty(35); // ⇒ 55
```

functions

example: adder

Challenge Set 3

`javascript-workshop/03-functions`

Part II

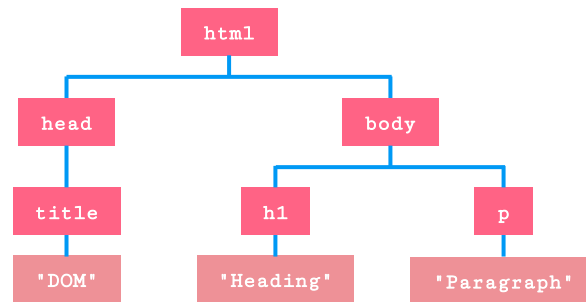
Javascript in the Browser

Interacting with HTML

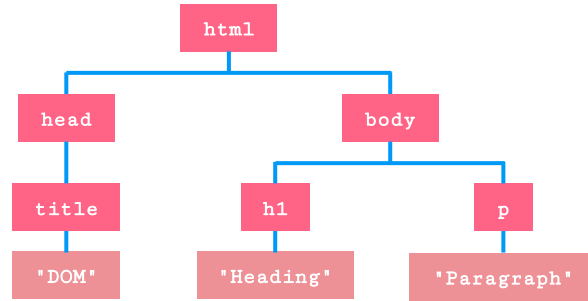
```
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1>Heading</h1>
    <p>Paragraph</p>
  </body>
</html>
```

HyperText Markup Language
aka HTML

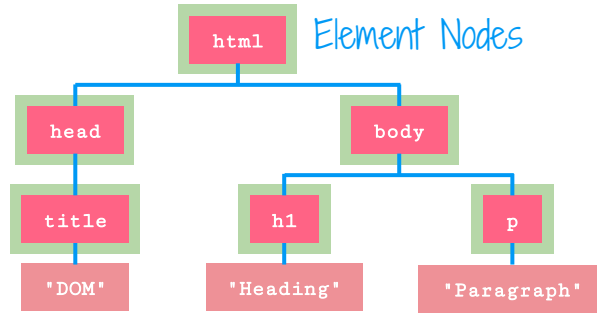
```
<html>
  <head>
    <title>DOM</title>
  </head>
  <body>
    <h1>Heading</h1>
    <p>Paragraph</p>
  </body>
</html>
```



Document Object Model
aka DOM

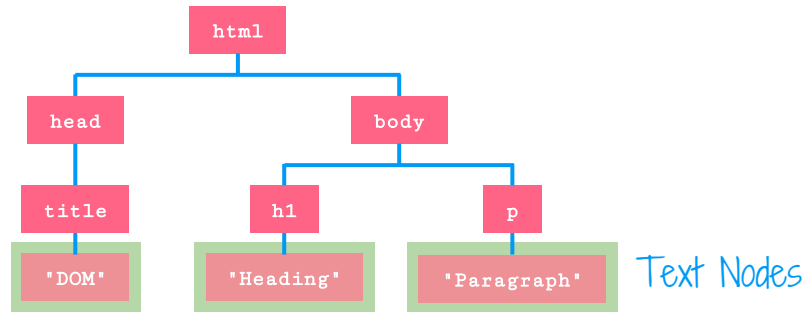


Document Object Model
aka DOM



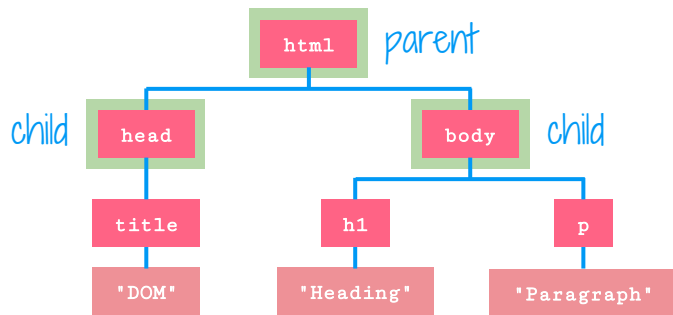
Document Object Model

element nodes



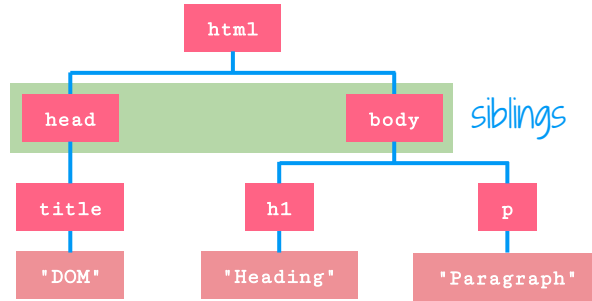
Document Object Model

text nodes



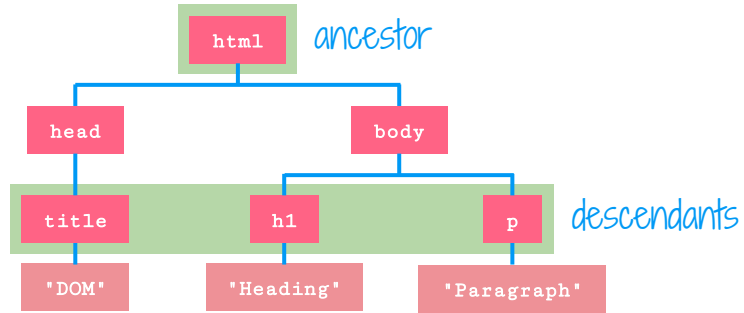
Document Object Model

“family tree”



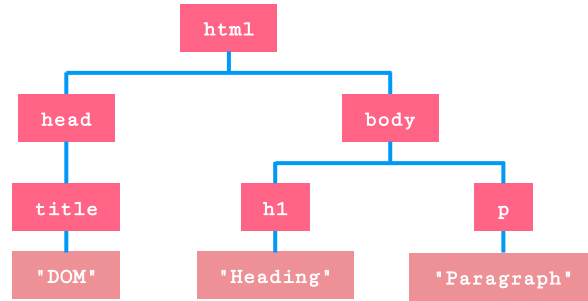
Document Object Model

“family tree”

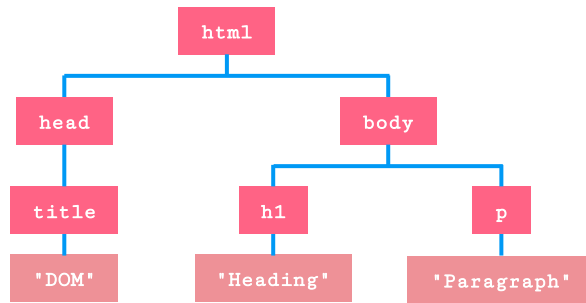


Document Object Model

“family tree”

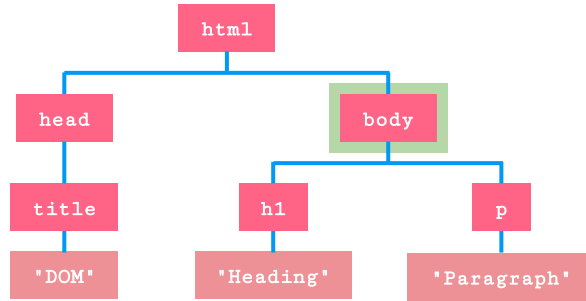


selectors



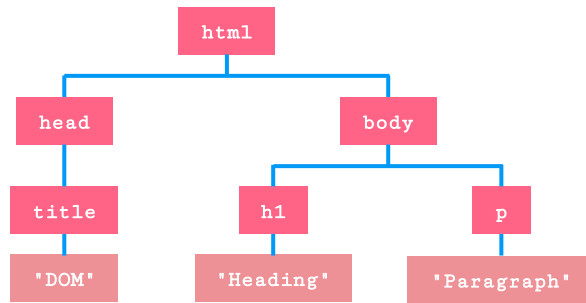
`selector: 'body'`

`selectors`



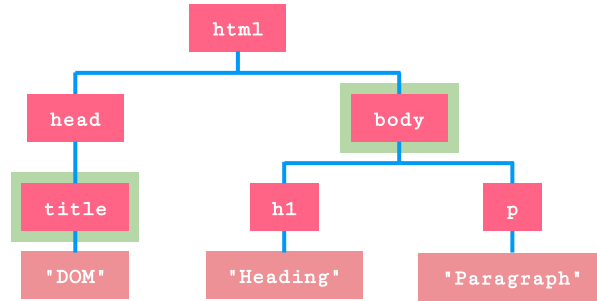
selector: 'body'

selectors



`selector: 'body, title'`

selectors



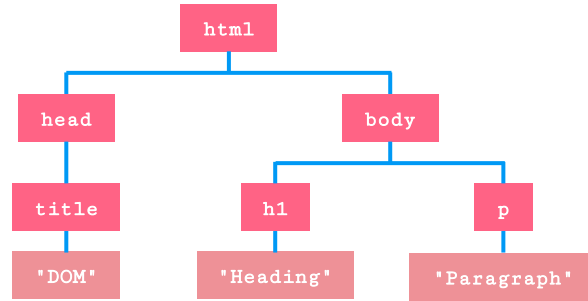
`selector: 'body, title'`

selectors

bit.ly/selectors-reference

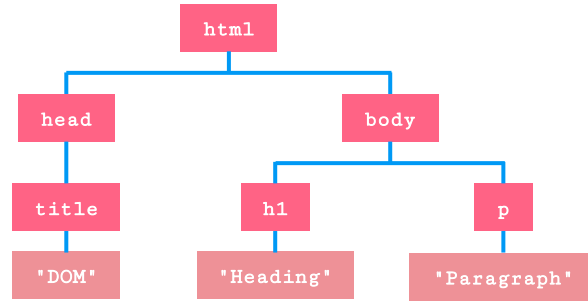
selectors

more information about selectors



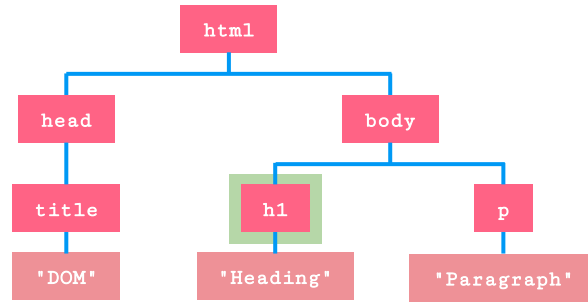
```
document.querySelector(selector);
```

selecting elements



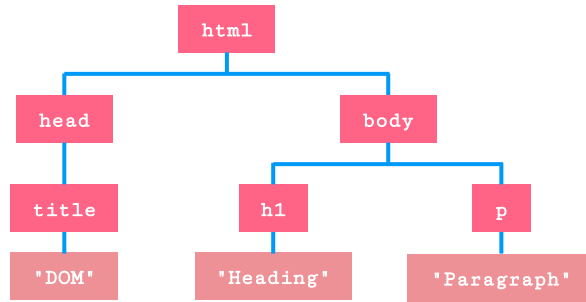
```
document.querySelector( 'h1' );
```

selecting elements



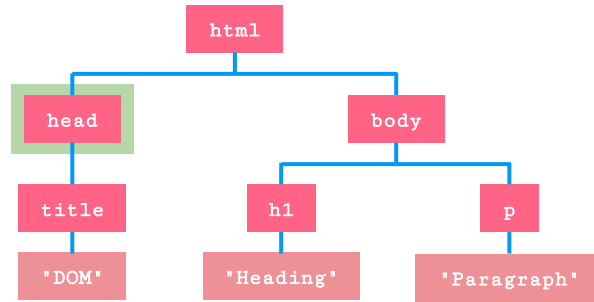
```
document.querySelector( 'h1' );
```

selecting elements



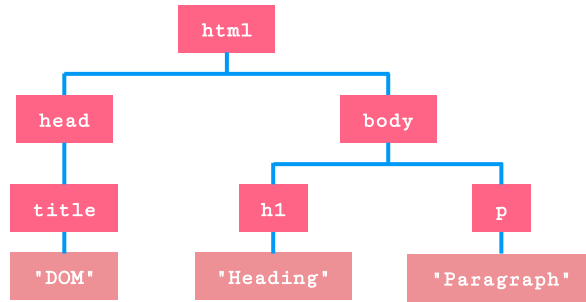
```
document.querySelector( 'head' );
```

selecting elements



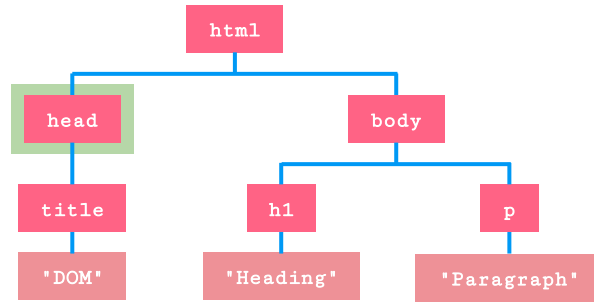
```
document.querySelector( 'head' );
```

selecting elements



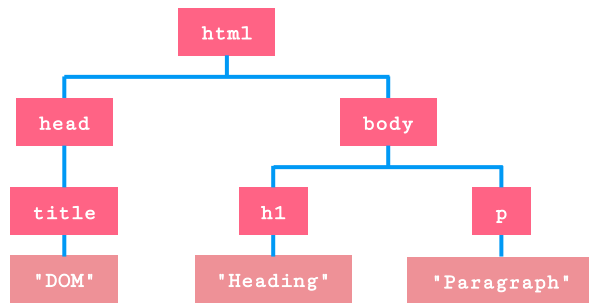
```
document.querySelector('h1, head');
```

selecting elements



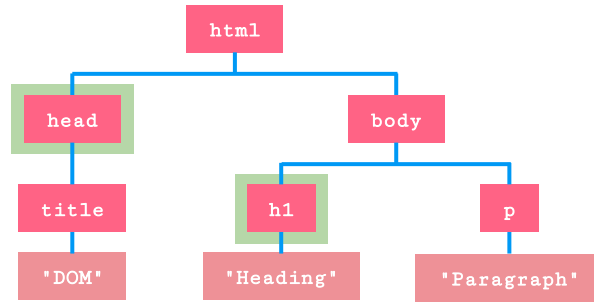
```
document.querySelector('h1, head');
```

selecting elements



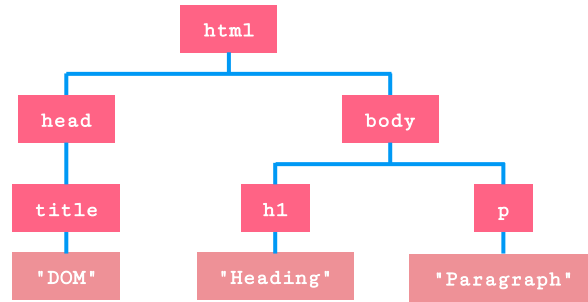
```
document.querySelectorAll('h1, head');
```

selecting elements

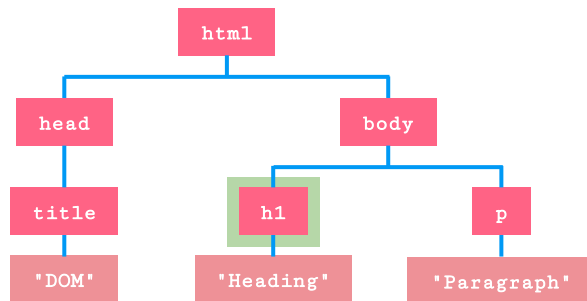


```
document.querySelectorAll('h1, head');
```

selecting elements



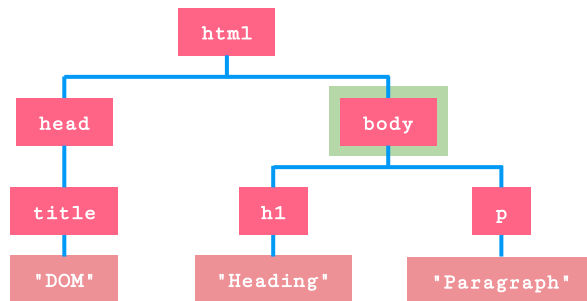
moving around the DOM tree



```
var h1 = document.querySelector('h1');
```

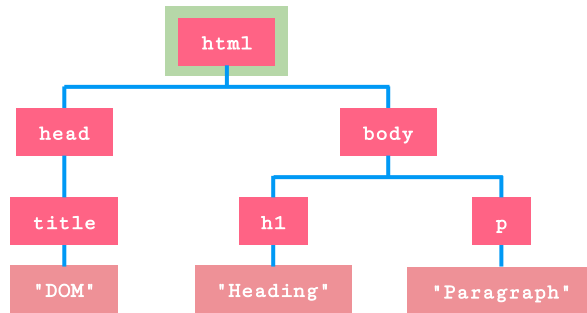
moving around the DOM tree

getting the parent element



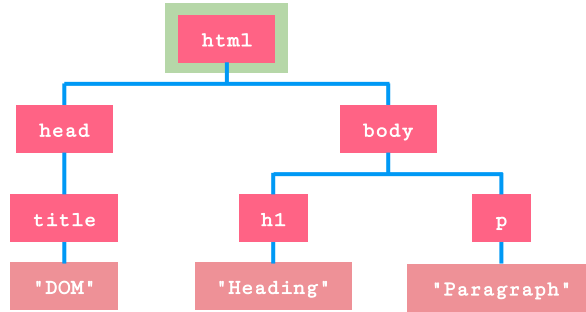
```
var h1 = document.querySelector('h1');  
h1.parentElement;
```

moving around the DOM tree
getting the parent element



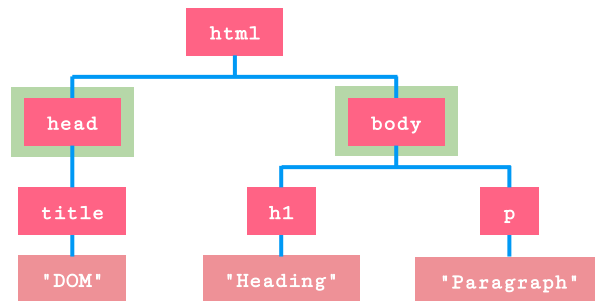
```
var h1 = document.querySelector('h1');  
h1.parentElement.parentElement;
```

moving around the DOM tree
getting the parent element



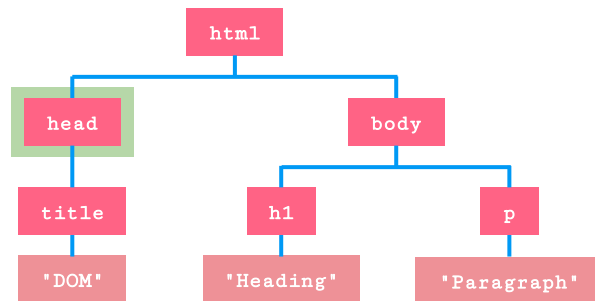
```
var html = document.querySelector('html');
```

moving around the DOM tree
getting element children



```
var html = document.querySelector('html');  
html.children;
```

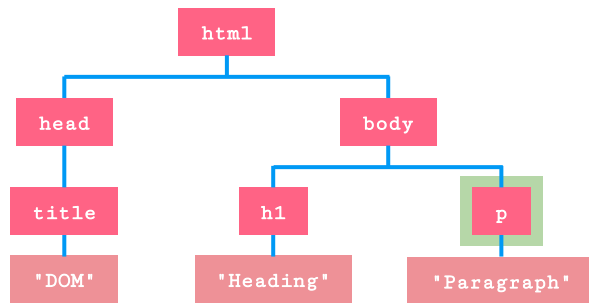
moving around the DOM tree
getting element children



```
var html = document.querySelector('html');  
html.children[0];
```

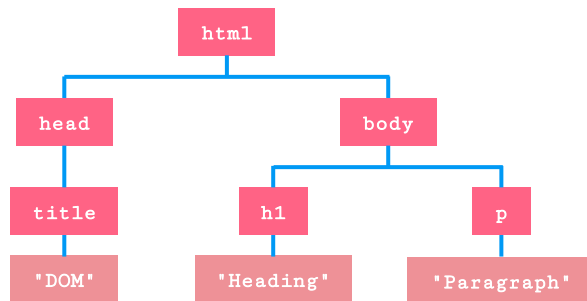
moving around the DOM tree
getting element children

lol got tired of making slides...
sorry $\mathbb{L}(\emptyset_\omega) = \neg$



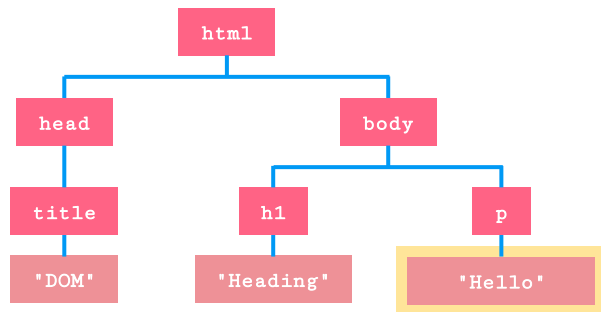
```
var p = document.querySelector('p');
```

manipulating elements



```
var p = document.querySelector('p');  
p.textContent = 'Hello';
```

manipulating elements



```
var p = document.querySelector('p');  
p.textContent = 'Hello';
```

manipulating elements

```
var h2 = document.querySelector('h2');  
h2.style.color = 'red';  
h2.style.backgroundColor = 'blue';
```

manipulating elements
manipulating their styles

```
var h2 = document.querySelector('h2');  
h2.style.color = 'red';  
h2.style.backgroundColor = 'blue';
```

manipulating elements
manipulating their styles

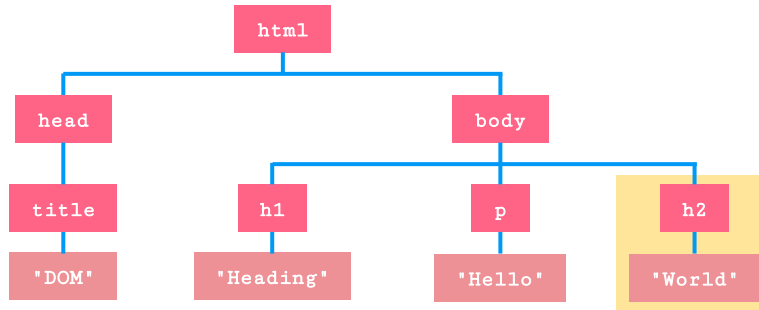

```
var h2 = document.createElement( 'h2' );  
h2.textContent = 'World';
```

inserting elements

create them first

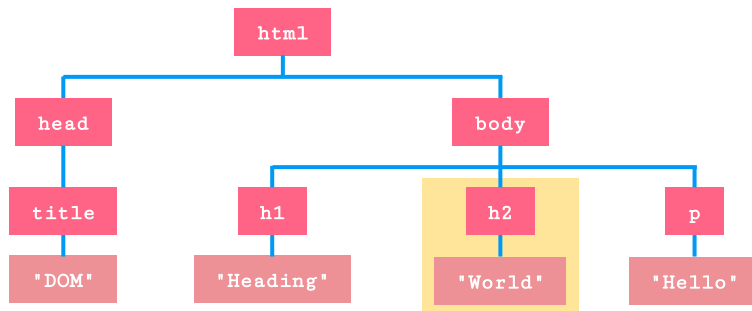
```
var h2 = document.createElement( 'h2' );  
h2.textContent = 'World';  
// <h2>World</h2>
```

inserting elements
create the element first



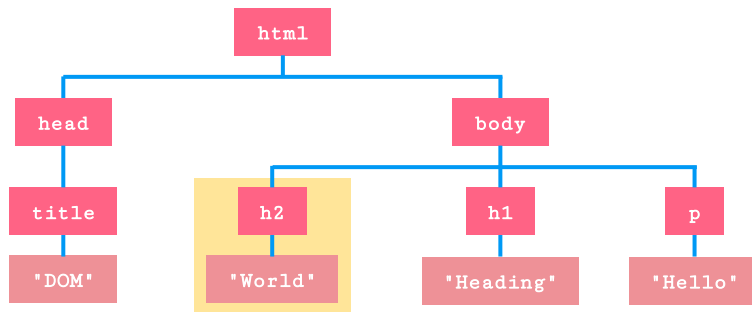
```
var body = document.querySelector('body');  
body.appendChild(h2);
```

inserting elements



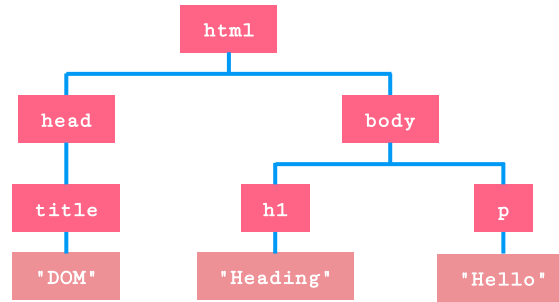
```
var body = document.querySelector('body');  
var p = document.querySelector('p');  
body.insertBefore(h2, p);
```

inserting elements

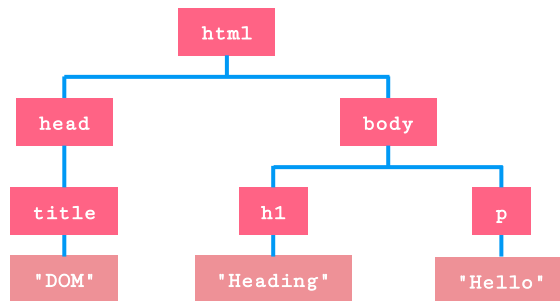


```
var body = document.querySelector('body');  
var h1 = document.querySelector('h1');  
body.insertBefore(h2, h1);
```

inserting elements

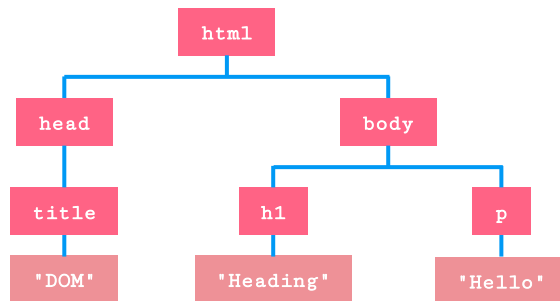


replacing elements



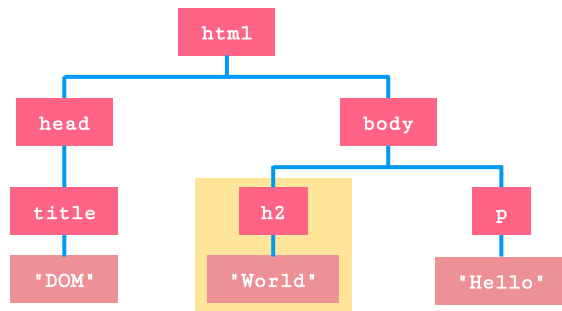
```
var body = document.querySelector('body');  
var h1 = document.querySelector('h1');
```

replacing elements



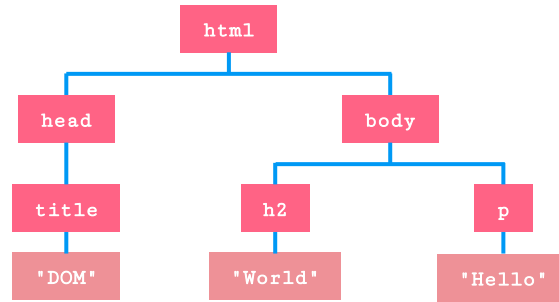
```
var body = document.querySelector('body');  
var h1 = document.querySelector('h1');  
body.replaceChild(h2, h1);
```

replacing elements

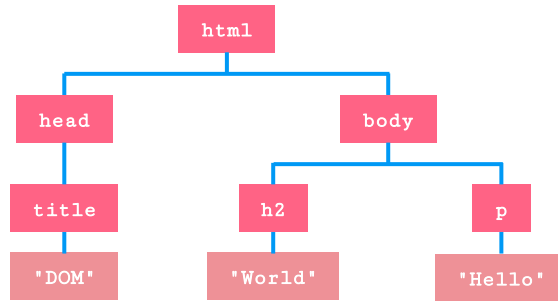


```
var body = document.querySelector('body');  
var h1 = document.querySelector('h1');  
body.replaceChild(h2, h1);
```

replacing elements

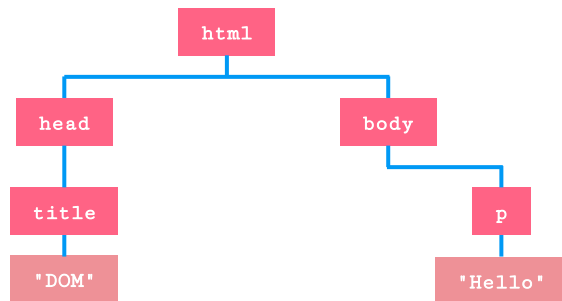


removing elements



```
var h1 = document.querySelector('h1');  
h1.remove();
```

removing elements



```
var h1 = document.querySelector('h1');  
h1.remove();
```

removing elements

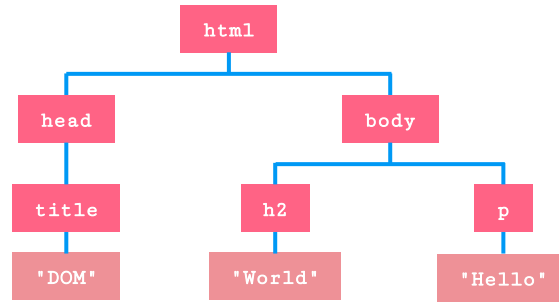
bit.ly/html-node

HTML nodes
reference

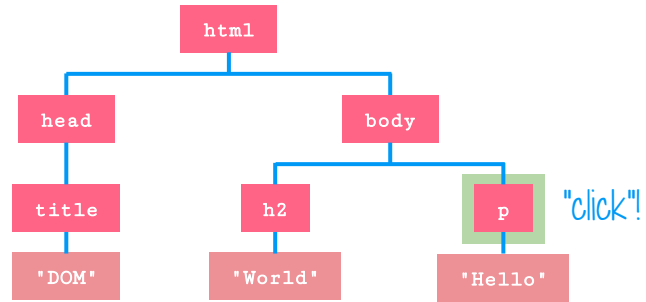
Challenge Set 4

javascript-workshop/04-dom

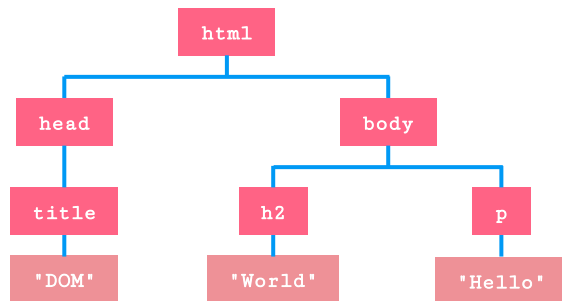
Javascript Events



events



events



```
var p = document.querySelector('p');
```

events

```
subject.addEventListener(name, callback);
```

events

```
p.addEventListener(name, callback);
```

events

```
p.addEventListener('click', callback);
```

events

```
p.addEventListener('click', function(e) {  
    // handle click event  
});
```

events

```
p.addEventListener('click', function(e) {  
    // handle click event  
});
```

events

Challenge Set 5

javascript-workshop/05-events

Part III

Bonus Part: Web APIs

Canvas API

bit.ly/canvas-api

Geolocation API

bit.ly/geolocation-api

Notifications API

bit.ly/notifications-api

bit.ly/all-web-apis

web apis are cool

The End

(ノ◉▽◉)ノ*:・° ✧ ✧°.: *ゝ(◉▽◉ゝ)

 /tilde-community/javascript-workshop

For updates and future talks
@tildecommunity

Special thanks to

ChannelFix.com