CSI 3540 - LECTURE 04

JavaScript

OBJECTIFS

- Chapitre 6-11 du manuel
- Écrire des programmes JavaScript
- Opérateurs, conditionnels et structures de bouclage
  - if, if/else, switch
  - while, do/while
  - for, for/in
- Les fonctions en JavaScript
- Méthodes globale
- Tableaux “Array”
- Objets

https://dart.dev/

https://clojurescript.org/

https://coffeescript.org/

https://www.typescriptlang.org/

https://en.wikipedia.org/wiki/VBScript

https://github.com/lumen/lumen

https://www.webcomponents.org/

https://webassembly.org/

https://en.wikipedia.org/wiki/ECMAScript

Netscape et Microsoft
Développé par ECMA International
Anciennement connue sous le nom "European Computer Manufacturers Association"
Maintenant appelé ECMAScrip

Hello World!

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Welcome</title>
  </head>
  <body>
    <script type="text/javascript">
      // Don't do this!
      document.writeln("<p>Welcome to HTML5!</p>");
    </script>
  </body>
</html>
```

Script en ligne

Très grossier, ne faites pas cela en production

Welcome to HTML5!

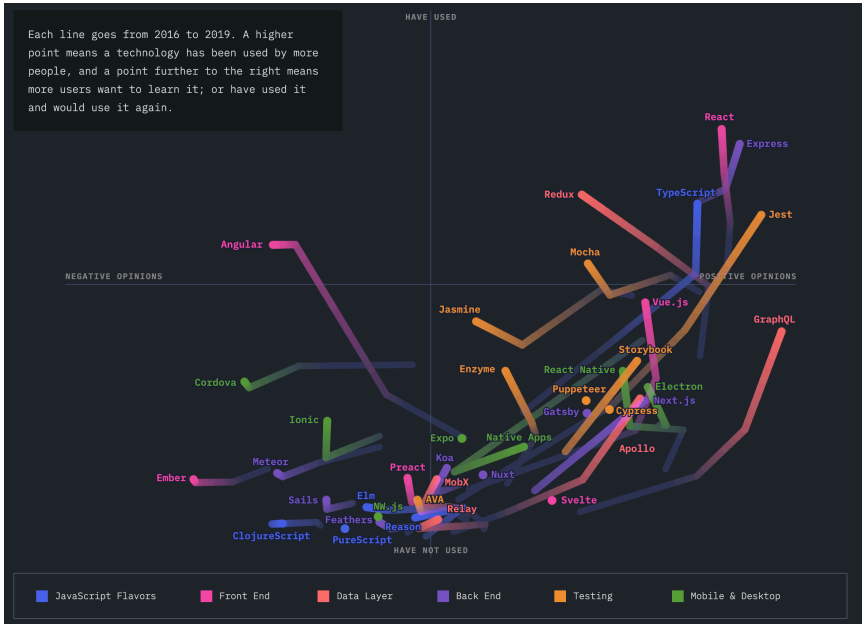
'b' + 'a' + + 'a' + 'a'

```
'b' + 'a' + + 'a' + 'a'
```

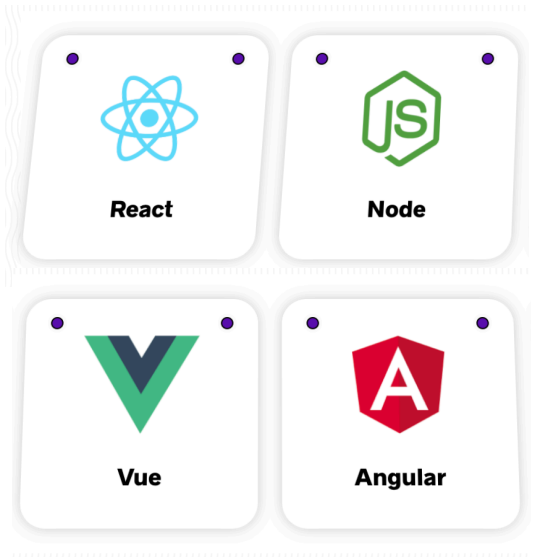
```
"baNaNa"
```


C#
JavaScript
Rust
C++
Python

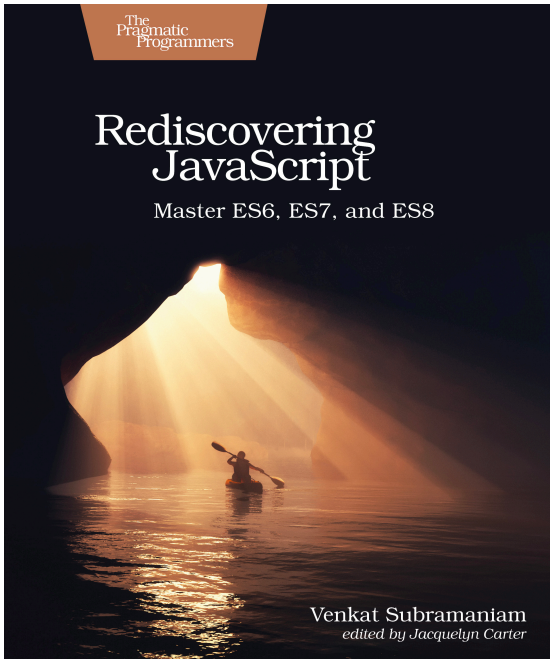
Version	Année
ECMAScript 1	June 1997
ECMAScript 2	June 1998
ECMAScript 3	December 1999
ECMAScript 5 (ES5)	December 2009
ECMAScript 2015 (ES6)	2015 (d'uh)
ECMAScript 2016	2016
...	...
ES.Next	



https://2019.stateofjs.com/overview/



https://beginnerjavascript.com/


The Pragmatic Programmers
Rediscovering JavaScript
Master ES6, ES7, and ES8
Venkat Subramaniam
edited by Jacquelyn Carter


The Pragmatic Programmers
Test-Driving JavaScript Applications
Rapid, Confident, Maintainable Code
Venkat Subramaniam
Edited by Jacquelyn Carter

Comment "écrire" à l'écran ?

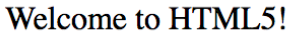
Comment "écrire" à l'écran ?

```
document.writeln("<p>Hi</p>")
```

Comment "écrire" à l'écran ?

```
document.writeln("<p>Hi</p>")
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Welcome</title>
  </head>
  <body>
    <script type="text/javascript">
      // Don't do this!
      document.writeln("<p>Welcome to HTML5!</p>");
    </script>
  </body>
</html>
```



Script en-ligne

```
<script type="text/javascript">
  // Don't do this!
  document.writeln(
    "<p>Welcome to HTML5!</p>");
</script>
```

Encore une fois, ne "writelin" dans production

Welcome to HTML5!

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Welcome</title>
    <!-- Load your libraries at the top -->
    <script type="text/javascript" src="02_library.js"></script>
  </head>
  <body>

    <!-- And your event processing at the bottom -->
    <script type="text/javascript" src="02_events.js"></script>
  </body>
</html>
```

Utilisez des fichiers JavaScript externes autant que possible

```
function helloWorld()
{
  document.writeln("<p>Welcome to HTML5!</p>");
}
```

Ficher JavaScript externe

Le fichier externe est JavaScript, alors vous ne mélangez pas avec l'HTML

```
function helloWorld()
{
  document.writeln("<p>Welcome to HTML5!</p>");
}
```

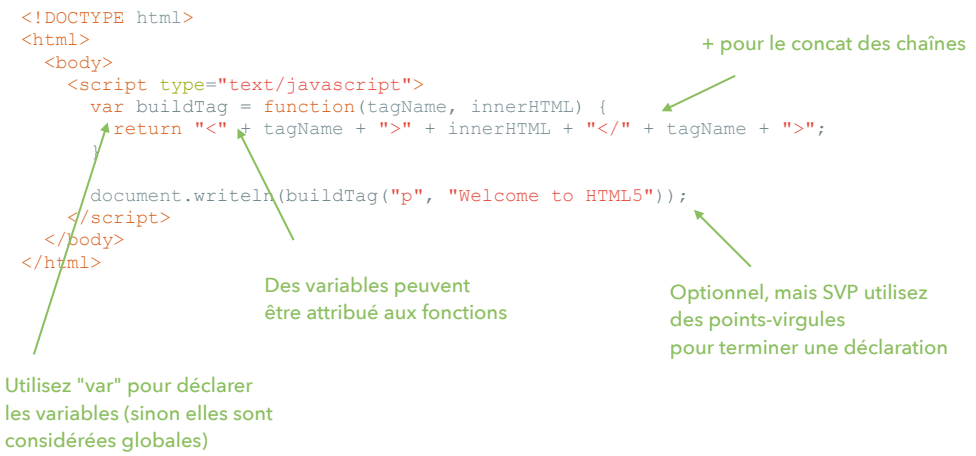
Concaténation des chaînes...

```
"Hi, " + name;
```

Gabarit...

```
`Hi, ${name}`;
```

Interpolation de javascript dans une chaîne.




```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">
      var buildTag = function(tagName, innerHTML) {
        return "<" + tagName + ">" + innerHTML + "</" + tagName + ">";
      }

      document.writeln(buildTag("p", "Welcome to HTML5"));
    </script>
  </body>
</html>
```

Concaténation des chaînes

```
let buildTag = function(tagName, innerHTML) {
  return "<"
    + tagName
    + ">"
    + innerHTML
    + "</" + tagName + ">";
}

document.writeln(
  buildTag(
    "p",
    "Welcome to HTML5"
  )
);
```

Réécrire en utilisant l'interpolation

```
function(tagName, innerHTML) {
  return "<"
    + tagName
    + ">"
    + innerHTML
    + "</" + tagName + ">";
}
```

```
function(tagName, innerHTML) {
  return `<${tagName}>${innerHTML}
</${tagName}>`;
}
```

```
var letters = "OK";
var snake_case = "OK, but unconventional";
var $beCareful = "OK, conflicts with jQuery";
var numbers1 = "OK, but not at the start";
// 133t -> pas OK, ne peut pas commencer avec un chiffre
```

Création de variables.

Variables ...

Variables ...

```
a = "i global";
var b = "kinda global";
```

Variables ..

```
a = "i global";
var b = "kinda global";
```

Variables …

```
let a = "local";
const b = "somewhat can’t change";
```

Calculer le coût total d'un milk-shake de $100 (taxe de 13 %)

Calculer le coût total d'un milk-shake de $100 (taxe de 13 %)

```
const tax_rate = 0.13;
const total_cost = 100 * (1 + tax_rate);
```

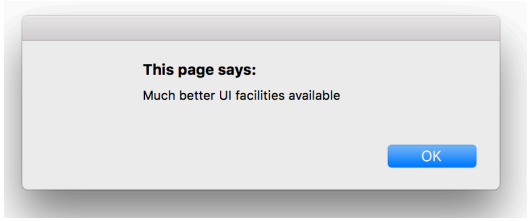
Calculez le total des achats, qui comprend un milk-shake de 100 $ (taxe de 13 %)

Calculez le total des achats, qui comprend un milk-shake de 100 $ (taxe de 13 %)

```
const tax_rate = 0.13;
let total_cost = 0;

...

total_cost += 100 * (1 + tax_rate);
```

ALERT

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">
      // Don't use!
      window.alert("Much better UI facilities available");
    </script>
  </body>
</html>
```

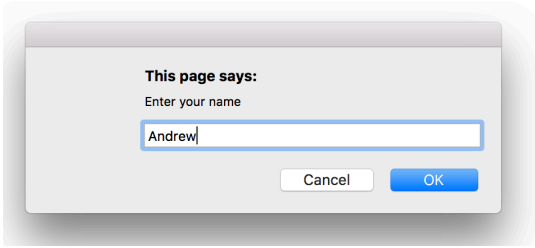


CARATÈRE "ESCAPE"

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">
      document.writeln("<p>Normal \"Escape\" with Slash \\</p>");
      document.writeln('<p>Or use \' to build your string when you want "');
      document.writeln("<pre>One\n\tTwo</pre>");
    </script>
  </body>
</html>
```

PROMPT

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">
      // Don't use!
      var name = window.prompt("Enter your name");
      document.writeln("<h1>Hello " + name + "</h1>");
    </script>
  </body>
</html>
```



```
var a;
isNull("a (new var)", a);

a = null;
isNull("a (assigned null)", a);

a = "";
isNull("a (assigned empty string)", a);

delete a;
isNull("a (deleted)", a);
```

a (new var) is null

a (assigned null) is null

a (assigned empty string) is NOT null; value is []

a (deleted) is NOT null; value is []

```
var a = 10;
var b = 15;
var sum = a + b;
document.writeln("As integers: " + a + " + " + b + " = " + sum);

document.writeln("<br>");

a = "10";
b = "15";
sum = a + b;
document.writeln("As strings: " + a + " + " + b + " = " + sum);
```

As integers: 10 + 15 = 25
As strings: 10 + 15 = 1015

```
document.writeln("<pre>");
document.writeln("ADD:  10 + 5 = " + (10 + 5) + "\n");
document.writeln("SUB:  10 - 5 = " + (10 - 5) + "\n");
document.writeln("MULT: 10 * 5 = " + (10 * 5) + "\n");
document.writeln("DIV: 10 / 5 = " + (10 / 5) + "\n");
document.writeln("REM: 11 % 5 = " + (11 % 5) + "\n");
document.writeln("</pre>");
```

```
document.writeln("<pre>");
document.writeln("ADD:  10 + 5 = " + (10 + 5) + "\n");
document.writeln("SUB:  10 - 5 = " + (10 - 5) + "\n");
document.writeln("MULT: 10 * 5 = " + (10 * 5) + "\n");
document.writeln("DIV: 10 / 5 = " + (10 / 5) + "\n");
document.writeln("REM: 11 % 5 = " + (11 % 5) + "\n");
document.writeln("</pre>");
```

```
ADD:  10 + 5 = 15

SUB:  10 - 5 = 5

MULT: 10 * 5 = 50

DIV: 10 / 5 = 2

REM: 11 % 5 = 1
```

```
var date = new Date();
if (date.getHours == 11) {
  document.writeln("Current time is 11 o'clock\n");
} else {
  document.writeln("Current time ("+ date +") is NOT 11 o'clock\n");
}
```

Un objet pour obtenir des informations sur l'heure actuelle

Current time (Sat Jan 07 2017 19:05:01 GMT-0500 (EST)) is NOT 11 o'clock

Condition IF

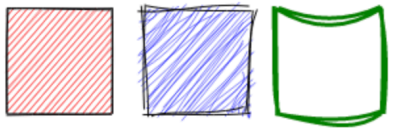
MOMENT.JS

```
<!DOCTYPE html>
<html>
  <head>
    <script src="moment.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      var showDate = function(prefix, m) {
        document.writeln(prefix + ": " + m.format('MMMM Do YYYY, h:mm:ss a') + "<br>");
      }

      var today = moment();
      showDate("NOW", today);
      showDate("LAST WEEK", today.subtract(1, 'weeks'));
      showDate("TOMORROW", today.add(1, 'day'));
    </script>
  </body>
</html>
```

NOW: January 7th 2017, 7:49:45 pm
LAST WEEK: December 31st 2016, 7:49:45 pm
TOMORROW: January 1st 2017, 7:49:45 pm

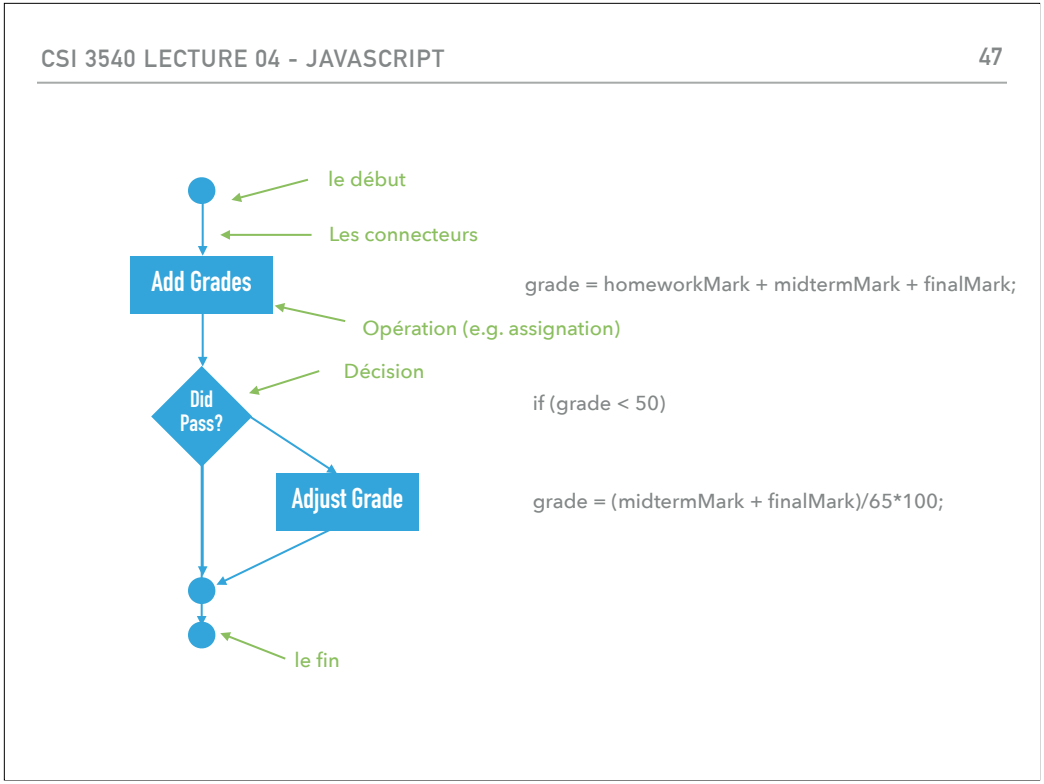
ROUGH.JS

Sketching style



```
rc.rectangle(15, 15, 80, 80, { roughness: 0.5, fill: 'red' });
rc.rectangle(120, 15, 80, 80, { roughness: 2.8, fill: 'blue' });
rc.rectangle(220, 15, 80, 80, { bowing: 6, stroke: 'green', strokeWidth: 3 });
```

https://roughjs.com/posts/release-4.0/



Algorithme - avec un diagram de flux (flowchart)

Une méthode qui peut être exprimée dans une quantité finie d'espace et de temps et dans un langage formel bien défini pour calculer une fonction
* Trier une list d’entiers
* Résolution des tours de Hanoi
* Calcul de tax sur une facture
Pour décrire un algorithme en JavaScript, votre code est une combinaison de
* Séquence
* Sélection
* Répétition

De haut en bas

- Commencez par pseudo code et convertissez en code réel
- A chaque itération, écrivez des instructions plus explicites
- Trois phase
  - Initialisation (Préparez vos entrées et vos structures de données intermédiaires)
  - Traitement (L'algorithm)
  - Terminaison (Préparez les résultats)

Erreurs

- Une erreur logique (“logic error”) a son effet au moment de l'exécution.
- Une erreur logique fatale (“fatal logic error”) provoque un échec du script et se termine prématurément.
- Une erreur de logique non fatale (“nonfatal logic error”) permet à un script de continuer à s'exécuter, mais le script génère des résultats incorrects.

break	case	catch	continue	default
delete	do	else	false	finally
for	function	if	in	instanceof
new	null	return	switch	this
throw	true	try	typeof	var
void	while	with	let	

Mots résérvés en JavaScript

Toutes les opérations qui renvoient “vraie-sy” vont dans le bloc IF

```
if (grade >= 90) {
  console.log("A+");
}
```

Pratique pour que plus que juste **FALSE** soit considéré comme une valeur "faux-sy"

```
if (false) {
  err();
} else {
  console.log("false is falsey");
}
```

```
if ("") {
  err();
} else {
  console.log("Empty string is falsey");
}
```

```
if (0) {
  err();
} else {
  console.log("0 (and -0) integers are falsey");
}
```

```
if (null) {
  err();
} else {
  console.log("null falsey");
}
```

```
if (undefined) {
  err();
} else {
  console.log("undefined falsey");
}
```

```
if (NaN) {
  err();
} else {
  console.log("NaN falsey");
}
```

Si vous n'êtes pas “faux-si”, alors vous êtes “vrai-si”

```
if ("0") {
 console.log("Is true?");
} else {
  console.log("Or false");
}
```

Seul "false" évalue vrai pour false === false

```
if (false === 0) {
 console.log("Is true?");
} else {
  console.log("Or false");
}
```

```
if (grade >= 90) {
  console.log("A+");
} else {
  console.log("You received " + grade);
}
```

If / else

Plusieurs déclarations est soutenu

```
var a, b;

b = null;
a = b ? b : "99";
console.log(a);

b = "101";
a = b ? b : "99";
console.log(a);

// NOT SUPPORTED IN JavaScript
// b = null;
// a = b ?: "99";
// console.log(a);
```

```
if (b) {
  a = b;
} else {
  a = “99”;
}
```

```
99

101
```

Même une syntaxe plus courte disponible dans certains languages (comme PHP) mais pas JavaScript

If / else “shorthand”

```
if (grade >= 90) {
  console.log("A+");
} else if (grade >= 85) {
  console.log("A");
} else if (grade >= 80) {
  console.log("A-");
} else if (grade >= 75) {
  console.log("B+");
} else if (grade >= 70) {
  console.log("B");
} else if (grade >= 65) {
  console.log("C+");
} else if (grade >= 60) {
  console.log("C");
} else if (grade >= 55) {
  console.log("D+");
} else if (grade >= 50) {
  console.log("D");
} else {
  console.log("E");
}
```

If / else if / else

```
var text = null;
switch (new Date().getDay()) {
  case 4:
  case 5:
    text = "Soon it's the Weekend";
    break;
  case 0:
  case 6:
    text = "It is Weekend";
    break;
  default:
    text = "Looking forward to the Weekend";
}
console.log(text);
```

```
function weekendThoughts(date) {
  switch (date.getDay()) {
    case 4:
    case 5:
      return "Soon it's the Weekend";
    case 0:
    case 6:
      return "It is Weekend";
    default:
      return "Looking forward to the Weekend";
  }
}

var text = weekendThoughts(new Date());
console.log(text);
```

Plus concis si “switch” est délégué à fonction séparée

Plus testable aussi, comme nous n'avons pas à prendre une "semaine" pour exécuter manuellement tous les cas

Meilleur switch

Controlled repetition based on a predicate
(We know it will do 5 loops – 0, 3, 6, 9, 12)

```
let i = 0;
while (i < 10) {
  i += 3;
}
console.log("The value of i is " + i);
```

The value of i is 12.

While

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">
      var level = 0;
      do {
        level += 1;
        document.writeln("<h"+ level +">Welcome to HTML5!</h"+ level +">");
      } while (level < 6);
    </script>
  </body>
</html>
```

Utile lorsque vous voulez toujours entrer la boucle au moins une fois

Do / while

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">
      for (var level = 1; level < 7; level++) {
        document.writeln("<h"+ level +">Welcome to HTML5!</h"+ level +">");
      }
    </script>
  </body>
</html>
```

Variable de contrôle

Condition à garder en boucle

Incrément de variable

For

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">

      var countBy = 18;
      var maxNum = 100;
      var count = 0;
      document.writeln("Counting by " + countBy + " up to a max of " + maxNum);
      while (true) {
        count += 1;
        if (count > maxNum) {
          break;
        } else if (count % countBy != 0) {
          continue;
        } else {
          document.writeln("<li>" + count + "</li>");
        }
      }
    </script>
  </body>
</html>
```

Une boucle infinie

Arrêter le bouclage

Continuer à la prochaine boucle

Counting by 18 up to a max of 100

- 18
- 36
- 54
- 72
- 90

Break / Continue

Valeur Sentinel

- Une valeur spéciale pour indiquer la fin des entrée
- Également appelée valeur de signal, valeur “dummy” ou valeur “flag”
- Est souvent appelé répétition indéfinie, parce que le nombre de répétitions n'est pas connu à l'avance
- Choisissez une valeur de sentinelle qui ne peut pas être confondue avec une valeur d'entrée acceptable
- Évitez les valeurs sentienl, car elles peuvent être stupides

Converti une chaîne en entier

Cherchez pour “Not a Number”

```
var rawInput = "89";

var grade = parseInt(rawInput);

if (isNaN(grade)){
  console.log("Unable to process grade " + rawInput);
} else {
  grade += 3;
  console.log("After the 3% bonus, your grade is now " + grade);
}
```

parseInt

```
var c;

c = 10;
c += 3;
console.log("C is " + c);

c = 10;
c -= 3;
console.log("C is " + c);

c = 10;
c *= 3;
console.log("C is " + c);

c = 10;
c /= 5;
console.log("C is " + c);
```

c = c + 3

c = c - 3

c = c * 3

c = c / 3

C is 13

C is 7

C is 30

C is 2

Assignation

Inc de plus 1 avant évaluation de < 3

```
i = 0
while (++i < 3) {}
console.log("++i loop, i is now " + i);

i = 0
while (i++ < 3) {}
console.log("i++ loop, i is now " + i);
```

Inc après évaluation de < 3

++i loop, i is now 3

i++ loop, i is now 4

Même approche pour
le décrément (--i ou i--)

Incrémentation

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">
      var answer = Math.pow(3, 2);
      document.writeln("3^2 = " + answer);
    </script>
  </body>
</html>
```

3^2 = 9

Math.pow

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">

      var a = "OK";
      var b = "";

      if (a) {
        document.writeln("a is TRUE<br>");
      } else {
        document.writeln("a is FALSE<br>");
      }

      if (b) {
        document.writeln("b is TRUE<br>");
      } else {
        document.writeln("b is FALSE<br>");
      }

      if (a && b) {
        document.writeln("a && b is TRUE<br>");
      } else {
        document.writeln("a && b is FALSE<br>");
      }

      if (a || b) {
        document.writeln("a || b is TRUE<br>");
      } else {
        document.writeln("a || b is FALSE<br>");
      }

      if (!a) {
        document.writeln("!a is TRUE<br>");
      } else {
        document.writeln("!a is FALSE<br>");
      }

      if (!b) {
        document.writeln("!b is TRUE<br>");
      } else {
        document.writeln("!b is FALSE<br>");
      }

    </script>
  </body>
</html>
```

a is TRUE
b is FALSE
a && b is FALSE
a || b is TRUE
!a is FALSE
!b is TRUE

Opérateurs logiques

```
<!DOCTYPE html>
<html>
  <body>
    <script type="text/javascript">

      var a = "OK";
      var b = "";

      if (!!a) {
        document.writeln("!!a is TRUE<br>");
      } else {
        document.writeln("!!a is FALSE<br>");
      }

      if (!!a == a) {
        document.writeln("!!a == a is TRUE<br>");
      } else {
        document.writeln("!!a == a is FALSE<br>");
      }

      if (!!b == b) {
        document.writeln("!!b == b is TRUE<br>");
      } else {
        document.writeln("!!b == b is FALSE<br>");
      }

      if (!!a === true) {
        document.writeln("!!a === true is TRUE<br>");
      } else {
        document.writeln("!!a === true is FALSE<br>");
      }

      if (!!b === false) {
        document.writeln("!!b === false is TRUE<br>");
      } else {
        document.writeln("!!b === false is FALSE<br>");
      }

    </script>
  </body>
</html>
```

!!a is TRUE
!!a == a is FALSE
!!b == b is TRUE
!!a === true is TRUE
!!b === false is TRUE

Cast à boolean

JavaScript Teacher
@js_tut

Array methods you should know:
[].map()
[].filter()
[].reduce() – sometimes used to calculate shopping cart total
[].some() – returns if at least one item matches condition
[].flat()
[].flatMap()
[].every()
[].sort() – used this one to create tables sorted by column

12:25 PM · Jan 15, 2020 · Twitter Web App

```
let entity = {};
entity.balance = 0.0;
entity.fees = 1.00;

function applyTransaction(delta) {
  entity.balance = entity.balance
    + delta
    - entity.fees;
}

applyTransaction(10);
```

```
let entity = {};
entity.balance = 0.0;
entity.fees = 1.00;

entity.showBalance = function() {
  document.writeln(`Balance ${this.balance}<br>`);
}

entity.showBalance();
```

```
var BankAccount = (function () {
  let entity = {};
  entity.balance = 0.0;
  return entity;
}());

BankAccount.blance;
```

Un module est juste une variable avec des propriétés (qui peuvent être des données ou des fonctions, nommé “méthodes”)

```
var BankAccount = (function () {
  var entity = {};
  entity.balance = 0.0;
  entity.fees = 1.00;

  function applyTransaction(delta) {
    entity.balance = entity.balance + delta - entity.fees;
  }

  entity.showBalance = function() {
    document.writeln("Balance " + this.balance + "<br />");
  }

  entity.deposit = function(amount) {
    document.writeln("Deposit " + amount + "<br />");
    applyTransaction(amount);
    this.showBalance();
    return this.balance;
  }

  entity.withdraw = function(amount) {
    document.writeln("Withdraw " + amount + "<br />");
    applyTransaction(-1 * amount);
    this.showBalance();
    return this.balance;
  }

  return entity;
}());

BankAccount.showBalance();
BankAccount.deposit(100.0);
BankAccount.withdraw(10.0);
```

Propriétés du module

Fonction privée

Les méthodes publique du modules

Ne faites pas cela, juste pour la démonstration

Un “closure” que nous appelons immédiatement

Balance 0

Deposit 100

Balance 99

Withdraw 10

Balance 88

Modules et Méthodes

```
var BankAccount = (function () {
  var entity = {};
  entity.balance = 0.0;
  entity.fees = 1.00;

  function applyTransaction(delta) {
    entity.balance = entity.balance + delta - entity.fees;
  }

  entity.showBalance = function() {
    document.writeln("Balance " + this.balance + "<br />");
  }

  entity.deposit = function(amount) {
    document.writeln("Deposit " + amount + "<br />");
    applyTransaction(amount);
    this.showBalance();
    return this.balance;
  }

  entity.withdraw = function(amount) {
    document.writeln("Withdraw " + amount + "<br />");
    applyTransaction(-1 * amount);
    this.showBalance();
    return this.balance;
  }

  return entity;
}());

BankAccount.showBalance();
BankAccount.deposit(100.0);
BankAccount.withdraw(10.0);
```

Balance 0

Deposit 100

Balance 99

Withdraw 10

Balance 88

"Ancien" module comme entrée

```
var Math = (function (module) {
  module.inc = function(num) {
    return num + 1;
  }
  return module;
}(Math))

document.writeln(`Math.inc(7) = ${Math.inc(7)}<br>`);
```

Passez dans le module existant dans la fermeture.

Augmentation (Monkey patching)

"Ancien" module comme entrée

```
var Math = (function (module) {
  module.inc = function(num) {
    return num + 1;
  }
  return module;
}(Math))
```

Passez dans le module existant dans la fermeture.

```
document.writeln(`Math.inc(7) = ${Math.inc(7)}<br>`);
```

Math.inc(7) = 8

Augmentation (Monkey patching)

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

function showFactorial(n) {
  var answer = factorial(n);
  document.writeln(`${n} != ${answer}<br>`);
}

showFactorial(-10);
showFactorial(0);
showFactorial(1);
showFactorial(2);
showFactorial(3);
showFactorial(4);
```

Retourne rien “undefined”

Retourne une valeur

Retourne à la fin de la fonction

Retour du contrôle

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

function showFactorial(n) {
  var answer = factorial(n);
  document.writeln(`${n} != ${answer}<br>`)
}

showFactorial(-10);
showFactorial(0);
showFactorial(1);
showFactorial(2);
showFactorial(3);
showFactorial(4);
```

Retourne rien “undefined”

Retourne une valeur

Retourne à la fin de la fonction

-10! = undefined

0! = 1

1! = 1

2! = 2

3! = 6

4! = 24

Retour du contrôle

```
function beCareful() {
  x = 99;
}

function buggyInc(y) {
  return x + 1;
}

beCareful();
var answer = buggyInc(35);

document.writeln(`buggyInc(35) = ${answer}<br>`);
```

Toujours utilize “let” (ou "const", ou "var")
pour declarer les variables,
sinon les bugs suivront

“x” est maintenant globale

Let, const (au moin), var toujours!

```
function beCareful() {
  x = 99;
}

function buggyInc(y) {
  return x + 1;
}

beCcareful();
var answer = buggyInc(35);

document.writeln(`buggyInc(35) = ${answer}<br>`);
```

buggyInc(35) = 100

Let, const (au moin), var toujours!

```
var Dice = (function() {
  var entity = {
    "history": []
  };

  entity.roll = function() {
    var nextValue = Math.floor(1 + Math.random() * 6);
    this.history.push(nextValue);
    return nextValue;
  }

  entity.showHistory = function() {
    if (this.history.length == 0) {
      document.writeln("<br>No die has been thrown yet.<br>");
    } else {
      document.writeln("<br>Dice rolls ("+ this.history.length +")<br>");
      for (var i = 0, len = this.history.length; i < len; i++) {
        document.writeln("<li>" + this.history[i] + "</li>");
      }
    }
  }

  return entity;
}());

Dice.showHistory();

Dice.roll();
Dice.roll();
Dice.roll();

Dice.showHistory();
```

Nombre fait au hasard entre 0.0 et 1.0 (mais pais 1.0)

Couper toute valeur decimale

No die has been thrown yet.

Dice rolls (3)

- 3
- 1
- 1

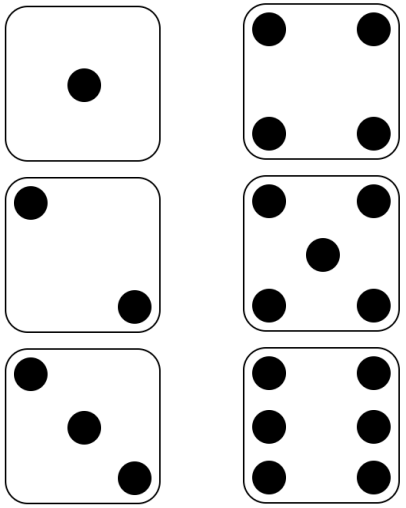
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Welcome</title>
    <style>
      .dice {
        margin: 10px;
        position: relative;
        width: 100px;
        height: 100px;
        border: 1px solid black;
        border-radius: 15px;
      }

      .dice .middle {
        position: absolute;
        top: 40px;
        left: 40px;
      }

      .dot {
        border: 1px solid black;
        background-color: black;
        border-radius: 50%;
        width: 20px;
        height: 20px;
      }

    </style>
  </head>
  <body>

    <div class="dice" />
      <div class="middle"><div class="dot"></div></div>
    </div>

  </body>
</html>
```

Le bordure des dés

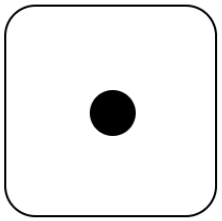
Absolu est "relatif" au parent, lorsque le parent est "relatif"

La position du point

Le point noir

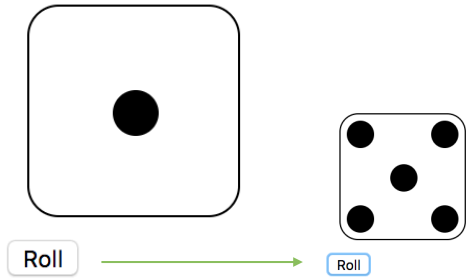


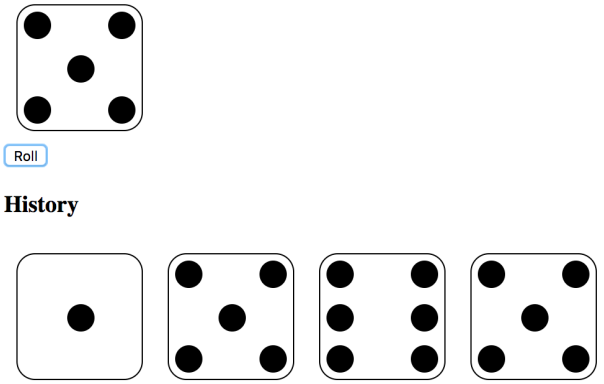
Dé CSS



Roll

History


Roll
History


Roll
History

```
var roller = document.getElementById("roller");
var rollHistory = document.getElementById("rollHistory");
```

When someone clicks the button, the "onclick" event / function is executed

```
roller.onclick = function() {
  var nextValue = Dice.roll();
  var nextHtml = Dice.showDie("die01", nextValue);
  rollHistory.innerHTML +=
    `<div class="dice-container">${nextHtml}</div>`;
}

Dice.showDie("die01", 1);
```

Plus sûr pour commencer à exécuter la configuration des événements lorsque le page est prêt

```
window.addEventListener("load", start, false);

function start() {

  var roller = document.getElementById("roller");
  var rollHistory = document.getElementById("rollHistory");

  roller.addEventListener(
    "click",
    function () {
      var nextValue = Dice.roll();
      var nextHtml = Dice.showDie("die01", nextValue);
      rollHistory.innerHTML += `<div class="dice-container">${nextHtml}</div>`;
    },
    false);

  Dice.showDie("die01", 1);
}
```

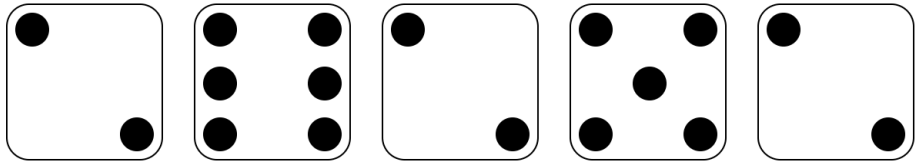
Peut être appliqué à n'importe quel élément DOM.

Peut être un nom de fonction, ou le fonction direct (un closure)

window.load

```
roller.onclick = function() {
    ...
}
```

Mais les appels direct sont généralement meilleurs



Roll

```
window.onload = function()
{
  var roller = document.getElementById("roller");
  roller.addEventListener(
    "click",
    function () {
      Dice.showDie("die01", Dice.roll());
      Dice.showDie("die02", Dice.roll());
      Dice.showDie("die03", Dice.roll());
      Dice.showDie("die04", Dice.roll());
      Dice.showDie("die05", Dice.roll());
    },
    false);

  Dice.showDie("die01", 0);
  Dice.showDie("die02", 0);
  Dice.showDie("die03", 0);
  Dice.showDie("die04", 0);
  Dice.showDie("die05", 0);
}
```

Lancer plusieurs dés

Roll

```
var Dice = (function() {
  var entity = { "min": 1, "max": 6, "history": [] };

  entity.roll = function() {
    var nextValue = Math.floor(this.min + Math.random() * this.max);
    this.history.push(nextValue);
    return nextValue;
  }

  function htmlDie(num) {
    var html = '<div class="dice" />';
    var pos = positions(num);

    if (Array.isArray(pos)) {
      for (var i = 0, len = pos.length; i < len; i++) {
        html += htmlDot(pos[i]);
      }
    } else {
       html += '<div class="middle">'+ num +'</div></div>'
    }

    html += '</div>';
    return html;
  }

  function htmlDot(position) {
    return '<div class="'+ position +'"><div class="dot"></div></div>';
  }

  function positions(num) {
    switch (num) {
      case 0: return [];
      case 1: return ["middle"];
      ...
      default: return num;

    }
  }

  return entity;
}());
```

Enregistrez les valeurs min et max.

Nombre aléatoire entre min et max

Si nous pouvons "dessiner" la face du dé, faites-le

Sinon imprimer le numéro

Pour un grand nombre, il suffit de retourner lui-même à la place de la position des points

Dés avec X faces

```
x = "GlobalX";
var y = "QuasiGlobalY";

function showY() {
  var y = "ScopedY"
  var p = "ScopedP"
  document.writeln("=== start showY =====<br>");
  document.writeln("x = " + x + "<br>");
  document.writeln("y = " + y + "<br>");
  document.writeln("p = " + p + "<br>");
  if (typeof(z) != "undefined") {
    document.writeln("z = " + z + "<br>");
  } else {
    document.writeln("z is undefined<br>");
  }
  document.writeln("=== end showY =====<br>");
}

function addZ()
{
  z = "AlsoGlobalZ";
}

showY();
addZ();
document.writeln("=== start main =====<br>");
document.writeln("x = " + x + "<br>");
document.writeln("y = " + y + "<br>");
document.writeln("z = " + z + "<br>");
if (typeof(p) != "undefined") {
  document.writeln("p = " + p + "<br>");
} else {
  document.writeln("p is undefined<br>");
}
document.writeln("=== end main =====<br>");
showY();
```

"x" est globale

"y" et "p" sont locaux à la fonction

"z" est global

Porté (scope)

```
x = "GlobalX";
var y = "QuasiGlobalY";

function showY() {
  var y = "ScopedY"
  var p = "ScopedP"
  document.writeln("=== start showY =====<br>");
  document.writeln("x = " + x + "<br>");
  document.writeln("y = " + y + "<br>");
  document.writeln("p = " + p + "<br>");
  if (typeof(z) != "undefined") {
    document.writeln("z = " + z + "<br>");
  } else {
    document.writeln("z is undefined<br>");
  }
  document.writeln("=== end showY =====<br>");
}

function addZ()
{
  z = "AlsoGlobalZ";
}

showY();
addZ();
document.writeln("=== start main =====<br>");
document.writeln("x = " + x + "<br>");
document.writeln("y = " + y + "<br>");
document.writeln("z = " + z + "<br>");
if (typeof(p) != "undefined") {
  document.writeln("p = " + p + "<br>");
} else {
  document.writeln("p is undefined<br>");
}
document.writeln("=== end main =====<br>");
showY();
```

```
=== start showY =====
x = GlobalX
y = ScopedY
p = ScopedP
z is undefined
=== end showY =====
=== start main =====
x = GlobalX
y = QuasiGlobalY
z = AlsoGlobalZ
p is undefined
=== end main =====
=== start showY =====
x = GlobalX
y = ScopedY
p = ScopedP
z = AlsoGlobalZ
=== end showY =====
```

```
function showIsFinite(input)
{
  if (isFinite(input)) {
    document.writeln("Yes, ["+ input +"] is finite<br>");
  } else {
    document.writeln("No, ["+ input +"] is not finite<br>");
  }
}

showIsFinite(null);
showIsFinite(10);
showIsFinite(NaN);
showIsFinite(Number.POSITIVE_INFINITY);
showIsFinite(Number.NEGATIVE_INFINITY);
showIsFinite(Number.BLAH);
```

isFinite

```
function showIsFinite(input)
{
  if (isFinite(input)) {
    document.writeln("Yes, ["+ input +"] is finite<br>");
  } else {
    document.writeln("No, ["+ input +"] is not finite<br>");
  }
}

showIsFinite(null);
showIsFinite(10);
showIsFinite(NaN);
showIsFinite(Number.POSITIVE_INFINITY);
showIsFinite(Number.NEGATIVE_INFINITY);
showIsFinite(Number.BLAH);
```

Yes, [null] is finite
Yes, [10] is finite
No, [NaN] is not finite
No, [Infinity] is not finite
No, [-Infinity] is not finite
No, [undefined] is not finite

```
function showIsNaN(input)
{
  if (isNaN(input)) {
    document.writeln("Yes, ["+ input +"] is NaN (not a number)<br>");
  } else {
    document.writeln("No, ["+ input +"] is not NaN (it's not not a number)<br>");
  }
}

showIsNaN(parseInt("abc"));
showIsNaN("abc");
showIsNaN(1/0);
showIsNaN(-1/0);
showIsNaN(10);
```

isNaN

```
function showIsNaN(input)
{
  if (isNaN(input)) {
    document.writeln("Yes, ["+ input +"] is NaN (not a number)<br>");
  } else {
    document.writeln("No, ["+ input +"] is not NaN (it's not not a number)<br>");
  }
}

showIsNaN(parseInt("abc"));
showIsNaN("abc");
showIsNaN(1/0);
showIsNaN(-1/0);
showIsNaN(10);
```

Yes, [NaN] is NaN (not a number)
Yes, [abc] is NaN (not a number)
No, [Infinity] is not NaN (it's not not a number)
No, [-Infinity] is not NaN (it's not not a number)
No, [10] is not NaN (it's not not a number)

```
function factorial(n) {
  document.writeln("factorial("+ n +")<br>");
  if (n < 0) {
    document.writeln("> invalid input ["+ n +"]<br>");
    return;
  } else if (n == 0) {
    document.writeln("> base case 0! = 1<br>");
    return 1;
  } else {
    document.writeln("> recursive case "+ n +"! = "+ n +" * ("+ n +"-1)!<br>");
    return n * factorial(n - 1);
  }
}

function showFactorial(n) {
  document.writeln("<br>CALCULATING " + n +"! ... <br>");
  var answer = factorial(n);
  document.writeln("ANSWER " + answer + "<br>");
}

showFactorial(-10);
showFactorial(0);
showFactorial(1);
showFactorial(2);
showFactorial(3);
showFactorial(4);
```

Élégant, mais...

Habituellement, il faut plus de temps et de mémoire

Récursion

```
function factorial(n) {
  document.writeln("factorial("+ n +")<br>");
  if (n < 0) {
    document.writeln("> invalid input ["+ n +"]<br>");
    return;
  } else if (n == 0) {
    document.writeln("> base case 0! = 1<br>");
    return 1;
  } else {
    document.writeln("> recursive case "+ n +"! = "+ n +" * ("+ n +"-1)!<br>");
    return n * factorial(n - 1);
  }
}

function showFactorial(n) {
  document.writeln("<br>CALCULATING " + n +"! ... <br>")
  var answer = factorial(n);
  document.writeln("ANSWER " + answer + "<br>");
}

showFactorial(-10);
showFactorial(0);
showFactorial(1);
showFactorial(2);
showFactorial(3);
showFactorial(4);
```

```
CALCULATING 4! ...
factorial(4)
> recursive case 4! = 4 * (4-1)!
factorial(3)
> recursive case 3! = 3 * (3-1)!
factorial(2)
> recursive case 2! = 2 * (2-1)!
factorial(1)
> recursive case 1! = 1 * (1-1)!
factorial(0)
> base case 0! = 1
ANSWER 24
```

```
var fibonacci = (function () {
    var memo = [0, 1];
    var fib = function (n) {
        var result = memo[n];
        if (typeof(result) !== 'number') {
            result = fib(n-1) + fib(n-2);
            memo[n] = result;
        }
        return result;
    };
    return fib;
}());

function showFibonacci(n) {
  var answer = fibonacci(n);
  document.writeln(
    `fibonacci(${n} = ${answer}<br>`);
}

showFibonacci(0);
showFibonacci(1);
showFibonacci(2);
showFibonacci(3);
showFibonacci(4);
showFibonacci(5);
showFibonacci(6);
```

Fib (x) est toujours le même (sans effets secondaires) afin que nous puissions mettre en cache les résultats.

Memoize

```
var fibonacci = (function () {
    var memo = [0, 1];
    var fib = function (n) {
        var result = memo[n];
        if (typeof(result) !== 'number') {
            result = fib(n-1) + fib(n-2);
            memo[n] = result;
        }
        return result;
    };
    return fib;
}());

function showFibonacci(n) {
  var answer = fibonacci(n);
  document.writeln(
    `fibonacci(${n} = ${answer}<br>`);
}

showFibonacci(0);
showFibonacci(1);
showFibonacci(2);
showFibonacci(3);
showFibonacci(4);
showFibonacci(5);
showFibonacci(6);
```

fibonacci(0) = 0
fibonacci(1) = 1
fibonacci(2) = 1
fibonacci(3) = 2
fibonacci(4) = 3
fibonacci(5) = 5
fibonacci(6) = 8

Memoize

```
var memoizer = function (formula) {
  var memo = [];
  var recur = function (n) {
    var result = memo[n];
    if (typeof result == 'undefined') {
      result = formula(recur, n);
      memo[n] = result;
    }
    return result;
  };
  return recur;
};

var factorial = memoizer(function (recur, n) {
  document.writeln("factorial("+ n +")<br>");
  if (n < 0) {
    document.writeln("> invalid input ["+ n +"]<br>");
    return;
  } else if (n == 0) {
    document.writeln("> base case 0! = 1<br>");
    return 1;
  } else {
    document.writeln("> recursive case "+ n +"! = "+ n +" * ("+ n +"-1)!<br>");
    return n * recur(n - 1);
  }
});

function showFactorial(n) {
  document.writeln("<br>CALCULATING " + n +"! ... <br>");
  var answer = factorial(n);
  document.writeln("ANSWER " + answer + "<br>");
}

showFactorial(-10);
showFactorial(0);
showFactorial(1);
showFactorial(4);
showFactorial(2);
showFactorial(3);
```

Mais nous obtenons les valeurs en cache "gratuitement"

Même implémentation

Memoizer

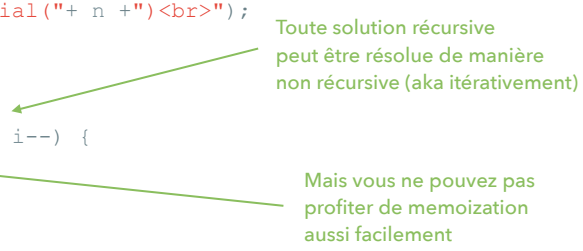
```
var memoizer = function (formula) {
  var memo = [];
  var recur = function (n) {
    var result = memo[n];
    if (typeof result == 'undefined') {
      result = formula(recur, n);
      memo[n] = result;
    }
    return result;
  };
  return recur;
};

var factorial = memoizer(function (recur, n) {
  document.writeln("factorial("+ n +")<br>");
  if (n < 0) {
    document.writeln("> invalid input ["+ n +"]<br>");
    return;
  } else if (n == 0) {
    document.writeln("> base case 0! = 1<br>");
    return 1;
  } else {
    document.writeln("> recursive case "+ n +"! = "+ n +" * ("+ n +"-1)!<br>");
    return n * recur(n - 1);
  }
});

function showFactorial(n) {
  document.writeln("<br>CALCULATING " + n +"! ... <br>");
  var answer = factorial(n);
  document.writeln("ANSWER " + answer + "<br>");
}

showFactorial(-10);
showFactorial(0);
showFactorial(1);
showFactorial(4);
showFactorial(2);
showFactorial(3);
```

CALCULATING 4! ...
factorial(4)
> recursive case 4! = 4 * (4-1)!
factorial(3)
> recursive case 3! = 3 * (3-1)!
factorial(2)
> recursive case 2! = 2 * (2-1)!
ANSWER 24

CALCULATING 2! ...
ANSWER 2

CALCULATING 3! ...
ANSWER 6

```
function factorial(n) {
  document.writeln("factorial("+ n +")<br>");
  if (n < 0) {
    return;
  } else {
    var runningTotal = 1;
    for (var i = n; i > 0; i--) {
      runningTotal *= i;
    }
    return runningTotal;
  }
}

function showFactorial(n) {
  document.writeln("<br>CALCULATING " + n +"! ... <br>");
  var answer = factorial(n);
  document.writeln("ANSWER " + answer + "<br>");
}

showFactorial(-10);
showFactorial(0);
showFactorial(1);
showFactorial(2);
showFactorial(3);
showFactorial(4);
```



Itêration

```
function showArray(name, arr)
{
  var output = "Array "+ name +" ["+ arr.length+"] is [";
  for (var i=0, len = arr.length; i < len; i++) {
    output += arr[i] + ", ";
  }
  output += "]<br>";
  document.writeln(output);
}

var a = new Array(1, 2, 3);
var b = [1, 2, 3];
var c = [];
var d = new Array(3);
var e = [,,,];
c.push(1);
c.push(2);
c.push(3);

showArray("a", a);
showArray("b", b);
showArray("c", c);
showArray("d", d);
showArray("e", e);
```

Les tableaux sont de type “Array”

Initialiser avec les valeurs 1, 2, 3

Syntaxe raccourcie, utilisez ceci s'il vous plaît

Un tableau vide

Attention, un tableau de taille 3, PAS un tableau de taille un avec un “3" dedans

Ajouter des éléments à un tableau existant

Tableau “Array”

```
function showArray(name, arr)
{
  var output = "Array "+ name +" ["+ arr.length+"] is [";
  for (var i=0, len = arr.length; i < len; i++) {
    output += arr[i] + ", ";
  }
  output += "]<br>";
  document.writeln(output);
}

var a = new Array(1, 2, 3);
var b = [1, 2, 3];
var c = [];
var d = new Array(3);
var e = [,,,];
c.push(1);
c.push(2);
c.push(3);

showArray("a", a);
showArray("b", b);
showArray("c", c);
showArray("d", d);
showArray("e", e);
```

Array a [3] is [1, 2, 3,]

Array b [3] is [1, 2, 3,]

Array c [3] is [1, 2, 3,]

Array d [3] is [undefined, undefined, undefined,]

Array e [3] is [undefined, undefined, undefined,]

Pré-calculer la longueur pour éviter l'appel à chaque fois

```
var arr = ["apples", "bananas", "carrots"];

document.writeln("<h3>Method 1</h3>");
for (var i=0, len = arr.length; i < len; i++) {
  document.writeln("["+ i +"] = " + arr[i] + "<br>");
}
```

Le deuxième argument (index de l'élément) est optionnel

```
document.writeln("<h3>Method 2</h3>");
for (var i in arr) {
  document.writeln("["+ i +"] = " + arr[i] + "<br>");
}
```

Evitez for/in, il est beaucoup plus lent qu'un simple pour la boucle

```
document.writeln("<h3>Method 3 (slow)</h3>");
arr.forEach(function (ele, i /* optional */){
  document.writeln("["+ i +"] = " + ele + "<br>");
})
```

Boucle For

```
var arr = ["apples", "bananas", "carrots"];

document.writeln("<h3>Method 1</h3>");
for (var i=0, len = arr.length; i < len; i++) {
  document.writeln("["+ i +"] = " + arr[i] + "<br>");
}

document.writeln("<h3>Method 2</h3>");
for (var i in arr) {
  document.writeln("["+ i +"] = " + arr[i] + "<br>");
}

document.writeln("<h3>Method 3 (slow)</h3>");
arr.forEach(function (ele, i /* optional */){
  document.writeln("["+ i +"] = " + ele + "<br>");
})
```

Method 1

[0] = apples
[1] = bananas
[2] = carrots

Method 2

[0] = apples
[1] = bananas
[2] = carrots

Method 3 (slow)

[0] = apples
[1] = bananas
[2] = carrots

```
function show(name, o)
{
  document.writeln(name + " = " + o + "<br>");
}

function changeInt(i)
{
  i += 1;
}

function changeString(str)
{
  str += " a change";
}

function changeBoolean(b)
{
  b = !b;
}

var x;

x = 99;
show("x before", x);
changeInt(x);
show("x after", x);

x = "abc";
show("x before", x);
changeString(x);
show("x after", x);

x = true;
show("x before", x);
changeBoolean(x);
show("x after", x);
```

Semblable à java;
Entiers, les chaînes et
les booléens sont passés
par valeur

```
function show(name, o)
{
  document.writeln(name + " = " + o + "<br>");
}

function changeInt(i)
{
  i += 1;
}

function changeString(str)
{
  str += " a change";
}

function changeBoolean(b)
{
  b = !b;
}

var x;

x = 99;
show("x before", x);
changeInt(x);
show("x after", x);

x = "abc";
show("x before", x);
changeString(x);
show("x after", x);

x = true;
show("x before", x);
changeBoolean(x);
show("x after", x);
```

x before = 99

x after = 99

x before = abc

x after = abc

x before = true

x after = true

```
function show(name, o)
{
  document.writeln(name + " = " + o + "<br>");
}

function showObj(name, o)
{
  document.writeln(name + " = " + o.name + ", "+ o.colour +"<br>");
}

function changeArray(arr)
{
  arr[0] = "candy";
  arr.push("oranges");
}

function changeObject(o)
{
  o.name = "James";
  o.colour = "green";
}

var x;

x = ["apples"];
show("x before", x);
changeArray(x);
show("x after", x);

x = {"name": "Andrew"};
showObj("x before", x);
changeObject(x);
showObj("x after", x);
```

Semblable à java; Les objets (qui inclue les tableau) sont passés par reference

```
function show(name, o)
{
  document.writeln(name + " = " + o + "<br>");
}

function showObj(name, o)
{
  document.writeln(name + " = " + o.name + ", "+ o.colour +"<br>");
}

function changeArray(arr)
{
  arr[0] = "candy";
  arr.push("oranges");
}

function changeObject(o)
{
  o.name = "James";
  o.colour = "green";
}

var x;

x = ["apples"];
show("x before", x);
changeArray(x);
show("x after", x);

x = {"name": "Andrew"};
showObj("x before", x);
changeObject(x);
showObj("x after", x);
```

x before = apples

x after = candy, oranges

x before = Andrew, undefined

x after = James, green

```
var arr = ["apples", "oranges", "carrots"];

document.writeln(arr.join() + "<br>");

document.writeln(arr.join("&#9786;") + "<br>");
```

Le séparateur par défaut est une virgule (sans espaces)

Peut être modifié

Array.join

```
var arr = ["apples", "oranges", "carrots"];

document.writeln(arr.join() + "<br>");

document.writeln(arr.join("&#9786;") + "<br>");
```

apples,oranges,carrots

apples ☺ oranges ☺ carrots

```
var arr = ["monkey", "apple", "zealot"];
document.writeln(arr.sort().join(", ") + "<br>");

var arr = [40, 8, 320];
document.writeln(arr.sort().join(", ") + "<br>");
```

Pas ce que
vous attendez

Array.sort

```
var arr = ["monkey", "apple", "zealot"];
document.writeln(arr.sort().join(", ") + "<br>");

var arr = [40, 8, 320];
document.writeln(arr.sort().join(", ") + "<br>");
```

apple, monkey, zealot

320, 40, 8

Algorithme de tri basé
sur des chaînes ("4" < "8", donc "40" <"8")

```
var arr = [40, 8, 40, 320];

arr.sort(function (a, b){
  return a - b;
})

document.writeln(arr.join(", ") + "<br>");
```

Vous pouvez passer (comme fonction) un comparateur

Il renvoie trois “types” de valeurs

Si “a” est avant “b”, renvoie un nombre négatif

Si “a” est après “b”, renvoie un nombre positif

Si elles sont identiques, retournez 0

Tri personnalisé

```
var arr = [40, 8, 40, 320];

arr.sort(function (a, b){
  return a - b;
})

document.writeln(arr.join(", ") + "<br>");
```

8, 40, 40, 320

```
function show(name, ele, index)
{
  document.writeln(name + " of " + ele + " is " + index + "<br>");
}

var arr = [40, 8, 40, 320, 40, 57];

show("indexOf(40)", 40, arr.indexOf(40));
show("lastIndexOf(40)", 40, arr.lastIndexOf(40));
show("indexOf(8)", 8, arr.indexOf(8));
show("lastIndexOf(8)", 8, arr.lastIndexOf(8));
show("indexOf(40, 1)", 40, arr.indexOf(40, 1));
show("lastIndexOf(40, 1)", 40, arr.lastIndexOf(40, 1));
show("lastIndexOf(40, 3)", 40, arr.lastIndexOf(40, 3));
show("indexOf(40, 5)", 40, arr.indexOf(40, 5));
show("indexOf(40,99)", 40, arr.indexOf(40,99));
```

L'index de "40" dans le tableau

Le dernier index de "40" dans le tableau

Offset de départ optionnelle

Renvoie -1 si le offset est hors limite ou si la valeur n'est pas trouvée

Pensez à la «lastIndexOf» comme la «indexOf», mais dans la direction opposée

indexOf(40) of 40 is 0
lastIndexOf(40) of 40 is 4
indexOf(8) of 8 is 1
lastIndexOf(8) of 8 is 1
indexOf(40,1) of 40 is 2
lastIndexOf(40,1) of 40 is 0
lastIndexOf(40,3) of 40 is 2
indexOf(40,99) of 40 is -1

indexOf / lastIndexOf

```
var arr = [
  ["Andrew", 44],
  ["Hayden", 12],
  ["Ayana", 44],
  ["August", 9]
];

arr.sort(function(a,b){
  if (a[1] == b[1]) {
    return a[0].localeCompare(b[0]);
  } else {
    return b[1] - a[1];
  }
})

document.writeln(arr.join("<br>"));
```

Les tableaux de n'importe quel nombre de dimensions

Mais utilisez plutôt une structure de données plus appropriée

Ensuite, par nom

Du plus ancien au plus jeune

Andrew,44
Ayana,44
Hayden,12
August,9

Tableau multidimensional

Un objet global nommé Math

```
Math.abs(-1);
Math.floor(38.9);
Math.ceil(12.2);
Math.round(14.49);
Math.max(-2, 4);
Math.min(-2, 4);
Math.PI;
```

Plusieurs fonctions sont disponibles, comme un nombre d'arrondis à l'entier le plus proche

```
Math.abs(-1);
Math.floor(38.9);
Math.ceil(12.2);
Math.round(14.49);
Math.max(-2, 4);
Math.min(-2, 4);
Math.PI;
```

Math.abs(-1) = 1
Math.floor(38.9) = 38
Math.ceil(12.2) = 13
Math.round(14.49) = 14
Math.max(-2, 4) = 4
Math.min(-2, 4) = -2
Math.PI = 3.141592653589793

Une chaîne est également un objet avec des méthodes

```
"this course rocks".charAt(2);
"this course rocks".charCodeAt(2);
"this course".concat(" is just ok").replace("just ok", "rocks");
"this course is just ok".replace("just ok", "awesome actually");
"this course is just ok".split(" ");
"STOP YELLING".toLowerCase();
"I said pardon?".toUpperCase();
"Off by one errors".substring(0,9);
```

Lisez attentivement les documents de l'API pour connaître la signification des arguments (par exemple l'index ou à la longueur)

```
"this course rocks".charAt(2);
"this course rocks".charCodeAt(2);
"this course".concat(" is just ok").replace("just ok", "rocks");
"this course is just ok".replace("just ok", "awesome actually");
"this course is just ok".split(" ");
"STOP YELLING".toLowerCase();
"I said pardon?".toUpperCase();
"Off by one errors".substring(0,9);
```

"this course rocks".charAt(2) = i

"this course rocks".charCodeAt(2) = 105

"this course".concat(" is just ok").replace("just ok", "rocks") = this course is rocks

"this course is just ok".replace("just ok", "awesome actually") = this course is awesome actually

"this course is just ok".split(" ") = this,course,is,just,ok

"STOP YELLING".toLowerCase() = stop yelling

"I said pardon?".toUpperCase() = I SAID PARDON?

"Off by one errors".substring(0,9) = Off by on

String / Chaine

Comme pour les méthodes Array

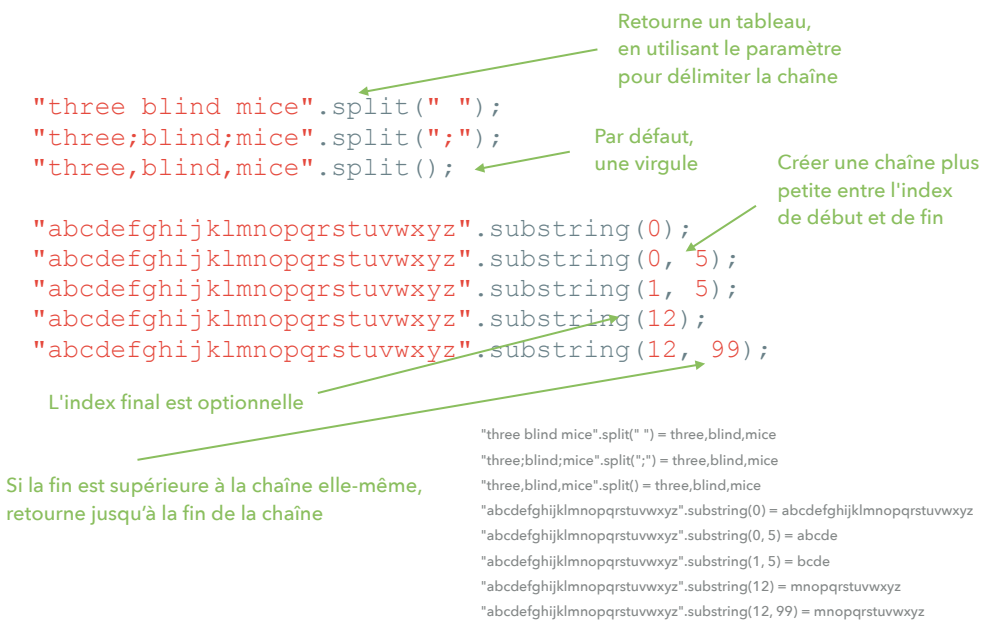
```
"three blind mice".indexOf("blind");
"three seeing mice".indexOf("blind");
"there are these things in the woods".lastIndexOf("the");
"there are these things in the woods".indexOf("the");
"there are these things in the woods".indexOf("the", 1);
"there are these things in the woods".indexOf("the", 11);
"there are these things in the woods".indexOf("the", 27);
"there are these things in the woods".lastIndexOf("the", 9);
"there are these things in the woods".lastIndexOf("the", 25);
```

Index de départ optionnelle

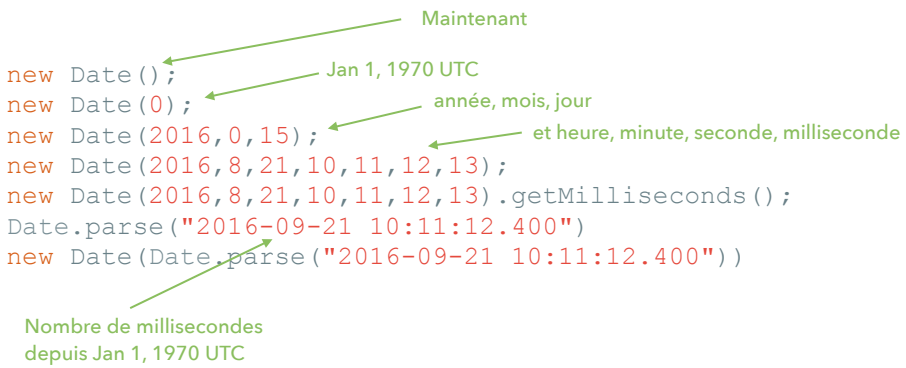
Travailler en arrière avec lastIndexOf

"three blind mice".indexOf("blind") = 6
"three seeing mice".indexOf("blind") = -1
"there are these things in the woods".lastIndexOf("the") = 26
"there are these things in the woods".indexOf("the") = 0
"there are these things in the woods".indexOf("the", 1) = 10
"there are these things in the woods".indexOf("the", 11) = 26
"there are these things in the woods".indexOf("the", 27) = -1
"there are these things in the woods".lastIndexOf("the", 9) = 0
"there are these things in the woods".lastIndexOf("the", 25) = 10

indexOf / lastIndexOf



split / substring




```
new Date();
new Date(0);
new Date(2016,0,15);
new Date(2016,8,21,10,11,12,13);
new Date(2016,8,21,10,11,12,13).getMilliseconds();
Date.parse("2016-09-21 10:11:12.400")
new Date(Date.parse("2016-09-21 10:11:12.400"))
```

```
new Date() = Sun Jan 15 2017 09:58:09 GMT-0500 (EST)
new Date(0) = Wed Dec 31 1969 19:00:00 GMT-0500 (EST)
new Date(2016,0,15) = Fri Jan 15 2016 00:00:00 GMT-0500 (EST)
new Date(2016,8,21,10,11,12,13) = Wed Sep 21 2016 10:11:12 GMT-0400 (EDT)
new Date(2016,8,21,10,11,12,13).getMilliseconds() = 13
Date.parse("2016-09-21 10:11:12.400") = 1474467072400
new Date(Date.parse("2016-09-21 10:11:12.400")) = Wed Sep 21 2016 10:11:12 GMT-0400 (EDT)
```

Date

```
new Date();
(new Date()).getFullYear();
(new Date()).getUTCFullYear();
(new Date()).getMonth();
(new Date()).getDate();
(new Date()).getDay();
(new Date()).getHours();
(new Date()).getUTCHours();
(new Date()).getMinutes();
(new Date()).getSeconds();
(new Date()).getMilliseconds();
(new Date()).getTime();
(new Date()).getTimezoneOffset();
new Date((new Date()).setMinutes(59));
```

Chaque getX a un getUTCX

Mois de l'année (0-12)

Jour du mois (1-31)

Jour de la semaine (0 - 6)
Dimanche à Samedi

En minutes

Chaque getX a un setX

```
new Date() = Sun Jan 15 2017 10:10:52 GMT-0500 (EST)
(new Date()).getFullYear() = 2017
(new Date()).getUTCFullYear() = 2017
(new Date()).getMonth() = 0
(new Date()).getDate() = 15
(new Date()).getDay() = 0
(new Date()).getHours() = 10
(new Date()).getUTCHours() = 15
(new Date()).getMinutes() = 10
(new Date()).getSeconds() = 52
(new Date()).getMilliseconds() = 962
(new Date()).getTime() = 1484493052962
(new Date()).getTimezoneOffset() = 300
new Date((new Date()).setMinutes(59)) = Sun Jan 15 2017 10:59:52 GMT-0500 (EST)
```

Méthodes de Date

```
new Boolean(true);
new Boolean("James");
new Boolean("0");

new Boolean(false);
new Boolean(0);
new Boolean("");
new Boolean();
new Boolean(null);
new Boolean(NaN);
new Boolean(undefined);
```

"0" est vrai

0 est faux

new Boolean(true) = true
new Boolean("James") = true
new Boolean("0") = true
new Boolean(false) = false
new Boolean(0) = false
new Boolean("") = false
new Boolean() = false
new Boolean(null) = false
new Boolean(NaN) = false
new Boolean(undefined) = false

```
new Number(10)
Number.MAX_VALUE
Number.MIN_VALUE
Number.NaN
Number.NEGATIVE_INFINITY
Number.POSITIVE_INFINITY
```

Ne faites pas cela

Certaines constantes disponibles, de la plus grande valeur est NaN ("Not a Number")

new Number(10) = 10
Number.MAX_VALUE = 1.7976931348623157e+308
Number.MIN_VALUE = 5e-324
Number.NaN = NaN
Number.NEGATIVE_INFINITY = -Infinity
Number.POSITIVE_INFINITY = Infinity

Number

RÉFÉRENCES

- http://www.adequatelygood.com/JavaScript-Module-Pattern-In-Depth.html
- http://codepen.io/Oguima/pen/eJpaOW
- http://media.io/
- https://www.safaribooksonline.com/library/view/javascript-the-good/9780596517748/ch04s15.html
- https://coderwall.com/p/kvzbpa/don-t-use-array-foreach-use-for-instead
- http://www.quirksmode.org/js/cookies.html
- http://stackoverflow.com/questions/4201441/is-there-any-practical-reason-to-use-quoted-strings-for-json-keys
- http://stackoverflow.com/questions/19839952/all-falsey-values-in-javascript