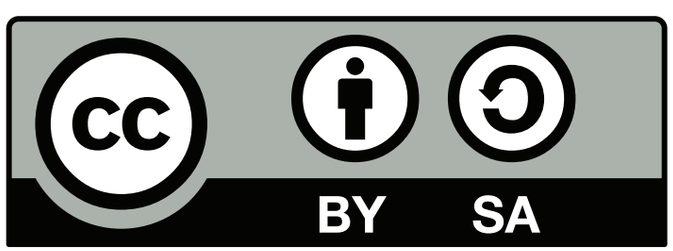
Applicazioni Web 2013/14

Lezione 3 - JavaScript

Matteo Vaccari
http://matteo.vaccari.name/
matteo.vaccari@uninsubria.it

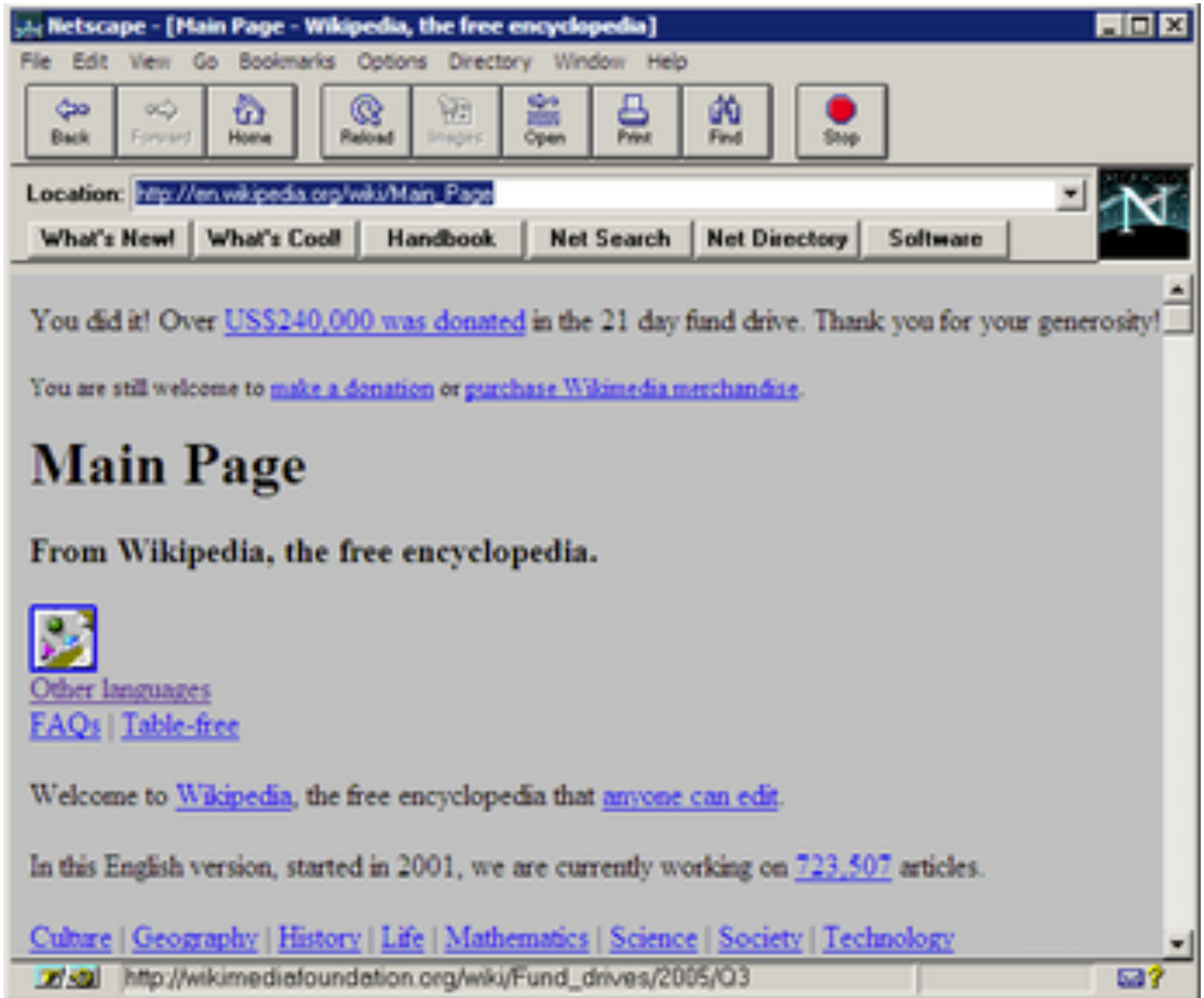


JavaScript? Why?

- Rich web applications
- Server side 
- Mobile
- Desktop
- ....

JavaScript: what?

Netscape Navigator 2.0, September 1995



JavaScript

- C-like syntax
- Dynamic typing
- Functional
- Object-based

C-like syntax

```
var x;
var y = 2;

console.log("Hello world!");
```

```
function factorial(n) {
    if (n === 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

JavaScript scope

```
var foo = 3; // global scope

function f() {
  var foo = 3; // function scope
}

bar = 33;  // se non esisteva, viene automaticamente definita
(EVITARE!)

function f() {
  baz = 123; // viene definita globalmente! (EVITARE!)
}
```

Dynamic typing

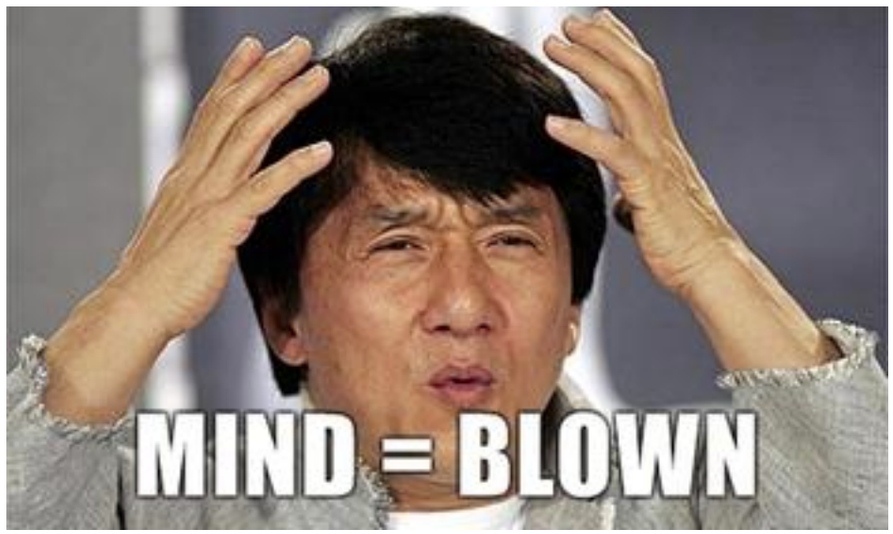
```
function f(a, b, c) {
  return a + b + c;
}

f(1, 2, 3) // returns 6
f("a", "b", "c") // returns "abc"
```

```
function sum() {
  var result = arguments[0];
  for (var i=1; i<arguments.length; i++) {
    result += arguments[i];
  }
  return result;
}

sum(3,4) // returns 7
sum("pippo", "pluto") // returns "pippopluto"
```

Functional

```
function square(x) { return x*x; }

// same as:
var square = function(x) { return x*x; }

[1, 2, 3, 4].map(square) // returns [1, 4, 9, 16]
```

```
// anonymous functions
[1, 2, 3, 4].map(function(x) {
  return x*x*x;
}); // returns [1, 8, 27, 64]
```



Esercizio

Scrivi una funzione *repeat* che faccia questo:

```
repeat(3, function() { console.log("Hello!") });
Hello!
Hello!
Hello!
```

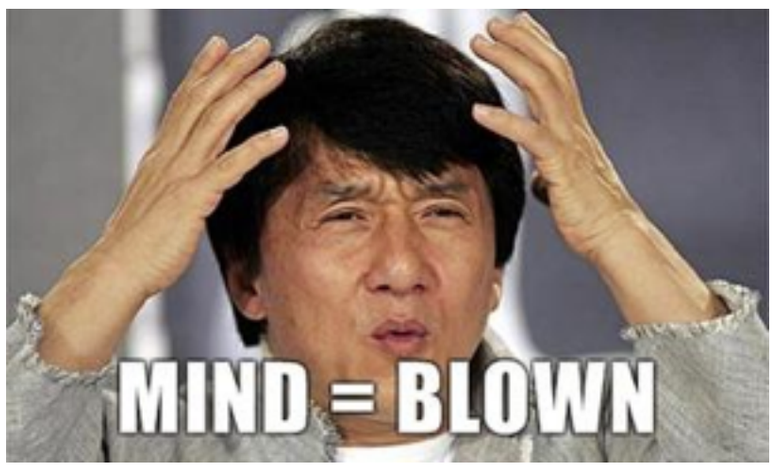
Closures

```
function multiplier(x) {
  return function(y) {
    return x*y;
  }
}
```

```
// remember x is 3
function(y) {
  return x*y;
}
```

```
var multiply_by_3 = multiplier(3);

multiply_by_3(4)
12

var multiply_by_pi = multiplier(3.14159);
multiply_by_pi(4)
12.56636

multiplier(9)(8);
72
```

```
function(y) {
  return 3*y;
}
```



Object Based

```
var an_object = {};
an_object.foo = "bar";
an_object["pi"] = 3.14159;

this.assertEquals("bar",   an_object.foo);
this.assertEquals("bar",   an_object["foo"]);
this.assertEquals(3.14159, an_object.pi);
```

Functions as properties

```
var an_object = {};

an_object.a = 6;
an_object.b = 7;

anObject.times = function() {
    return this.a * this.b;
}

this.assertEquals(42, an_object.times());
```

JavaScript Object Notation (JSON)

```
var an_object = {
    a: 256,
    b: 256,
    times: function() {
        return this.a * this.b;
    },
};

this.assertEquals(65536, an_object.times());
```

Object-Oriented

```
function Teacher(name, subject) {
    this.name = name;
    this.subject = subject;
}

var pippo = new Teacher("Pippo de Pippis", "epistemologia");
var paperino = new Teacher("Paolino Paperino", "astronavigazione");

this.assertEquals("epistemologia", pippo.subject);
this.assertEquals("Paolino Paperino", paperino.name);
```

Methods and prototypes

```
function Teacher(name, subject) {
    this.name = name;
    this.subject = subject;
}
```

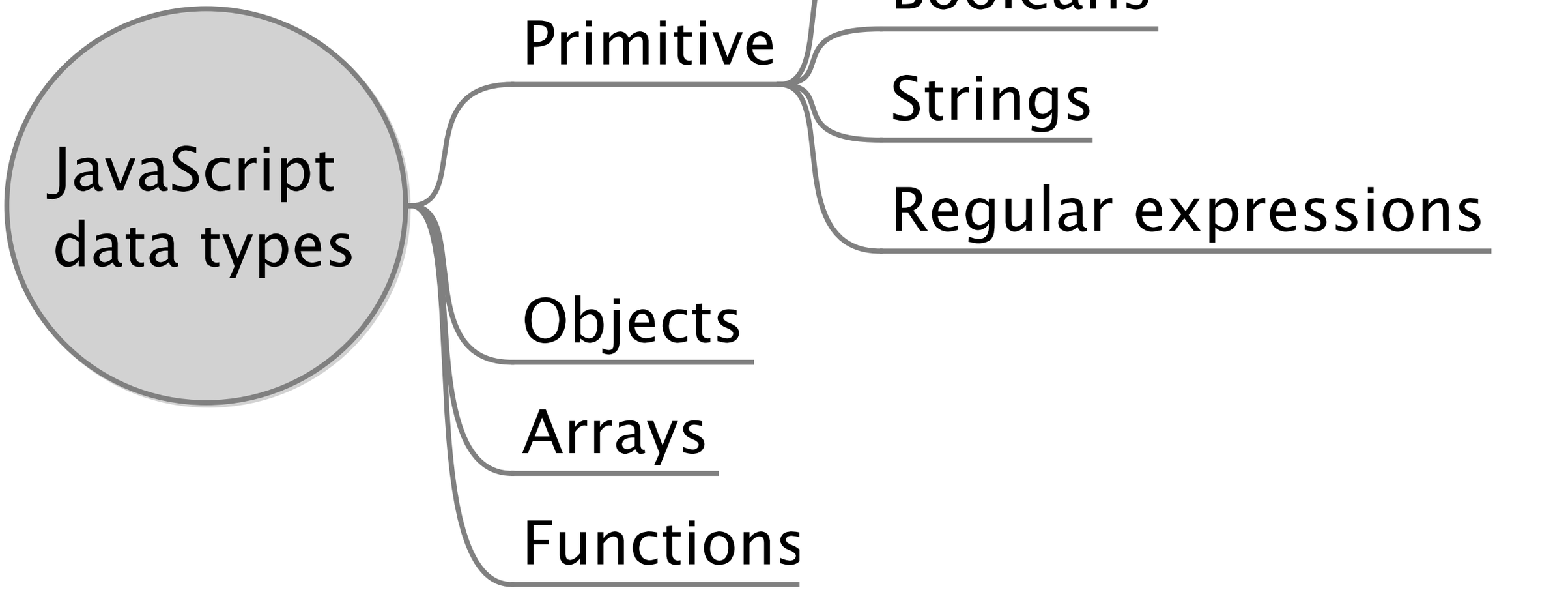
```
var pippo = new Teacher("Pippo de Pippis", "epistemologia");
var paperino = new Teacher("Paolino Paperino", "astronavigazione");

Teacher.prototype.description = function() {
    return this.name + " insegna " + this.subject;
}

this.assertEquals("Paolino Paperino insegna astronavigazione",
                  paperino.description());
this.assertEquals("Pippo de Pippis insegna epistemologia",
                  pippo.description());
```

Attach a method to an object

```
function Person(name) { this.name = name; }
var pippo = new Person("Pippo");

// associo il metodo al "prototipo" del costruttore
Person.prototype.sayHello = function() {
    return "Ciao da " + this.name;
}
this.assertEquals("Ciao da Pippo", pippo.sayHello());

// associo il metodo a un oggetto
pippo.toUpper = function() {
    return this.name.toUpperCase();
}
this.assertEquals("PIPPO", pippo.toUpper());
```

Prototype inheritance

```
function User(name, password) {
    this.name = name;
    this.password = password;
}
// User estende Person
User.prototype = new Person();

var pluto = new User("Pluto", "secret");
this.assertEquals("Ciao da Pluto", pluto.sayHello());
```


JavaScript data types
Primitive
Floating point numbers
Booleans
Strings
Regular expressions
Objects
Arrays
Functions

Arrays

```
var a = new Array();
a[0] = 123;
a[1] = "foo";
```

```
this.assertEquals(123, a[0]);
this.assertEquals("foo", a[1]);
this.assertEquals(undefined, a[2]);

b = [123, "foo"];
this.assertEquals(a, b);
```

Two ways to execute JavaScript in the browser

```
<script type="text/javascript">
  alert("hello!");
</script>
```

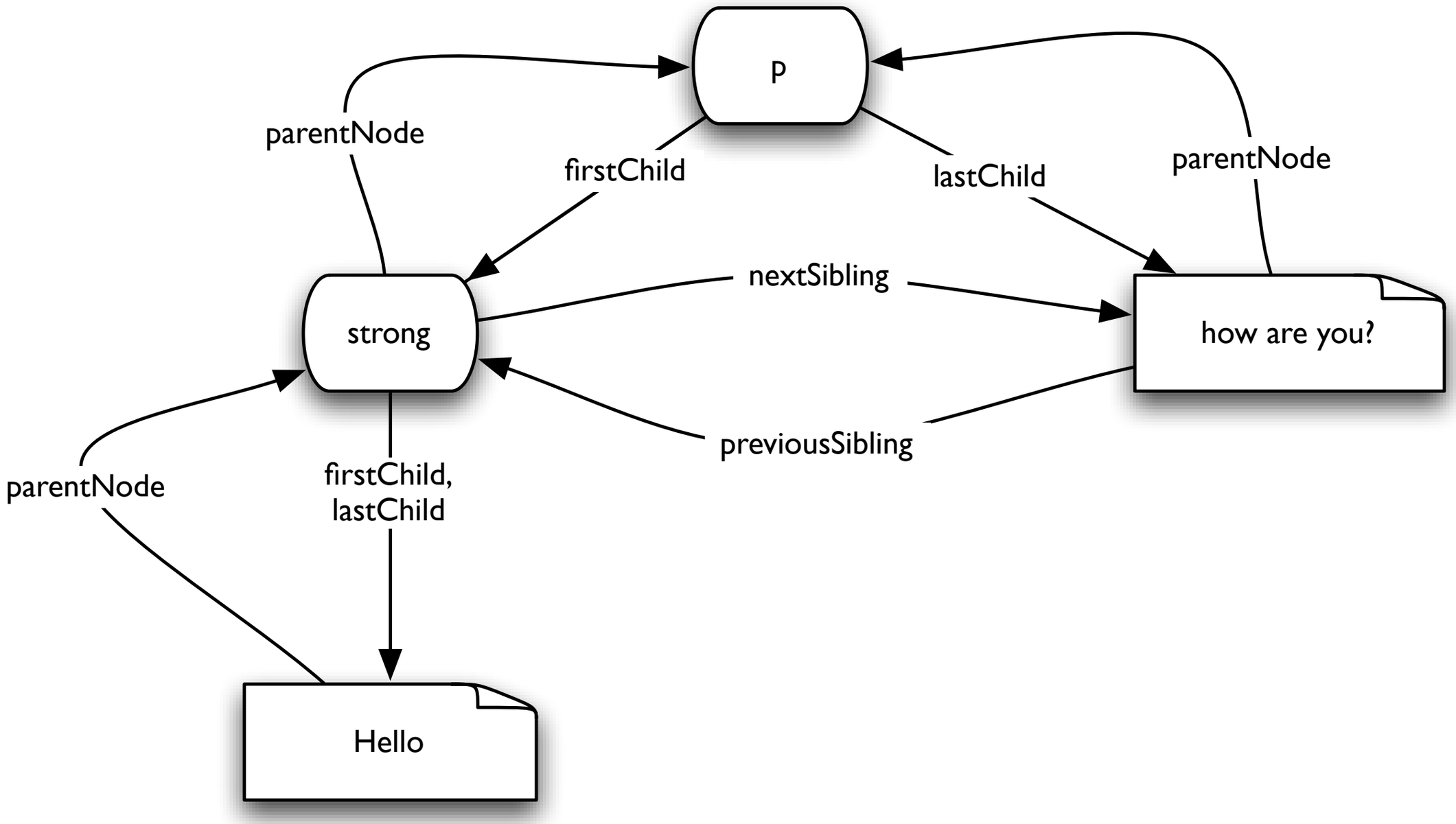
```
<script type="text/javascript" src="lib/testcase.js"></script>
```

What JavaScript can't do

- Access the file system
- Open arbitrary sockets

The Document Object Model

Hello how are you?



```html
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <h1 id="hello">Hello, World!</h1>
    ...
  </body>
</html>
```

```
// how to access the h1 element?  This does not work!!
document.documentElement  // returns the html element
  .firstChild             // should return the head element?
  .nextSibling            // should return the body element?
  .firstChild             // should return the h1 element?
```

The first child of the *html* element is a *whitespace* text node!

```
function next(elem) {
    do {
        elem = elem.nextSibling ;
    } while (elem && elem.nodeType != 1);
    return elem;
}

function first(elem) {
    elem = elem.firstChild;
    if (elem && elem.nodeType != 1) {
        return next(elem);
    } else {
        return elem;
    }
}

var h1 = first(next(first(document.documentElement)));
this.assertEquals("Hello, World!", h1.innerHTML);
```

Using utility functions to navigate the DOM

A *much* easier way to navigate the DOM

```
var h1 = document.getElementById("hello");
this.assertEquals("Hello, World!", h1.innerHTML);
```

```
var allHeadings = document.getElementsByTagName("h1");
this.assertEquals(1, allHeadings.length);
this.assertEquals("Hello, World!", allHeadings[0].innerHTML);
```

Working with attributes

```
var h1 = document.getElementById("hello");
h1.style.background = "red";
```

Changing the content of an element

```
var list = document.getElementById("list");
list.innerHTML = "<li>foo</li>"                          // replace contents
list.innerHTML += "<li>bar</li>"                         // append contents
list.innerHTML = "<li>zot</li>" + list.innerHTML;        // prepend contents
```

Browser events

Executing when the page is fully loaded

```
window.onload = addBorderToHello;

function addBorderToHello() {
    document.getElementById("hello").style.border = "1px solid green";
}
```

Working with links

```
<ul id="list">
</ul>
```

```
<script type="text/javascript">
function doSomething() {
  document.getElementById("list").innerHTML += "<li>another</li>";
}
</script>

<a href="#" onclick="doSomething();">Click Me</a>
```

Working with forms

```
<p id="validation"></p>

<form id="my-form" action="" method="get">
  <input id="foo" type="text" name="foo" value="" />
  <br />
  <input type="submit" />
</form>
```

```
<script type="text/javascript" charset="utf-8">
  var form = document.getElementById("my-form");
  form.onsubmit = function() {
      var el = document.getElementById("foo");
      var msg = document.getElementById("validation");
      if (el.value.match(/^[a-zA-Z$][a-zA-Z0-9]*$/)) {
          msg.innerHTML = "ok";
      } else {
          msg.innerHTML = "not a valid JavaScript identifier";
      }
      return false; // do not execute action
  }
</script>
```

A simple demo

```
<script type="text/javascript" charset="utf-8">
  function factorial(n) {
    if (n == 0)
      return 1;
    else
      return n * factorial(n-1);
}
...
```

```
function table_row(cell0, cell1, cell2) {
  return "<tr><td>" + cell0 + "</td>" +
                "<td>" + cell1 + "</td>" +
                "<td>" + cell2 + "</td></tr>";
}
```

```
document.write("<table>")
for (i=0; i<30; i++) {
  document.write(table_row(i + "!", "=", factorial(i)));
  }
  document.write("</table>")
</script>
```

0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000
21! = 51090942171709440000
22! = 1.1240007277776077e+21
23! = 2.585201673888498e+22
24! = 6.204484017332394e+23
25! = 1.5511210043330986e+25
26! = 4.0329146112660565e+26
27! = 1.0888869450418352e+28
28! = 3.04888344611713 84e+29
29! = 8.841761993739701e+30

```
<script type="text/javascript" charset="utf-8">
  function factorial(n) {
    if (n == 0)
      return 1;
    else
      return n * factorial(n-1);
  }

  function table_row(cell0, cell1, cell2) {
    return "<tr><td>" + cell0 + "</td>" +
           "<td>" + cell1 + "</td>" +
           "<td>" + cell2 + "</td></tr>";
  }

  document.write("<table>")
  for (i=0; i<30; i++) {
    document.write(table_row(i + "!", "=", factorial(i)));
  }
  document.write("</table>")
</script>
```