Javascript, Fondamentaux

Javascript, Fondamentaux

Cdiscount

Découvrez l'essentiel du langage JavaScript pour développer vos pages web dynamiques.

15/11/2017 - 17/11/2017

@thomasvergnaud

Tour De Table ?

Quis Suis-Je ?

Developer Front-End

DESIGNER 4 ANS
FRONT END 11 ANS
FULLSIX, VIADEO, AUFEMININ, CLEVER-AGE

Tooling Ide

CHROME + DEVELOPPER TOOLBAR
FIREFOX + FIREBUG
SUBLIME TEXT
TERMINAL

Sublime Text

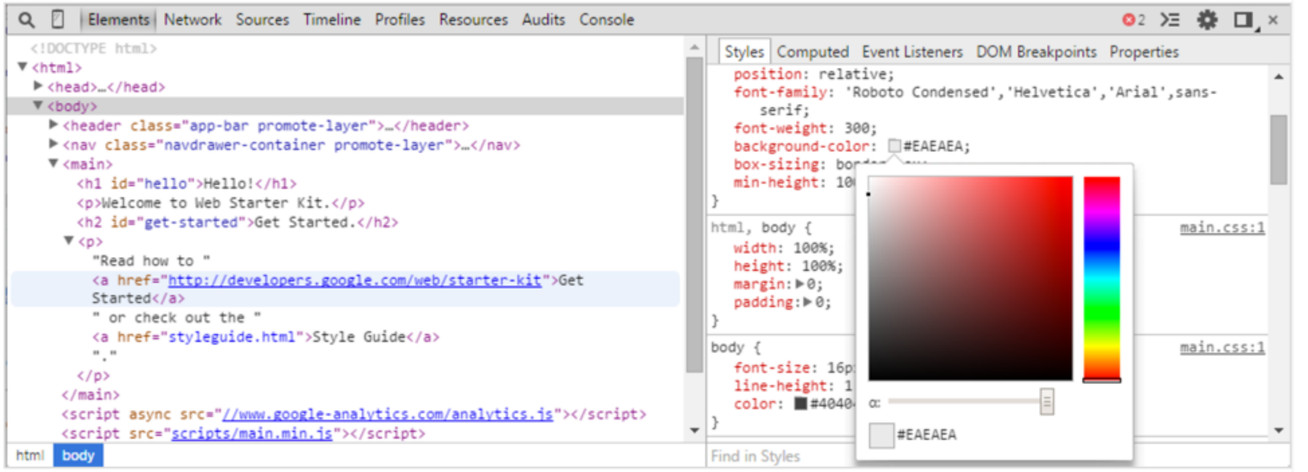
Sublime Text is a sophisticated text editor for code, markup and p
ove the slick user interface, extraordinary features and amazing p

Demonstration

```
_encode(const uint8_t * data, size_t length, char * dst)

src_idx = 0;
dst_idx = 0;
(src_idx + 2) < length; src_idx += 3, dst_idx += 4)

t8_t s0 = data[src_idx];
t8_t s1 = data[src_idx + 1];
t8_t s2 = data[src_idx + 2];

[dst_idx + 0] = charset[(s0 & 0xfc) >> 2];
[dst_idx + 1] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
[dst_idx + 2] = charset[((s1 & 0x0f) << 2) | (s2 & 0xc0) >> 6];
[dst_idx + 3] = charset[(s2 & 0x3f)];


_idx < length)

t8_t s0 = data[src_idx];
t8_t s1 = (src_idx + 1 < length) ? data[src_idx + 1] : 0;

[dst_idx++] = charset[(s0 & 0xfc) >> 2];
[dst_idx++] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
(src_idx + 1 < length)
 dst[dst_idx++] = charset[((s1 & 0x0f) << 2)];
```

The DevTools window

The DevTools are organised into task-oriented groups in the toolbar at the top of the window. Each toolbar item and corresponding panel let you work with a specific type of page or app information, including DOM elements, resources, and sources.



The colorpicker available in the DevTools..

Overall, there are eight main groups of tools available view Developer Tools:

- **Elements**
- **Resources**
- **Network**
- Sources
- **Timeline**
- **Profiles**
- Audits
- **Console**

Sublime Config

Package Most Used

- git
- sublimeLinter
- JavaScript Snippets
- Color Highlighter
- Bracket Highlighter
- CSS Comments
- emmet
- sublime-JSHint

Programme

Introduction Au Développement Web Dynamique

Le Langage Javascript

Le Dom (Document Object Models)

Evènements interactifs

Ajax : Dialoguer Avec Le Serveur

Outils Et Bonnes Pratiques De Développement

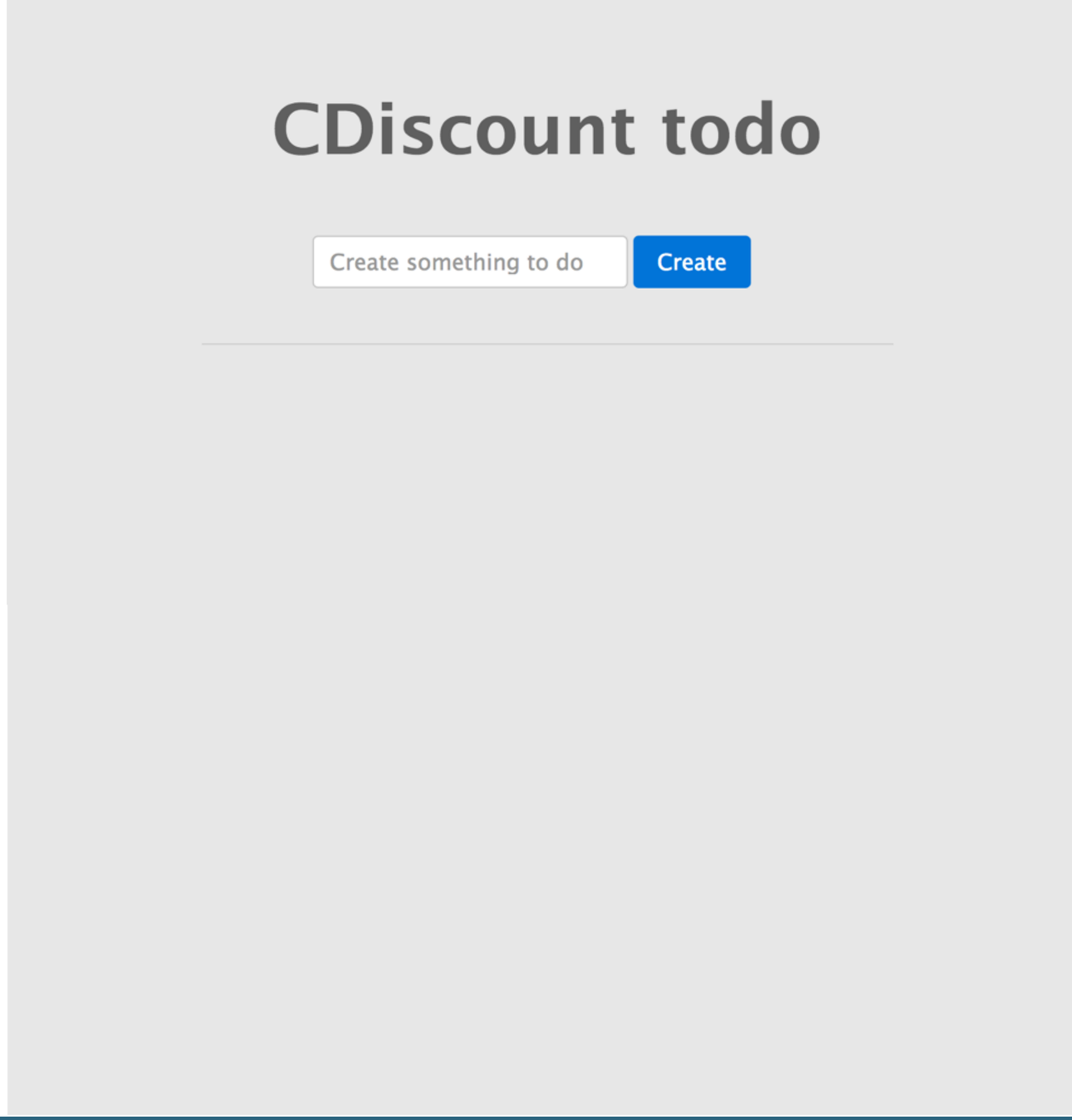
Les Données Et Les Boucles

Les Fonctions

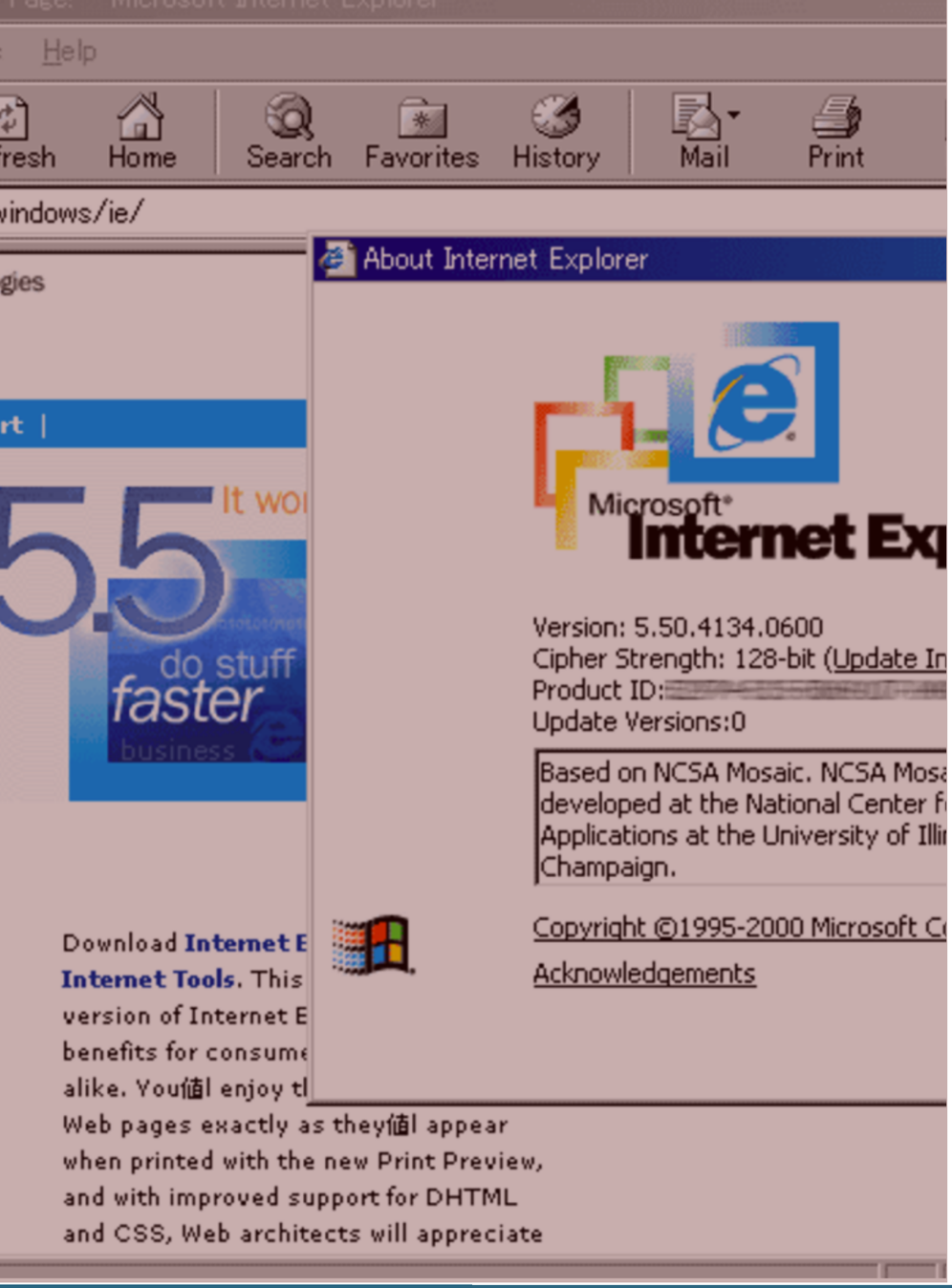
Nodejs / More

Coding Project

- Réaliser une Todo app page en 5 coding time



Marché



Avant

- Usage domestique ou professionnel faible
- Résolution basse pour tout le monde
- Peu de navigateurs : IE5.5 IE6 Nescape 4

Après

- Mobile first : un ordinateur en poche
- Beaucoup de navigateurs
  - IE 7,8,9,10,11
  - Firefox, Safari, Chrome, Opera
- 4G et fibre
- Domotique ?

Rôle & Standards

Definition

JavaScript®, est le language de script développé par Netscape utilisé dans des millions de pages web et d'applications serveur dans le monde entier.
JavaScript est un language léger, interprété, orienté objet et événementiel.

https://developer.mozilla.org/fr/docs/Web/JavaScript/A_propos

Definition

- Officiellement javascript s’appelle **ECMASCRIPT**
- Inventé en 1995 par Brendan Eich
- language client side & not compile code
- javascript == parsed code.

Language

- Mots clés, instructions en séquences.
- Les types de variables (booléen, nombre, chaîne de caractères, fonction, objet…)
- Case sensitive, utilisation partielle du ;
- L'objet global

De Nos Jours

- Language le plus populaire du monde source Github
- Version 1.8.5 actuelle
- Supporté par tous les navigateurs récents.
- **313,403 repository results**

Languages	
JavaScript	183,739
HTML	42,057
CSS	12,983
PHP	5,612
TypeScript	3,550
Java	3,236
Ruby	2,597
Python	2,297
CoffeeScript	1,946
C#	1,284

Utilisation

- Utilisation web interne aux navigateurs (moteur d’éxécution JS)
- Utilisation Back et Front (Nodejs)
- Utilisation objets connectés javascript ex : Tessel

Idées Reçues

- **JAVA** !== **JAVASCRIPT**
- **JAVA** backend language / **JAVASCRIPT** Front End language
- **JAVASCRIPT** runs on browser (client side)
- **JAVASCRIPT** == scripting language

Framework Js

- Librairies JS année 2000 -> jQuery, motools, Prototype.
- 2009 -> Sortie de Nodejs
- Framework MVC -> Angularjs, Backbone, React, Ember
- Today -> VueJS, Angular 2, React

Problème

- Pollution du scope global
- Utilisation du ;
- Debug pauvre -> console ???

Javascript & Html

Synchrone Vs Asynchrone

- Synchrone: On attend qu'une action A soit finie avant de démarrer l'action B.
- Asynchrone: Deux actions A et B peuvent être démarrées ensemble, on ne sait pas laquelle se terminera en premier.

Balise Script In Html

- script tag
- internal / external
- place dans le dom
- attribute defer & async

Defer : browser will run your script when the page finished parsing. use for GA

Async : browser will continue to load the HTML page and render it while the browser load and execute the script at the same time.

Javascript
Le Langage

Type

- les booléens (Boolean)
- les chaînes de caractères (String)
- les nombres (Number)
- Undefined
- Null
- *les objets (Object)*

```
1 var n = 42;
2 typeof n; // "number"
3
4 var b = true;
5 typeof b; // "boolean"
6
7 var s = "42";
8 typeof b; // "string"
9
10 var u;
11 typeof u // "Undefined"
12
13 var o = {hello : "world"};
14 typeof o // "Object"
```

Comments

- Single line
- Multiline

```
1 // Single comment line
2
3 /*
4 Multiple lines
5 comments
6 /*
```

Var Keyword

- declaration
- coercion
- concatenation

```
/**
  Var keyword : declaration multiple,
  declaration simple, coercion, concatenation
*/
var number = 12,
   string = '1',
   secondString = "hello",
   thirdString = 'world'
   bool = true,
   other = null,
   array = [],
   object = {}
;

var otherVariable = 12;
var helloWorld = 'hello world';

// Coercion type
number + "";
// "12"
+string
// 1

// Concatenation
var str = secondString + ' ' + thirdString + ' ?';
```

Var, Let, Const

- MAJ ECMAScript 6

```
/*
var Scoping Refresher
*/

// var variable can be updated and redefined
// var are function scoped

var width = 100;

// let and const and block scoped
// (if statement or function statement ex)

let height = 200;
const value = 'value';
```

Let, Const

- MAJ ECMAScript 6

```
/*
Let vs Const
*/
// let and const are only declare once and scoped to block
// const can not be updated
// Let can be updated

// I can't update key
const key = '123094094';
let winner  = false;
let points = 48;

// let winner as same name but
// block scoped winner = true, scoped by block
if(points <= 50) {
  let winner = true;
  console.log('Log winner let 02 ' + winner);
}
```

Variables & Fonctions

- declaration
- visibility

```
// Declare variable
var maVariable;

// Declare Function
function maFonction() {};

// Call Function
maFonction();
```

```
var a = 'kikoo';

function fn() {
  var b = 'lol';

  console.log(a); // kikoo
  console.log(b); // lol
};

fn();
console.log(a);
// kikoo
console.log(b);
// ReferenceError: b is undefined
```

Manipulation

Forcer un type: Number

```
1 var foo = "10.5"; // String
2 foo.parseInt(foo);    // 10
3 foo.parseFloat(foo);  // 10.5
```

Manipuler un String

```
1 var s = "Jack l'opossum";
2 s.indexOf('opossum');        // 7
4 s.split(' ');// ["Jack", "l'opossum"]
6 s.substr(1, 3);     // "ack"
7 s.toLowerCase(); // "jack l'opossum"
8 s.toUpperCase(); // "JACK L'OPOSSUM"
```

Manipulation

Arrays

```
var a = ["apple","orange","banana"];
a.push("kiwi");
//        3 - ["apple","orange","banana","kiwi"]
a.shift();
//  "apple" - ["orange","banana", "kiwi"]
a.unshift("lemon");
//        3 - ["lemon","orange","banana",kiwi"]
a.reverse();
// ["kiwi","banana","orange","lemon"]
a.sort();
// ["banana","kiwi","lemon","orange"]
```

Sous Type Object

- les tableaux (Array)

```
16 //Declarations ARRAY
17 var array = [];
18 var array = new Array;
19
20 //Array always begin with 0
21 var a = [ ];
22 a.length;   // 0
23 a[0] = 1;
24 a[1] = "2";
25 a[2] = [ 3 ];
26 a.length;   // 3
27 a // [1, "2", [3]]
28
29 // Declaration Date
30 var today  = new Date;
31 console.log(today); //Sun SEP 23 2016 16:41:37
32
```

Condition

- if
- else
- else if
- switch case

```
// Check if condition is true
if ( true ) {
  //Execute some code
};

// Else statement
if ( true ) {
  //Execute some code
} else {
  //Otherwise execute this code
}
// Else if statement
if ( true ) {
  //Execute some code
} else if ( true ) {
  //Otherwise execute this code
} else {
  // Else execute this code
}
// Match condition value
switch(condition) {
  case value1:
    // do something
    break;
  case value2:
    // do something
    break;
  default:
    // default action
}
// Ternair operator
// condition ? Execute some code : Otherwise execute this code

```

Operators

- Arithmethics
- Increments
- Comparaison

```
1  // Arithmethics operations
2  var add = 2+10;
3  var sub = 5-10.8;
4  var mult = 10*5;
5  var div = 2456/3;
6
7  // Increment operations
8  var i = 1;
9  i = i + 1;
10 i++;
11 ++i;
12
13 // Comparison operators
15 ==
16 // True, if the two operands are equal, ignore type
18 ===
19 // True if same value and type (more safe !!!)
21 !=
22 // True, if the two operands are not equal, ignore type
24 !==
25 //True if not same value and type
27 // Lower and greather op
28 >, >=, <, <=
```

Operators

- Logical

```
1 var sentence  = 'hello world';
2
3 // Logical operators
4 &&
5 // And
6
7 ||
8 // Or
9
10 !
11 // Not
12
13 // Exemple test if sentence is string type and not empty
14 if(typeof sentence === 'string' && sentence !== '') {
15   //Excecute some code
16   console.log("String is not empty, and append to String type")
17 }
18
19 // Exemple test if sentence is present
20 if (typeof sentence !== undefined) {
21   console.log("Sentence exist")
22 }
```

Loop

- for
- for in

```
1  // For Loop
2  var arr = ['item1', 'item2', 'item3']
3
4  for (var i=0; i < arr.length; i++) {
5    // I must be lower then a.length for iterate
6    // I is the iterator, i++
7    //execute code
8    console.log('log each item in arr ' + arr[i])
9  }
10
14
1  // ForIn Loop for obj
16 var obj = {key : 'value1', key2 : 'value2'};
17
18 for ( var item in obj) {
19   console.log('log key and value ' + obj[item], item);
20 }
```

Object Javascript Globaux

- Math
- Date
- http://momentjs.com/

```
1  // MATH object
2  Math.PI; // 3.14159...
3  Math.round(10.4); // 10
4  Math.round(10.5); // 11
5  Math.ceil(10.4);  // 11
6  Math.floor(10.5); // 10
7  Math.max(1, 2, 3); // 3
8  Math.min(1, 2, 3); // 1
9  Math.random(); // 0 a 1 …
10
11 // DATE object
12 new Date().getTime(); // Obtenir un timestamp
13 +new Date;            // Le même en version
14 compact
15 Date.now();           // Le même pour les
16          navigateurs modernes
17 var d = new Date('February 23, 2014');
18 d.getDate();  // retourne le jour du mois, 23 !
19 d.getMonth(); // retourne le mois de l'annâˆšée,
20            1 = Février !!!
21 d.getYear();  // retourne 114 !!!!!!! WTF!
```

Coding Time 1

Object

Un objet est un ensemble de propriétés et une propriété est une association entre un nom (aussi appelé clé) et une valeur (value).
La valeur d'une propriété peut être une fonction, auquel cas la propriété peut être appelée « méthode ».

https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Utiliser_les_objets

Object

Simple Key value Object

```
var car  = {
  seat : 4,
  citycar : true,
  color: "red",
  size :  {
    width : 200,
    height : 400
  }
};

// Create Object
var subcar = new Object(); // same as var
                subcar = {};

subcar["seat"] = 4;
subcar["citycar"] = true;
subcar["color"] = red;
subcar.seat // 4
```

Json

JavaScript Object Notation, est un format de données textuelles dérivé de la notation des objets du language JavaScript.

https://fr.wikipedia.org/wiki/JavaScript_Object_Notation

Object

JSON example

http://jsonlint.com/

http://www.json-generator.com/

http://codebeautify.org/jsonviewer

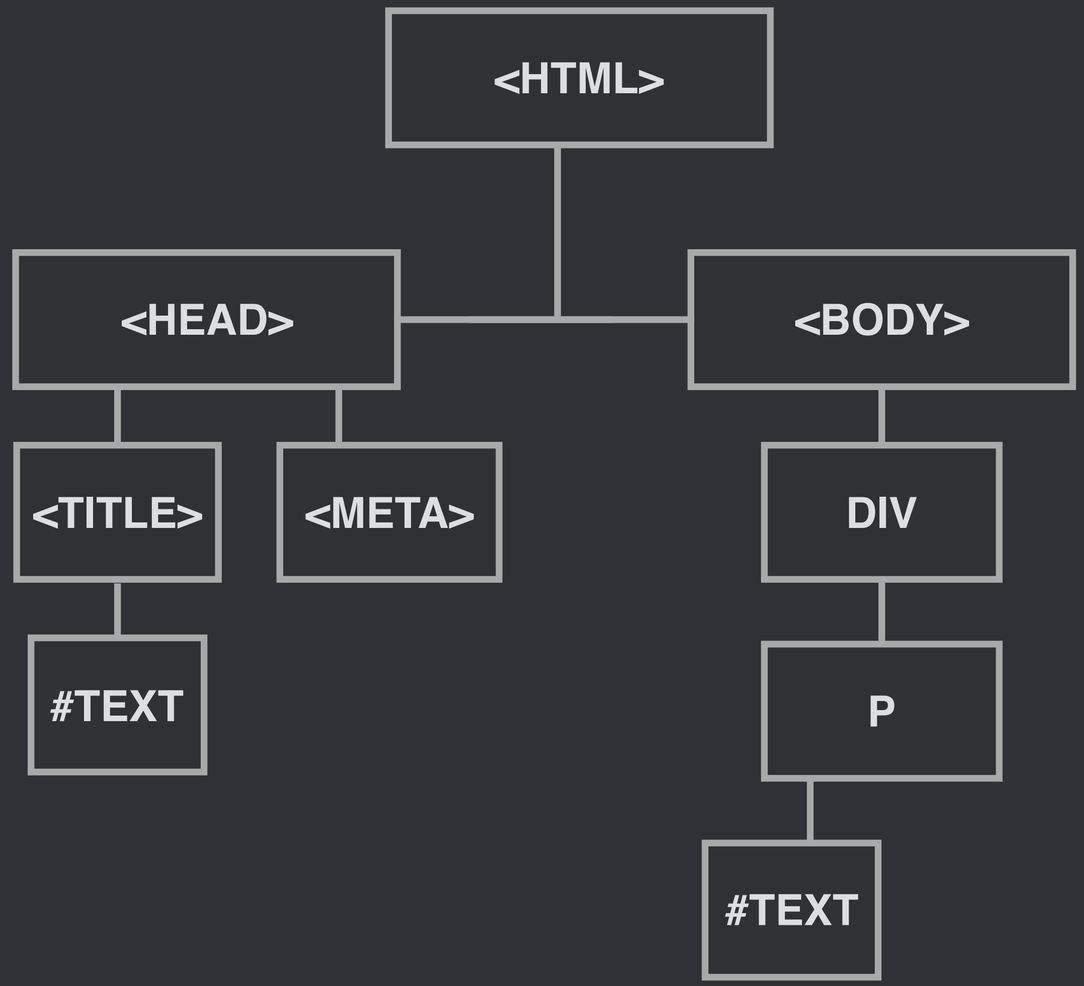
```
[
  {
    "_id": "583310e0c11dabdd0c78c795",
    "index": 0,
    "guid": "e8c77ec6-9b1a-4e2c-a537-
    046e3c5c57ff",
    "isActive": false,
    "balance": "$1,887.83",
    "picture": "http://placehold.it/32x32",
    "age": 22,
    "gender": "female",
    "company": "MALATHION",
    "email": "rhondadickson@malathion.com",
    "phone": "+1 (814) 461-2604",
    "address": "437 Gates Avenue, Crisman,
    Maryland, 719",
    "about": "Voluptate proident enim ea do.
    Voluptate laborum reprehenderit nulla ut"
    "registered": "2014-03-24T07:56:27 -01:00",
    "latitude": -38.696183,
    "longitude": 76.749437,
    "tags": [
      "ea",
      "adipisicing",
      "proident",
      "sint",
      "duis"
    ]
  }
]
```

Coding Time Object

Dom (Documents Object Models) & Manipulation

Dom

Le DOM permet aux programmes et scripts d'accéder et de modifier dynamiquement le contenu, la structure et le style de documents xml ou html –W3C



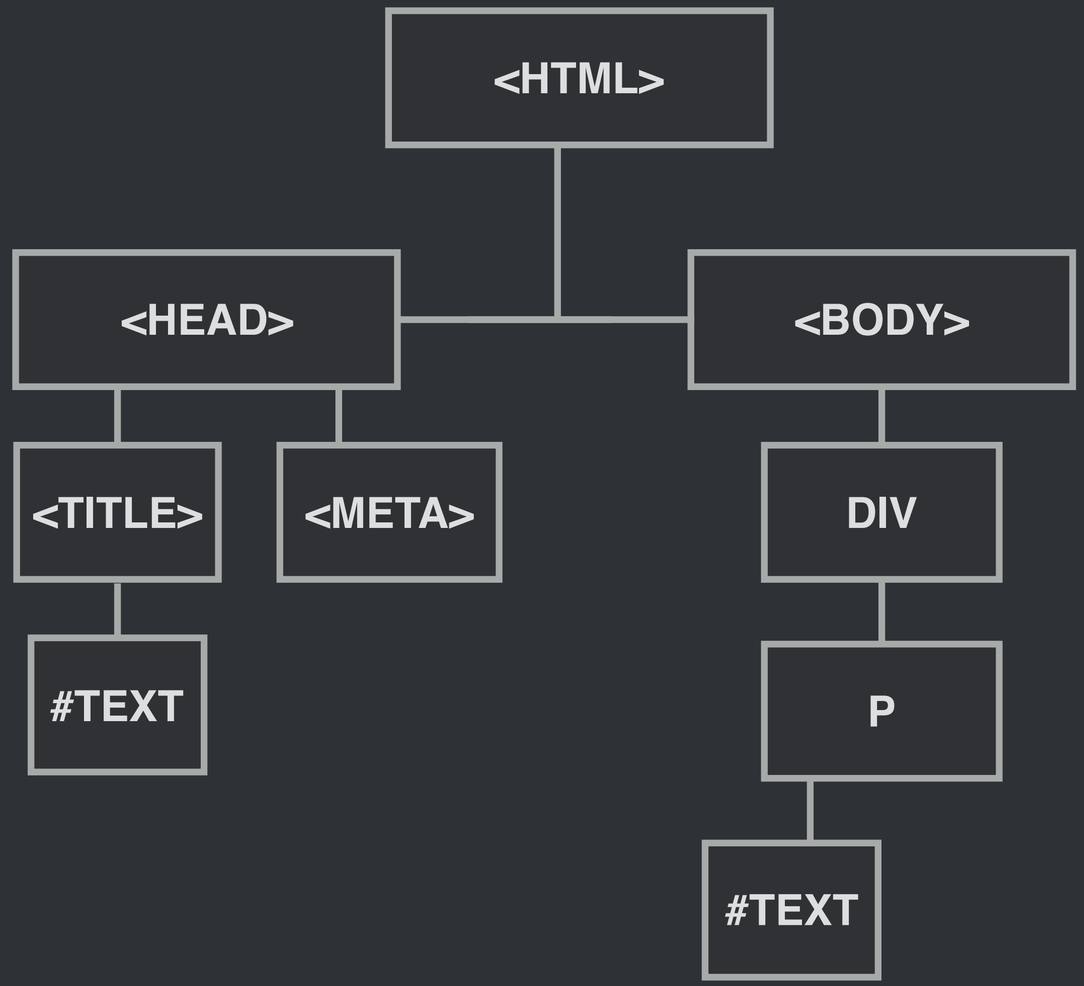
Interface

- Le DOM est donc une représentation du document en cours de consultation, à ceci près qu'elle a la particularité d'être évolutive.

```
<!DOCTYPE html>
<html lang="fr-FR" prefix="og: http://ogp.me/ns#
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-widt
<meta name="google-site-verification" content="r
<title>Clever Age - Assembleur de cocktails digi
<link rel="profile" href="http://gmpg.org/xfn/11
<link rel="pingback" href="http://www.clever-age

<link rel="apple-touch-icon" sizes="57x57" href=
<link rel="apple-touch-icon" sizes="114x114" hre
<link rel="apple-touch-icon" sizes="72x72" href=
<link rel="apple-touch-icon" sizes="144x144" hre
<link rel="apple-touch-icon" sizes="60x60" href=
<link rel="apple-touch-icon" sizes="120x120" hre
<link rel="apple-touch-icon" sizes="76x76" href=
<link rel="apple-touch-icon" sizes="152x152" hre
<link rel="apple-touch-icon" sizes="180x180" hre

<link rel="icon" type="image/png" href="http://w
<link rel="icon" type="image/png" href="http://w
<link rel="icon" type="image/png" href="http://w
<link rel="icon" type="image/png" href="http://w
<link rel="icon" type="image/png" href="http://w

<meta name="msapplication-TileColor" content="#2
```

Interface

- Le navigateur offre des méthodes permettant les manipulations sur cet objet, qu'on peut représenter simplement par un arbre.



https://developer.mozilla.org/en-US/docs/Web/API/Node

Interface Du Dom

- Créer un Elément
- Créer un attribut
- Récupérer un attribut
- Récupérer une liste
- Récupérer un element unique ID

https://developer.mozilla.org/fr/docs/Web/API/Document

```
// Create a new dom element passing
//the element tag name
var div = document.createElement('div');

// Create a node text passing node value
var node = document.createTextNode("hello
           world");

// Create attribute passing name and value
div.setAttribute('id', "main");
// <div id="main"></div>

// Get atribute value
div.getAttribute('id');
// main

// Retrieve element by ID
document.getElementById("main");

// Retrieve elemnt by tag name
document.getElementsByTagName('ul');
// Return HTML collection [ul, ul]

// Retrieve element by className
document.getElementsByClassName("list");
// Return HTML collection [list, list]

// Retrieve First element found
document.querySelector(« .list »); // class
document.querySelector(« #list »); // id
// Slow and Supported by IE8 and up
```

Interface Node Method

Insertion et remplacement dans les noeuds du dom

- appenchild()
- contains()
- removeChild()
- hasChildNode()
- inserBefore()

```
3   <nav id="topnav">
4   <span>Logo</span>
5   </nav>
6 </div>
7 */
8 // Create new element div
9 var ul = document.createElement("ul");
10 var nav = document.getElementById("topnav");
11 var span = document.getElementsByTagName('span')
12             [0];
13
14 // Add ul element to topnav element id
15 nav.appendChild(ul);
16
17 // Check if element contains ul
18 nav.contains(ul);
19
20 // Check if element has child node
21 nav.hasChildNodes();
22 // Return boolean
23
24 // Create link element a
25 var link = document.createElement("a");
26
27 // Get parent span
28 var parent = span.parentNode;
29
30 // Insert link in nav before child span
31 parent.insertBefore(link, span);
32
33 // Remove link node inside nav
34 nav.removeChild(link);
```

Interface Node Property

Ecriture de contenu

- innertHTML (HTML5 supported globaly)
- textContent

```
/*
 <div id="foo">
  <p>foo</p>
  <p>Bar</p>
</div>
*/

var foo = document.querySelectorAll("#foo p");
foo.forEach(function(p,i) {
  // InnerHTML -> parse html, slower //text/html
  // textContent -> faster, don't parse //text/plain
  console.log(p.textContent, p.innerHTML);
});
```

Interface Node Property

Référencement du noeud

- parentNode
- childNodes
- firstChild
- lastChild
- nextSibling
- previousSibling

```
/*
<main id="leftSection">
  <p>Foo</p>
  <span>Bar</span>
</main>
*/

var leftSection = document.querySelectorAll("#leftSelection")
console.log(leftSection)
```

```
▼NodeList[1] i
  ▼0: section#left
     accessKey: ""
     assignedSlot: null
   ▶ attributes: NamedNodeMap
     baseURI: "https://fiddle.jshell.net/_display/"
     childElementCount: 1
   ▶ childNodes: NodeList[3]
   ▶ children: HTMLCollection[1]
   ▶ classList: DOMTokenList[0]
     className: ""
     clientHeight: 18
     clientLeft: 0
     clientTop: 0
     clientWidth: 498
     contentEditable: "inherit"
   ▶ dataset: DOMStringMap
     dir: ""
```

Interface Class And Css

L'aspect graphique du node

- Css access node
- Class

```
/*
<div id="main">
  <nav id="topnav">
  <span>Logo</span>
  </nav>
</div>
*/
var nav = document.getElementById('topnav');

// Set css by style property of nav
nav.style.backgroundColor = "#EBEBEB"
nav.style.width = "100%";
nav.style.padding = "10px";
```

```
// List of class
nav.className
// return className string


// Add Class to element
nav.classList.add('bar');
// <nav class="top bar"></nav>

// Remove Class of element
nav.classList.remove('bar');
// <nav class="top"></nav>

// Check if class exist
console.log(nav.classList.contains('bar'));
// return boolean false
```

Coding Time 2

Time Handler & Dom Events

Events

- Un évènement est un changement d'état de l'environnement qui peut être intercepté par le code Javascript, et déclenché par :
- l'utilisateur (pression d'une touche, ...)
- le document (chargement d'une image, ...)
- ou le développeur (Custom Event)

Type Of Events

- focus
- click
- load
- keyUp
- keypress
- blur
- change
- submit

```
// HTML
// <form id="contact" action="/" method="post">
//   <label for="name">Name</label>
//   <input type="text" name="name" id="name" />
// </form>

var contact = document.getElementById('contact')
                ;
contact.addEventListener("submit", validForm,
                          false);

function validForm(e) {
  // Get target
  var target = e.target || e.srcElement;
  console.log(e, e.target);
  // Stop event sending form
  e.preventDefault();
  // Check if valid form inputs
};

// Keypress, keyup, keydown
window.addEventListener('keypress', function(
                         event) {
  console.log(event.key ,event.keyCode);
},false);

```

Settimeout

setTimeout(function, delay)
clearTimeout(function)

- exécute une fonction après un delay

```
1 var count = 0;
2
3 function log() {
4   if (count < 5) {
5     setTimeout(log, 500);
6   }
7
8   console.log('Time: ' + count++);
9 }
10
11 log();
```

Setinterval

setInterval(function, time millisecond)
clearInterval(function)

- Appel d'une fonction par répétition avec un délai en millisecond

```
1  // Declare var
2  var timer, count = 0;
3
4  function log() {
5    if (count >= 5) {
6      //clear events
7      clearInterval(timer);
8    }
9    // Iterate on count
10   console.log('Time: ' + count++);
11 }
12 // SetIterval timer init
13 timer = setInterval(log, 500);
```

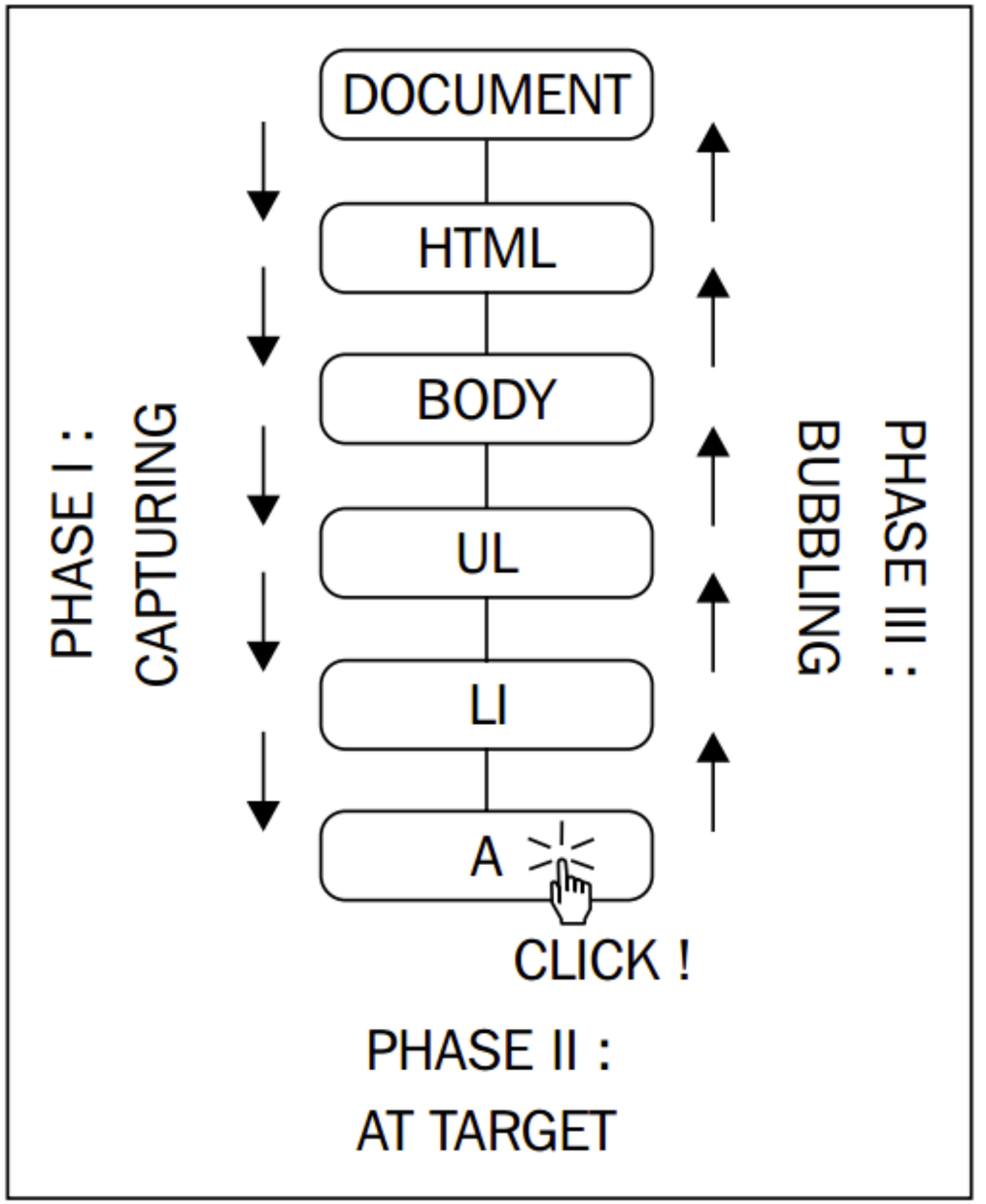
Dom Event Listeners

- addEventListener(DOMString type, EventListener event, boolean useCapture)
- removeEventListener(DOMString type, EventListener event, boolean useCapture)

```
// <!DOCTYPE html>
// <html>
// <head>
//     <title>Lorem</title>
// </head>
// <body>
//         <div id="main">
//             <strong>Hello world</strong>
//         </div>
// </body>
// </html>

var  main = document.getElementById('main');
main.style.cursor = 'pointer';

main.addEventListener('click', function(event){
    var target = event.target || event.
                 srcElement;
    event.stopPropagation();
    console.log("Clicked", target);
},false);

main.removeEventListener('click', null, false);
```

Bubbling

- Capture
- cible
- bouillonnement



Natif vs Jquery

- On
- Off
- Ready

```
1
2  // On event no jquery
3  el.addEventListener('onclick', function() {
4    //execute some code
5  });
6
7  // On event jquery
8  $(el).on('click', function(e){
9    //execute some code
10 });
11
12 // Document ready function in pure javascript
13 function ready(cb) {
14   if (document.readyState != 'loading'){
15     cb();
16   } else {
17     document.addEventListener('DOMContentLoaded'
18                                , cb);
19   }
20 }
21 ready(function(){
22   // Code to excetute when dom is ready
23 })
24
25 // Document ready in jquery
26 $(document).ready(function(){
27   // Code to excetute when dom is ready
28 });
```

Coding Time 3


Function

Function

- Function est un objet représente par le mot clé **function**.
- C’est un bloc de code à exécuter plus tard.
- On peut passer des arguments à la function.

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Fonctions

Function

Déclaration et expressions
Passage de paramètres

```
// Declare function with no params call log,
// call it after with name of the function and ();
function log() {
  console.log('Hello world');
}
log();


// Declare function anonymous in square variable // Not good
var square = function (a, b) {
 return a * b;
};

// Declare a function call fx // OK
function square(a, b) {
  return a * b;
};

// Declare squareFX as function in square var //  Better
var square = function squareFx(a, b) {
 return a * b;
};

// Declare an obj wich contains square key who hold
// square function
var obj = {
    square: function square(a, b) {
     return a * b;
   }
 };
```

Function

Déclaration et expressions
Portée de la function

```
1 var x = 42;
2
3 function x() { console.log('42'); }
4 x(); // => TypeError: number is not a function
5
6
7 // Explain
8 // Hoisting is when javascript pull up function
9 // on top before var
10 function x() { console.log('42'); }
11
12 // Hoisting var;
13 var x;
14
15 // continue
16 x = 42;
17
18 x();
19 // => TypeError: number is not a function
20
21 ///// Solution
22 var x = 42;
23 x = function() { console.log('youpi'); }
24 x();
25
```

Function

Closure

- La closure est une fonction qui conserve une référence de variables statiques propres.
- La valeur des variables contenues dans la fonction est déterminée indépendamment lors de chaque invocation.

```
1  function closureInc(){
2      var i = 0;
3      return function increment(){
4          return ++i;
5      }
6  }
7
8  var incrementA = closureInc();
9  var incrementB = closureInc();
10
11 console.log(incrementA());
12 // Show 1
13 console.log(incrementA());
14 // Show 2 because var i is static
15 console.log(incrementB());
16 // Show 1 each closure have
17 // a independent reference to i
```

Function

IIFE

- Fonction qui s’exécute d’elle même.
- Avec cette écriture les var ne sont pas accessible depuis l’extérieur.

```
1 (function(){
2   var a = 'hello world';
3 })();
4 console.log(typeof a);
5 // Show undefined
6
7
8 // Pattern IIFE
9 (function(){
10   console.log('Hello world');
11 })();
12
13 (function bar(){
14   console.log('Hello world from bar function');
15 })();
16
17 var bar = function(){
18   console.log('Hello world from function stored in bar var');
19 }();
```

Ajax &
Dialog Server

Ajax

AJAX : **A**synchronous **J**avascript **A**nd **X**ML.
Existe depuis 1998, permet de dialoguer avec le serveur en asynchrone.
Sans Ajax la page web fait une requête au serveur, renvoie du contenu et reload la page.
Ajax évite le rechargement de page, augmente la réactivité de l'application, transforme une page web en **RIA**

Utilisation & Principe

- Orienté vers la manipulation des données (Data Driven)
- Historiquement à l'aide de XML
- Aujourd’hui, JSON
- Affichage dynamique et intéraction via le DOM modèle
- Appel Asynchrone des données (Objet XmlHttpRequest)
- Langage javascript pour lier le tout.

Pourquoi

- Améliorer la performance (réduire le temps de transit de la data)
- Améliorer l’experience utilisateur
- Réduction du temps de rafraîchissement = réduction pertes d’attention
- Conserver la simplicité et la lisibilité de l’application

Risque

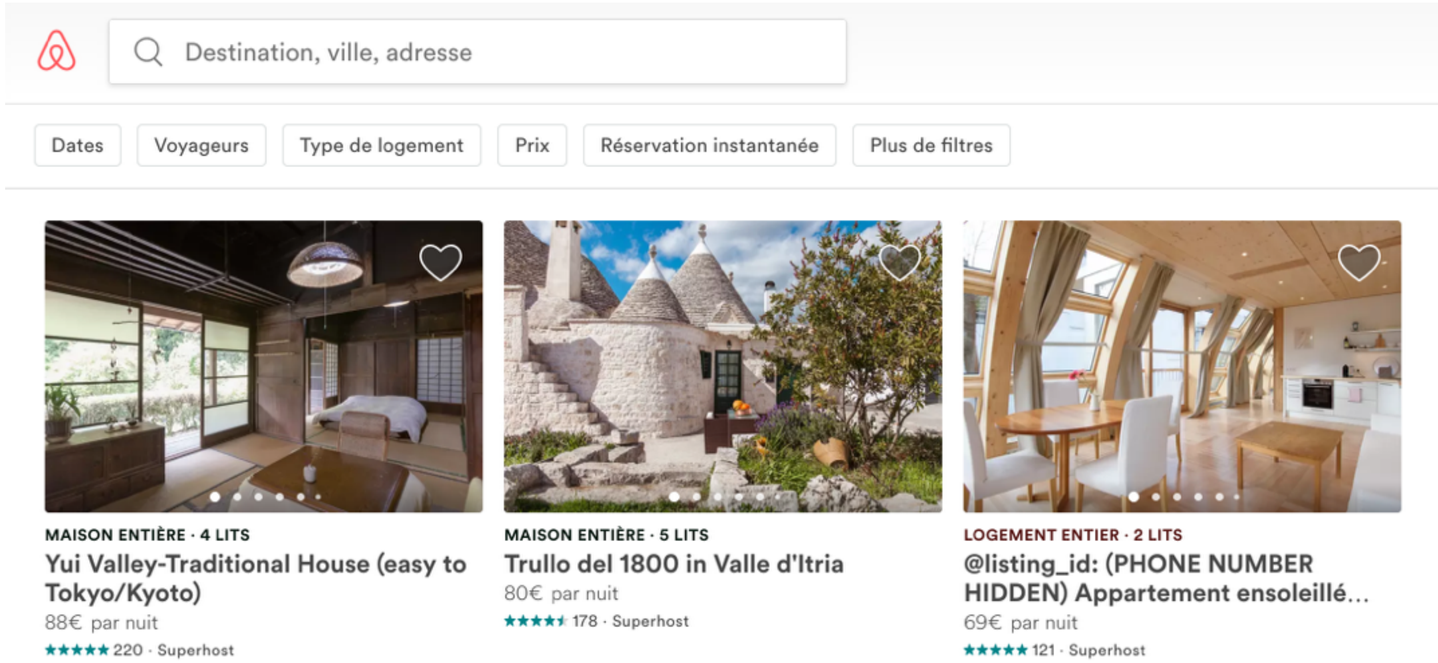
- < IE 8 problème de compatibilité et debug difficile.
- Différence entre les navigateurs web.
- Contexte d’éxécution uniquement côté client

Contexte Favorable

- Travailler avec le back-end pour avoir de bon webservices et privilégier l'utilisation d'API REST (Relationship between URL and HTTP methods) utilisation de HTTP VERBS.
- Côté client on peut faire de l'Ajax via Javascript natif ou via des librairies Javascript.

Exemples D’utilisation

- Login / signup
- search
- Edit profil
- autocompletion
- filtrage de produits
- Bref partout …



Httpverb & Status

- XMLHttpRequest.readyState -> état de la requête
- 0 UNSENT Client has been created. open() not called yet.
- 1 OPENED open() has been called.
- 2 HEADERS_RECEIVED send() has been called, and headers and status are available.
- 3 LOADING Downloading; responseText holds partial data.
- 4 DONE The operation is complete.

```
1  // HTTP VERB -> CRUD
2
3  /* POST -> create
4  GET -> read
5  PUT -> update/replace
6  PATCH -> update/modify
7  DELETE -> delete */
8
9  // HTTP STATUS
10 // 200 : OK
11 // 201 : created
12 // 301 : Moved Permanently
13 // 302 : Temporarily
14 // 400 : Bad Request
15 // 401 : Unauthorized
16 // 403 : Forbidden
17 // 404 : Not Found
18 // 500 : Internal Server Error
19 // 503 : Service Unavailable
```

Ajax By Exemple

Call AJAX GET request

```
// Instanciate a new  XMLHttpRequest object
call request
var request = new XMLHttpRequest();

// This function is trigger when
onreadystatechange property change
request.onreadystatechange = function() {
  // Readystate property complete
  if (this.readyState === 4) {
    if (this.status >= 200 && this.status < 400)
        {
      // Success!
      var resp = this.responseText;
    } else {
      // Error :(
    }
  }
};
// Open request  with params as  HTTP VERB and
URL
request.open('GET', '/echo/json/', true);

 // Open post request
request.open('POST', '/add', true);

// Send the request for get
request.send();
// Send the request for post
request.send(data);
```

L'objet Xmlhttprequest

Method

- Open(method, url async)
- Send(contenu)
- setRequestHeader(type, value)
  - used for post ajax call

```
1
2  // Instanciate a new  XMLHttpRequest object
3  call request
4  var request = new XMLHttpRequest();
5
6  // This function is trigger when
7  onreadystatechange property change
8  request.onreadystatechange = function() {
9    // Readystate property complete
10   if (this.readyState === 4) {
11     if (this.status >= 200 && this.status < 400)
12         {
13       // Success!
14       var resp = this.responseText;
15     } else {
16       // Error :(
17     }
18   }
19 };
20 // Open request  with params as  HTTP VERB and
21 URL
22 request.open('GET', '/echo/json/', true);
23
24 // Send the reuest
25 request.send();
```

L'objet Xmlhttprequest

Property

- readyState -> état de la requête
- status -> code http de la requête
- responseText
  - données retournées par le serveur
- statusText -> code de status
- onreadystatechange

```
1
2  // Instanciate a new  XMLHttpRequest object
3  call request
4  var request = new XMLHttpRequest();
5
6  // This function is trigger when
7  onreadystatechange property change
8  request.onreadystatechange = function() {
9    // Readystate property complete
10   if (this.readyState === 4) {
11     if (this.status >= 200 && this.status < 400)
12         {
13       // Success!
14       var resp = this.responseText;
15     } else {
16       // Error :(
17     }
18   }
19 };
20 // Open request  with params as  HTTP VERB and
21 URL
22 request.open('GET', '/echo/json/', true);
23
24 // Send the reuest
25 request.send();
```

Type Of Data To Receive

- xml
- text
- json
- html

```
1
2  // Instanciate a new  XMLHttpRequest object
3  call request
4  var request = new XMLHttpRequest();
5
6  // This function is trigger when
7  onreadystatechange property change
8  request.onreadystatechange = function() {
9    // Readystate property complete
10   if (this.readyState === 4) {
11     if (this.status >= 200 && this.status < 400)
12         {
13       // Success!
14       var resp = this.responseText;
15     } else {
16       // Error :(
17     }
18   }
19 };
20 // Open request  with params as  HTTP VERB and
21 URL
22 request.open('GET', '/echo/json/', true);
23
24 // Send the reuest
25 request.send();
```

Ajax and IE

- non supporté par toutes les versions d'IE. Pour les versions inférieures à IE7, on passe par une contrôle ActiveX

```
var xhr = null;
if (window.XMLHttpRequest || window.
    ActiveXObject) {
    if (window.ActiveXObject) {
        try {
            xhr = new ActiveXObject("Msxml2.
                XMLHTTP");
        } catch (e) {
            xhr = new ActiveXObject("Microsoft.
                XMLHTTP");
        }
    } else {
        xhr = new XMLHttpRequest();
    }
} else {
    alert("Votre navigateur ne supporte pas
          l'objet XMLHTTPRequest...");
    return;
}
```

Jquery Ajax?

- $.ajax
- post / get
- $.json

```
1  // GET request handling succes and error
2  $.ajax({
3    type: 'GET',
4    url: '/my/url',
5    success: function(resp) {
6      console.log(resp);
7    },
8    error: function() {
9      console.log("fail request");
10   }
11 });
12
13 // POST request with data params
14 $.ajax({
15   type: 'POST',
16   url: '/my/url',
17   data: data
18 });
19
20 // Load JSON
21 $.getJSON('/my/url', function(data) {
22   console.log(data);
23 });
```

Coding Time 4

Good Practices

Convention De Nomage

- http://jstherightway.org/#getting-started

Table des matières

Préface

Les sections suivantes décrivent un guide de style *raisonable* pour tout développement moderne en JavaScript et ne sont pas censées être normatives. Le plus important à appliquer est la **loi de cohérence dans le style de code**. Le style que vous avez choisi pour votre projet, quel qu'il soit, doit être considéré comme la loi. Référer à ce document pour afficher l'engagement de votre projet à utiliser un style de code cohérent, lisible et maintenable.

Jshint

- Outil qui analyse les erreurs de code et de syntaxe. Suivant un fichier convention de nommage.

- .jshintrc : fichier de nomenclature
- .jscsrc : fichier d’utilisation de mots clés javascript

```
{
  "disallowEmptyBlocks": true,
  "disallowImplicitTypeConversion": ["binary",
  "string"],
  "disallowKeywords": ["eval"],
  "disallowMixedSpacesAndTabs": true,
  "disallowMultipleLineStrings": true,
  "disallowNewlineBeforeBlockStatements": true,
  "disallowOperatorBeforeLineBreak": ["."],
  "disallowPaddingNewlinesBeforeKeywords": [
  "catch"],
  "disallowPaddingNewlinesInBlocks": true,
  "disallowSpaceAfterPrefixUnaryOperators":
  true,
  "disallowSpaceBeforeComma": true,
  "disallowSpaceBeforePostfixUnaryOperators":
  true,
  "disallowSpacesInCallExpression": true,
  "disallowSpacesInFunctionDeclaration": {
    "beforeOpeningRoundBrace": true
  }
}
```

Jsdoc

- Outils de generation de documentation
- Parse le fichier JS et render une documentation du projet en html
- http://usejsdoc.org/

@use JSDoc

Getting Started with JSDoc 3

Table of Contents

- Getting Started
- Adding Documentation Comments to Your Code
- Generating A Website

Getting Started

JSDoc 3 is an API documentation generator for JavaScript, similar to JavaDoc or PHPDoc. You add doc
JSDoc Tool will scan your source code, and generate a complete HTML documentation website for you.

Adding Documentation Comments to Your Code

JSDoc's purpose is to document the API of your JavaScript application or library. It is assumed that you
etc.

JSDoc comments should generally be placed immediately before the code being documented. It must s
beginning with `/*`, `/***`, or more than 3 stars will be ignored. This is a feature to allow you to suppress pars

The simplest documentation is just a description.

```
/** This is a description of the foo function. */
function foo() {
}
```

Debugging Console

- Debug tools pour Chrome
- Firebug pour Firefox
- Drosera pour Safari
- Debug Interne pour >IE8

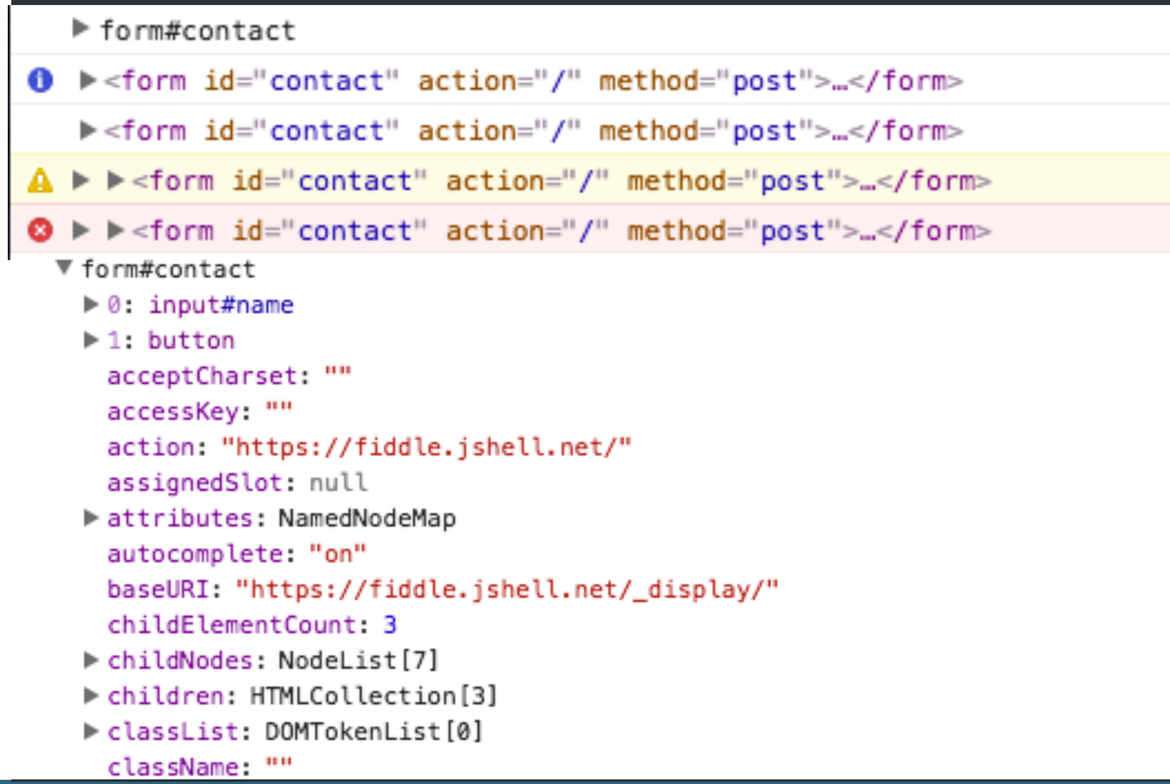
Debugging ?

- Identifier le code exécuté
- Créer des breakpoint(s) pour voir ce qui se passe de manière détaillée.
- Recharger la page pour relancer l'exécution du script
- Attendre que le debugger permette l'exécution du code pas à pas
- Observer les valeurs des variables (undefined, true/false)
- Utilisation de la command-line pour changer la valeurs de certaines variables
- Identifier précisément le problème à partir des erreurs

Debugger Word &Console

- debbuger ajoute un breakpoint manuellement dans la console Chrome.
- console object
- log, dir, info, warn

```
1  // Add debugger keywords;
2  debugger;
3  // Console object methods
4  console.dir(contact)
5  console.info(contact)
6  console.log(contact)
7  console.warn(contact)
8  console.error(contact)
   console.table(contact)
9  console.time("Start timing")
10 // some code
11 console.timeEnd("End timing")
```



Data & Loop

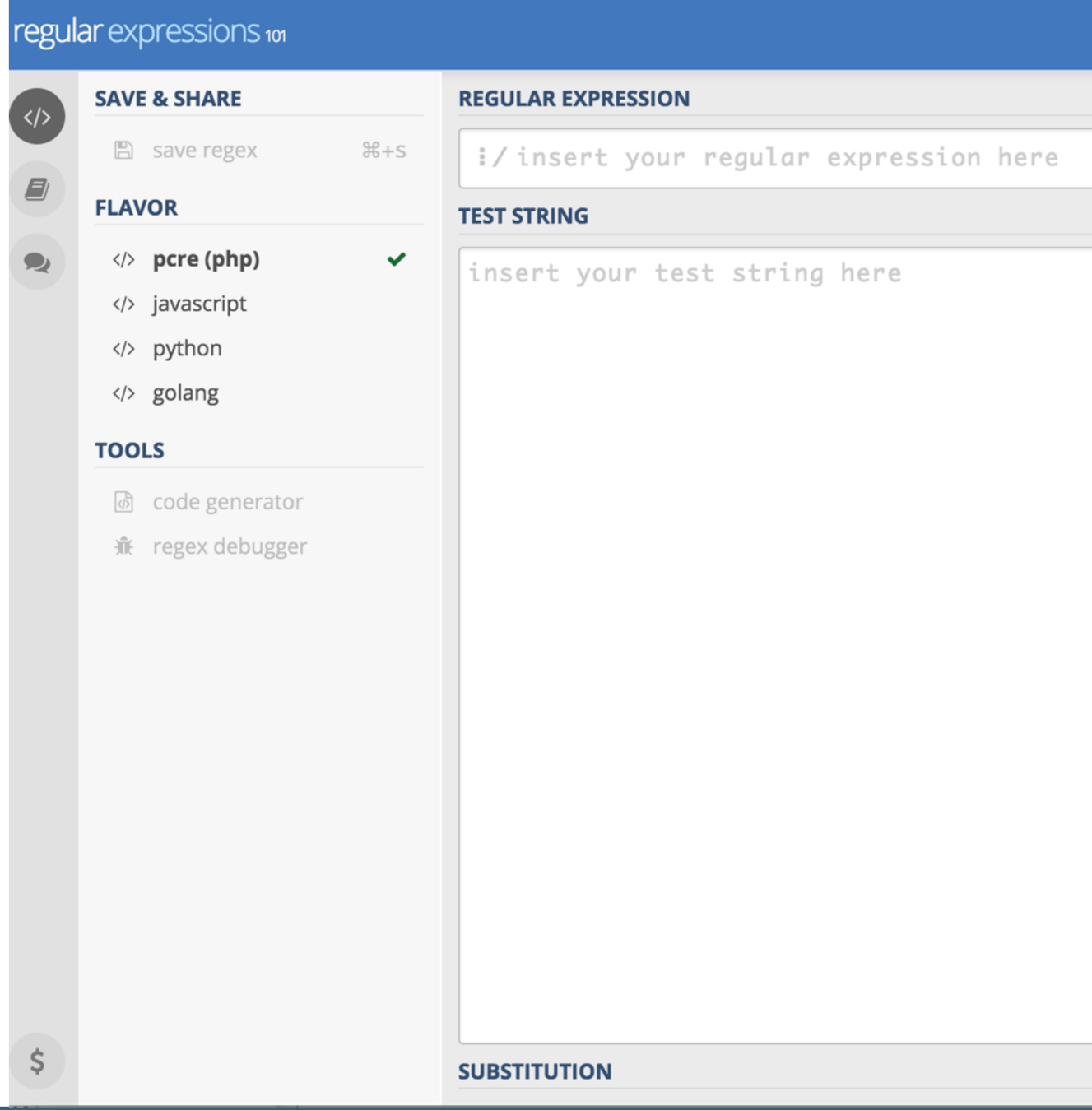
Json

- JSON parse
- JSON strinfigy

```
1 // Converting string to JSON objet
2 var data = '{"name":"thomas","age":"33"}';
3 var obj = JSON.parse(data);
4 //  {name: "thomas", age: "33"} typeof object
5
6
7 // Converting JavaScript data into JSON text
8 var simpleData = {
9   name : 'thomas',
10  age : '33'
11 }
12
13 var transFormToObj = JSON.stringify(simpleData);
14 //{"name":"thomas","age":"33"} typeof string
```

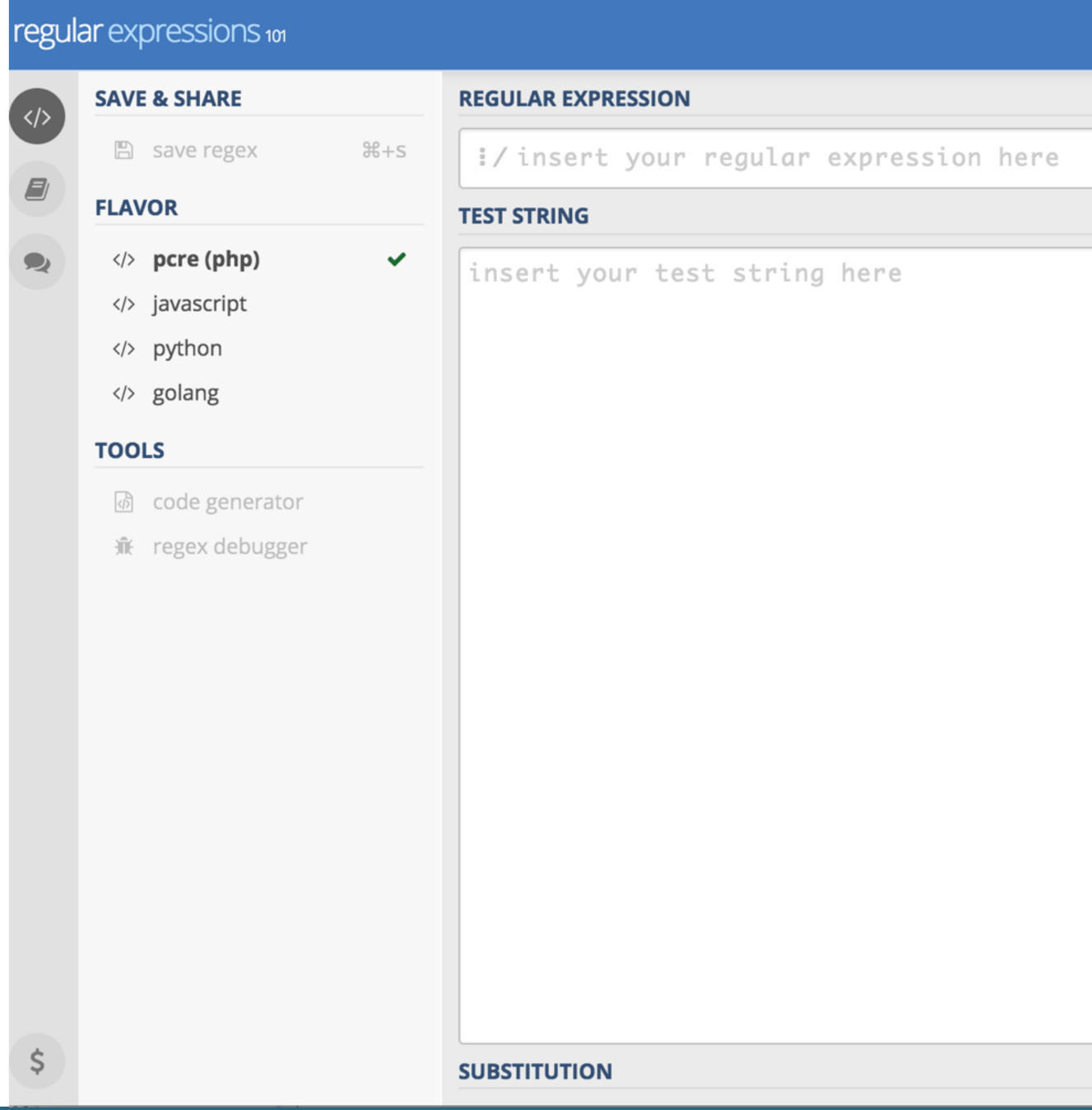
Regular Expressions

- Object
- declare pattern or new RegExp()
- / to start / to finish
- method : test,replace …
- https://regex101.com/ to test



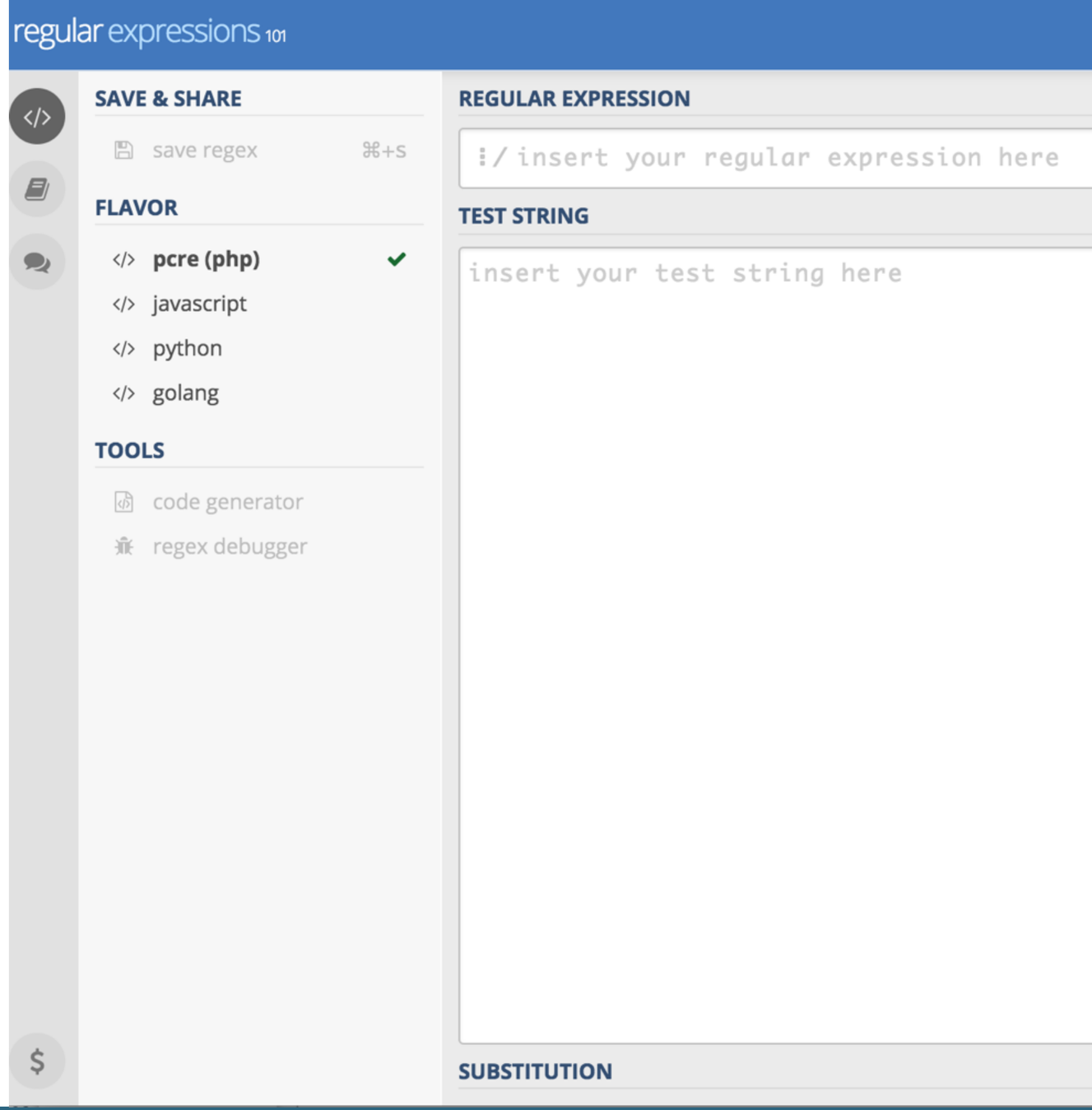
Pattern Regex

- a-z correspond aux 26 caractères de la langue française
- 0-9 correspond à tous les chiffres
- La négation est indiquée grâce au caractère ^
- Pour utiliser les caractères spéciaux, il faut les échapper ou les utiliser à l'aide de l'expression



Pattern Regex

- [aeiouy] : toutes les voyelles
- [^aeiouy] : toutes les consonnes
- [a-z-] : toutes les lettres de A à Z, suivie du caractère -
- Tous les caractères : \w ([A-Za-z0-9_])
- Tous les chiffres : \d ([0-9])




Nodejs & More

Nodejs

- **Node.js®** est un framework construit sur le moteur JavaScript V8 de Chrome.
- Node.js fournit un écosystème **NPM** -> bibliothèque open source de modules.

https://nodejs.org/en/

Nodejs & Npm

- helloworld file js
- npm init, install, -g
- node_module

```

// Type node file.js in the console
$ node helloworld.js

//In helloworld.js file
console.log('Hello World');

// Node will excute this file as javascript;
Hello World !!

// NPM
// First npm init
// Answer question -> create json package.json

// Find a module in npm and install it
npm install name_of_the_module --save

// Install module globally on your machine
npm install -g
```

Future Of Javascript

- ES6
- Babel