IUT Paul Sabatier
Département Informatique
BUT 3 AGED

R5.C.04
Le langage JavaScript

Julien Broisin
julien.broisin@irit.fr
https://www.irit.fr/TALENT/site

Objectifs de la ressource

- Compléter les techniques de développement web pour arriver à des capacités de développement full-stack
- Programmation web côté client
- Communication asynchrone entre client et serveur
- Utilisation de librairies

Contenu de la ressource

- Le langage JavaScript
- La gestion des évènements
- Ajax (Asynchronous Javascript And XML)
- Librairie Jquery

Espace de cours Moodle

- https://moodle.iut-tlse3.fr/course/view.php?id=1093
- Tous les supports
  - Cours magistraux (CM)
  - Travaux pratiques (TP)
- Forum pour les questions/réponses

Équipe enseignante

- Un enseignant du département…
  - Julien Broisin (principalement en CM)
- …et une DCE (doctorante contractuelle chargé d'enseignement)
  - Maéva Kurtz (presque tous les TP, des 2 groupes)

Évaluation

- 2 évaluations
  - Contrôle écrit à la fin du module
  - Projet à rendre
    - Réalisé en binôme
    - Réalisable sur les heures de TP
    - Rendu avant de partir en stage

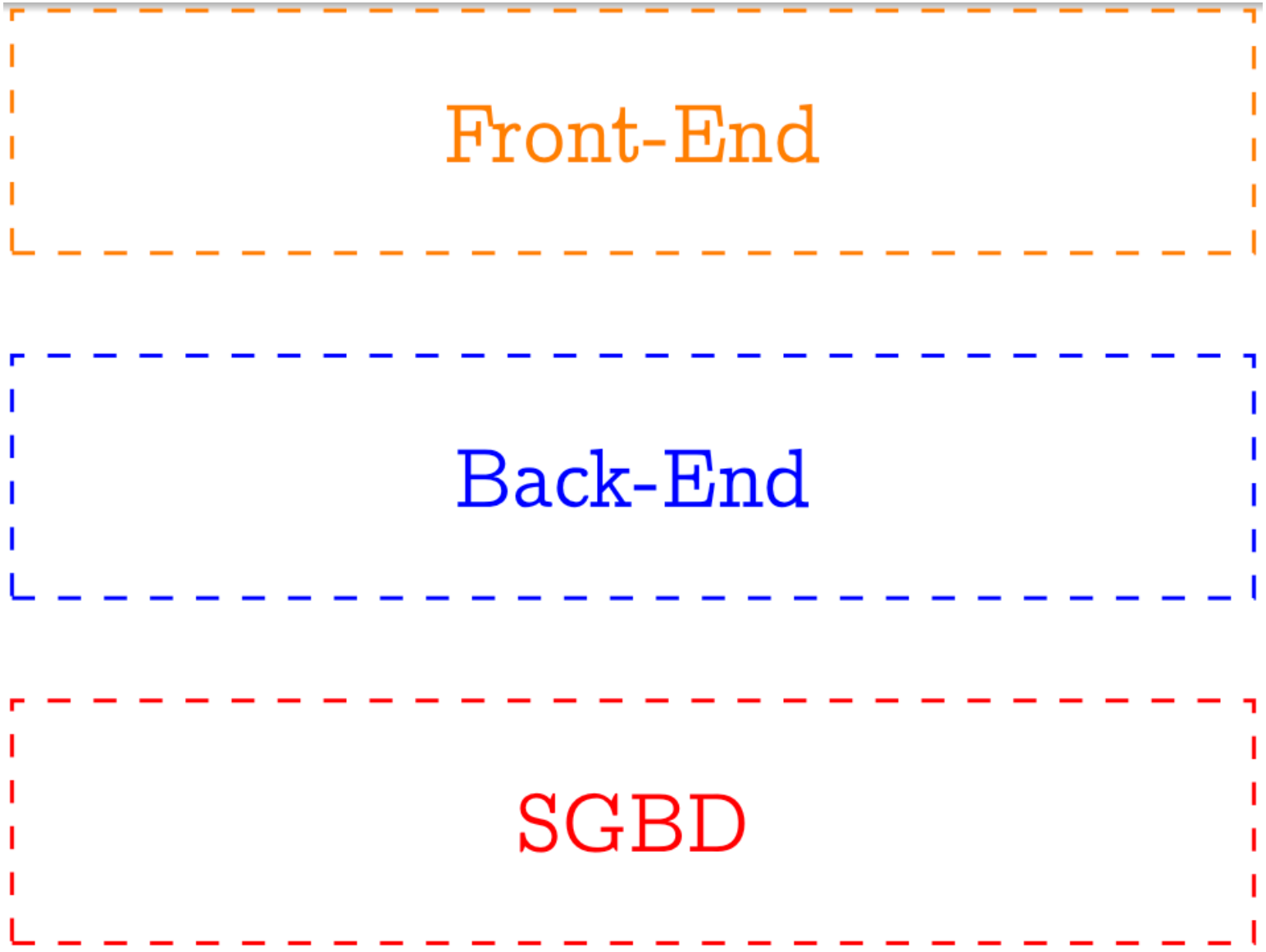
IUT Paul Sabatier
Département Informatique
BUT 2

R4.A.10 - Complément web

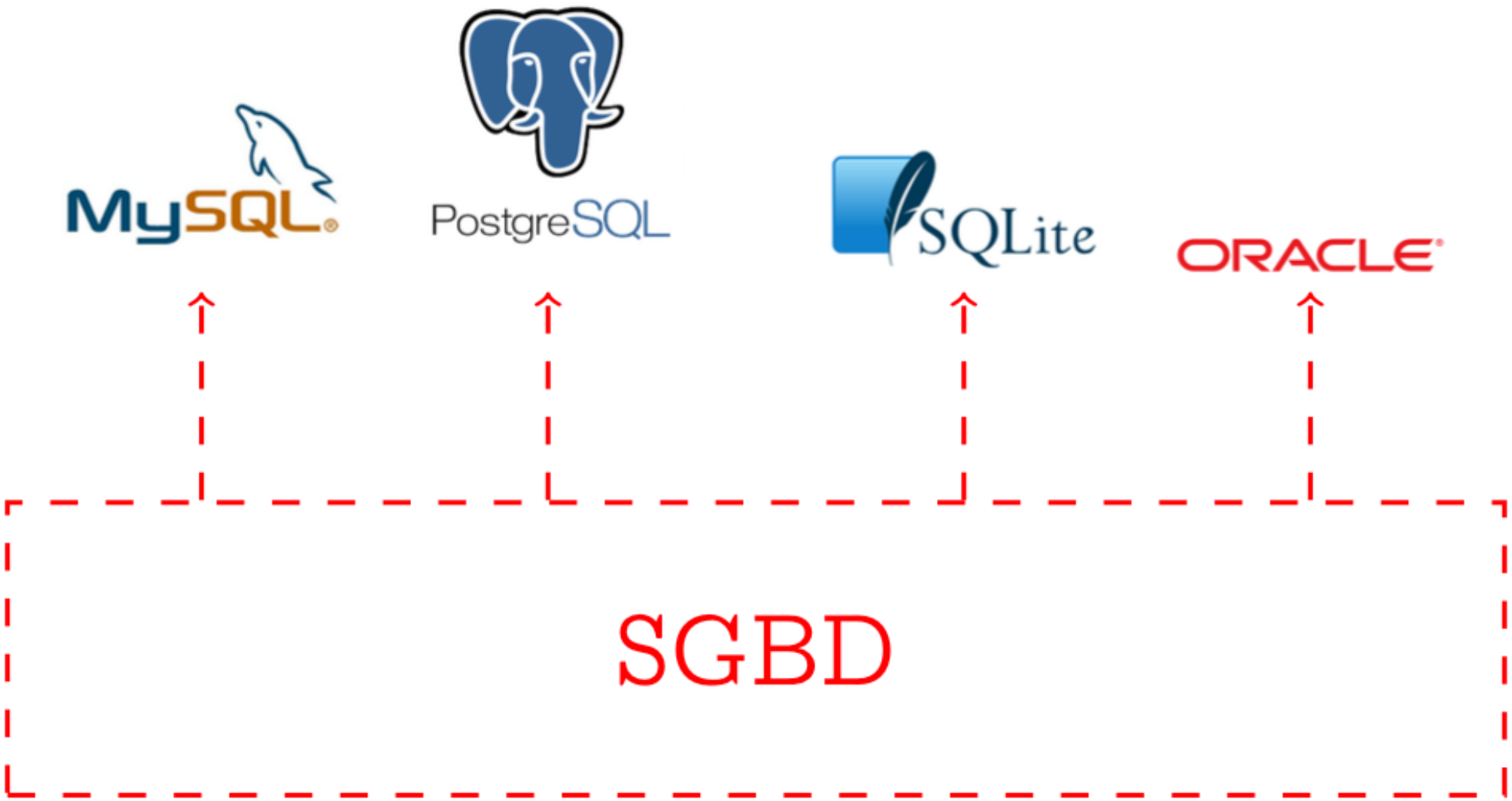
Le langage JavaScript

Table des matières

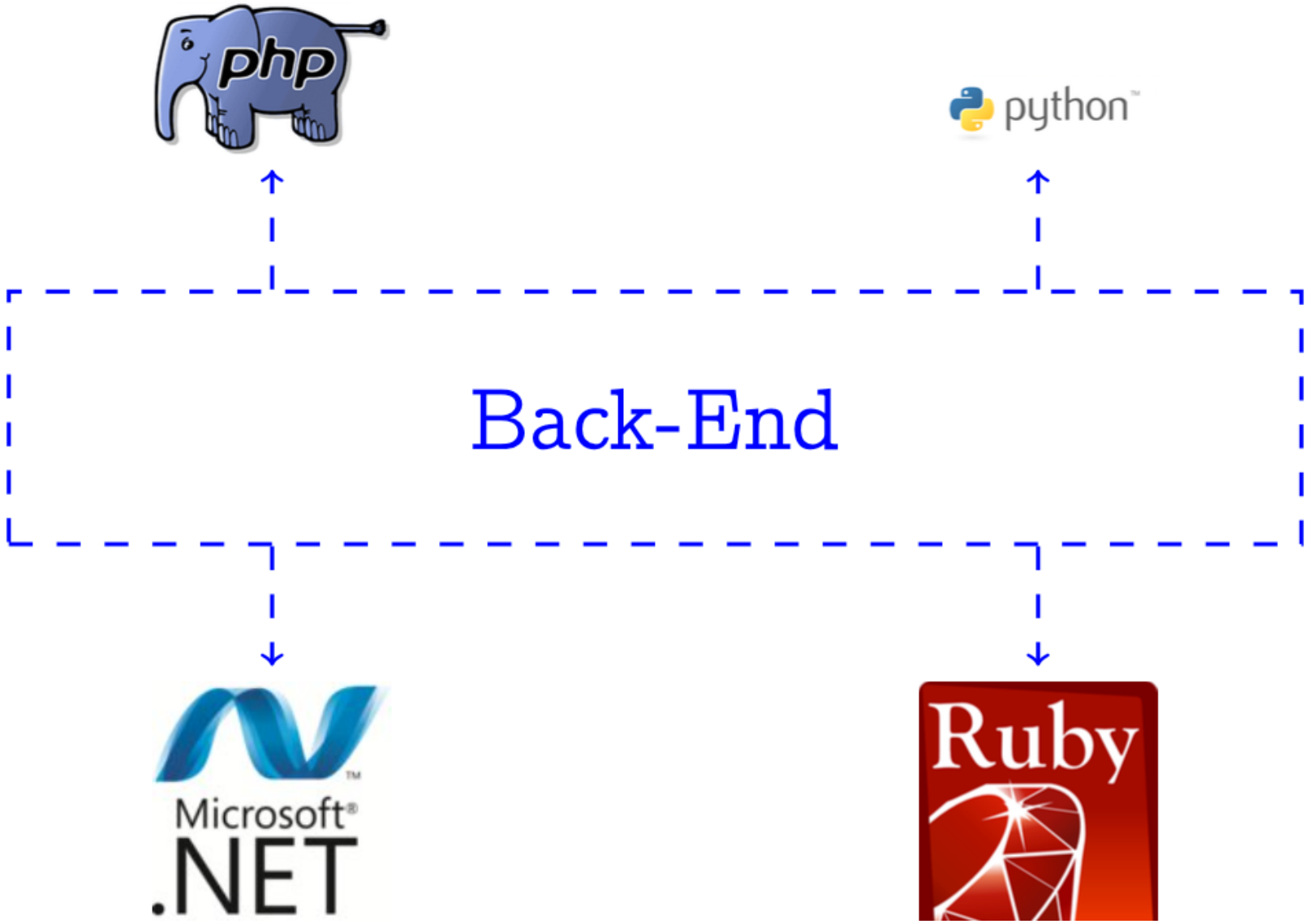
Applications web



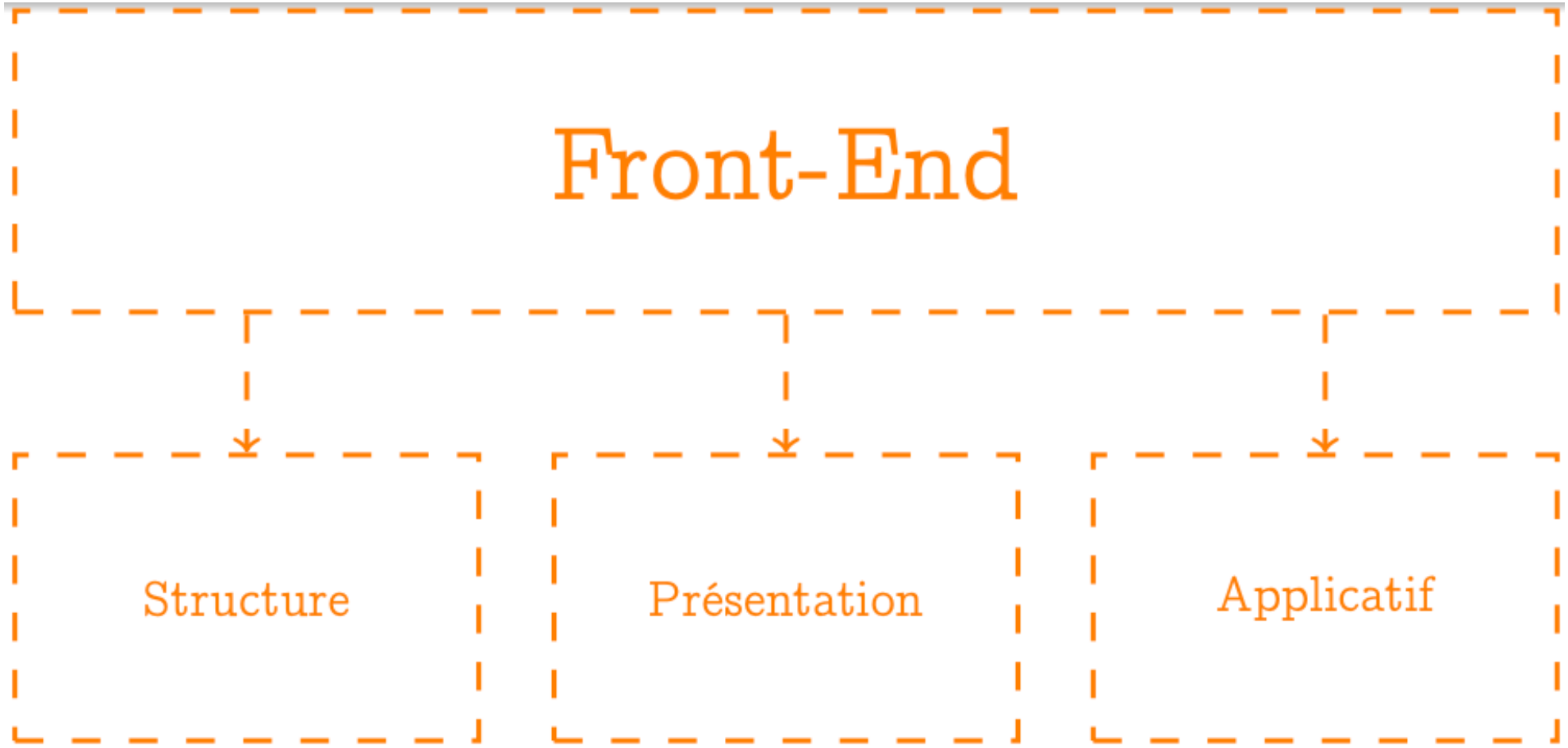
Applications web



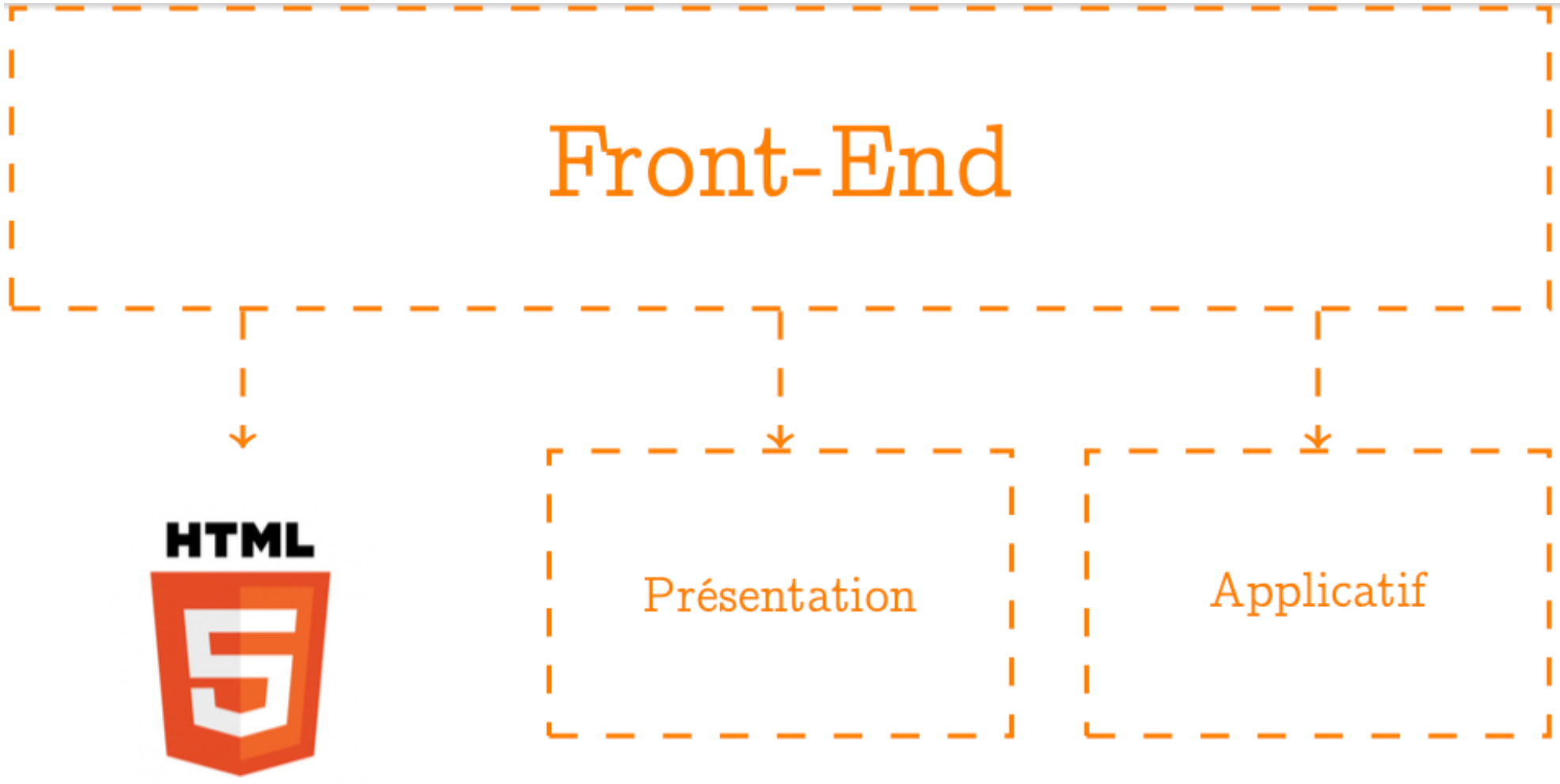
Applications web



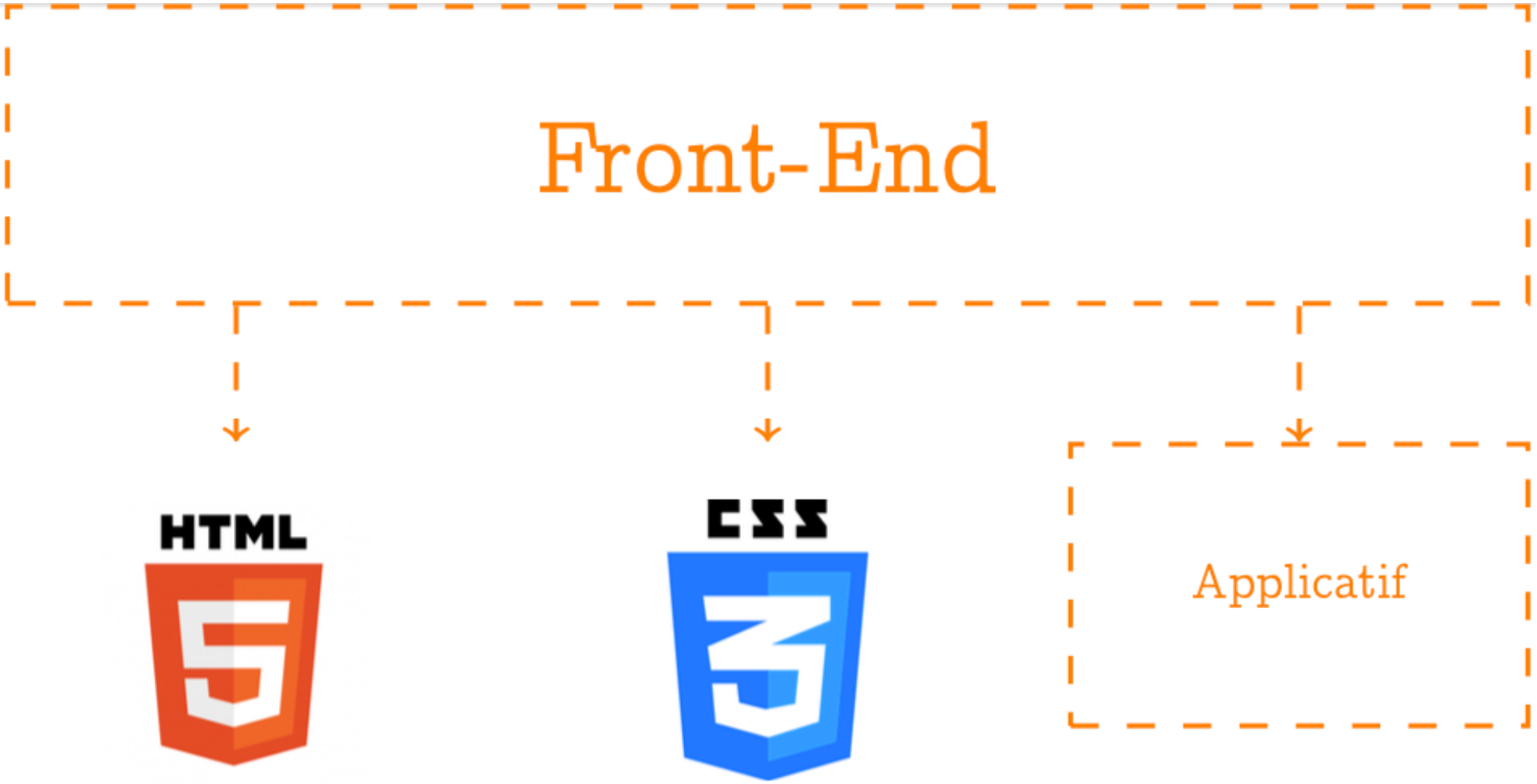
Applications web



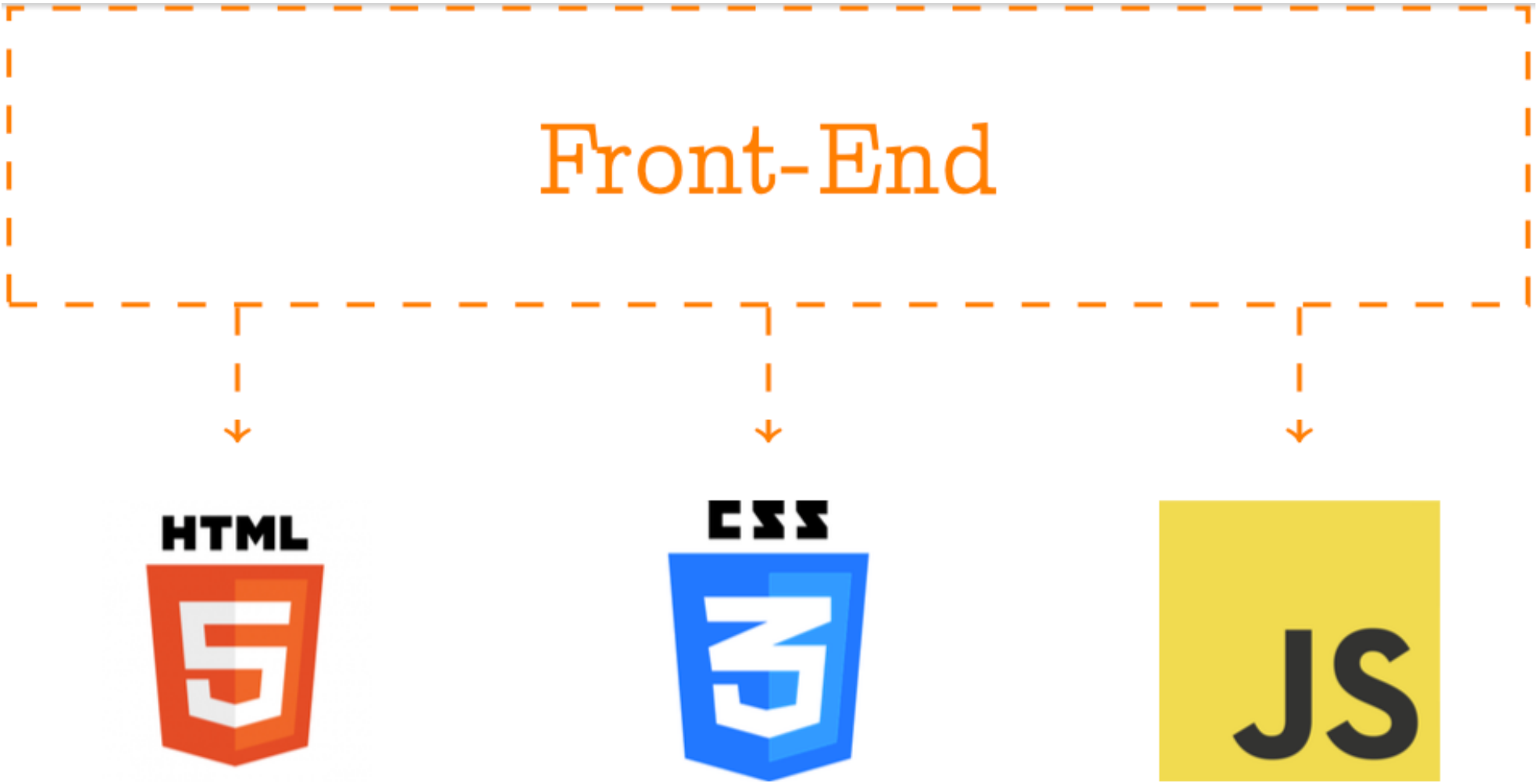
Applications web



Applications web



Applications web



Qu'est-ce que JavaScript ?

- Une extension du langage HTML
  - Apporte des améliorations en permettant d'exécuter des commandes
- Code JavaScript (JS) intégré dans le code HTML
  - Interprété par le navigateur

```
<script>
    //Placez ici le code de votre script
</script>
```

Qu'est-ce que JavaScript ?

- Autant de moteurs Javascript que de navigateurs…
  - SpiderMonkey pour Firefox (écrit en C)
  - JavaScriptCore pour Safari (écrit en C++)
  - V8 JavaScript engine pour Chrome (écrit en C++)
  - Carakan pour Opéra
  - Chakra pour Internet Explorer
- Comportements différents selon les navigateurs !!

Qu'est-ce que JavaScript ?

- Beaucoup d'APIs…
  - Ajax (XMLHttpRequest) : requête HTTP asynchrone
  - WebWorkers : programmation parallèle
  - WebSockets : communication bidirectionnelle entre le Client et le Serveur
- …de bibliothèques et *frameworks*
  - Jquery
  - Node.js, Angular.js, Prototype.js, Dojo, etc.

JavaScript : pour faire quoi ?

- Modifier le contenu du document HTML
  - Possibilité d'utiliser les simples et doubles quotes

```
document.getElementById("demo").innerHTML = "Hello JavaScript”;

document.getElementById('demo').innerHTML = 'Hello JavaScript';
```

- Modifier la valeur d'un attribut HTML

```
document.getElementById('myImage').src='picture.gif'
```

JavaScript : pour faire quoi ?

- Modifier les styles du document HTML (CSS)

```
document.getElementById("demo").style.fontSize = "25px";
```

- Afficher/masquer des éléments HTML

```
document.getElementById("demo").style.display = "none";

document.getElementById("demo").style.display = "block";
```

JavaScript : pour faire quoi ?

- Programmation évènementielle
  - Changer une image au survol du pointeur
  - Voir plus loin
- Effectuer des calculs
  - Lire la valeur saisie
  - Effectuer un traitement
  - Afficher le résultat
- Accéder à des bases de données
- etc.

Principe de base

- Le noyau comporte
  - Des types de données (Number, String, Object, etc.)
  - Des fonctions prédéfinies
  - Des opérateurs et structures de contrôle
- La couche pour le navigateur
  - Fenêtres
  - Formulaires
  - Images

Table des matières

Syntaxe

- Opérateurs, boucles, structures conditionnelles similaires au C
- ; pour marquer la fin d'une instruction (pas obligatoire)

```
for (var count=0;count<5,count++)
{
  print("valeur="+count);
}
var count=0;
var fini=false;
while(!fini)
{
  if (count>4)
    fini=true;
  else
  {
    print("valeur="+count);
  }
  count++;
}

var count=0;
do
{
  print("valeur="+count);
}while(count<5);
```

Commentaires

- Conventions utilisées en C/C++
  - // pour commenter une seule ligne
  - /* ..... */ pour commenter plusieurs lignes

```
<script>
    // Une ligne de commentaire

    /* Un ensemble
     de lignes
     commentées
    */
</script>
```

Les sorties Javascript

- Les données peuvent être affichées de différentes manières
  - dans un élément HTML : `innerHTML`
  - dans le document HTML : `document.write()`
  - dans une boîte d'alerte : `window.alert()`
  - dans la console du navigateur : `console.log()`

La propriété innerHTML

- Accès à un élément HTML

```
document.getElementById(id)
```

  - `id` spécifie l'élément HTML

- Modification du contenu de l'élément HTML

```
document.getElementById("demo").inne
rHTML = 5 + 6; //affecte 11 à
l'élément demo
```

La méthode write()

- A n'utiliser que pour des tests

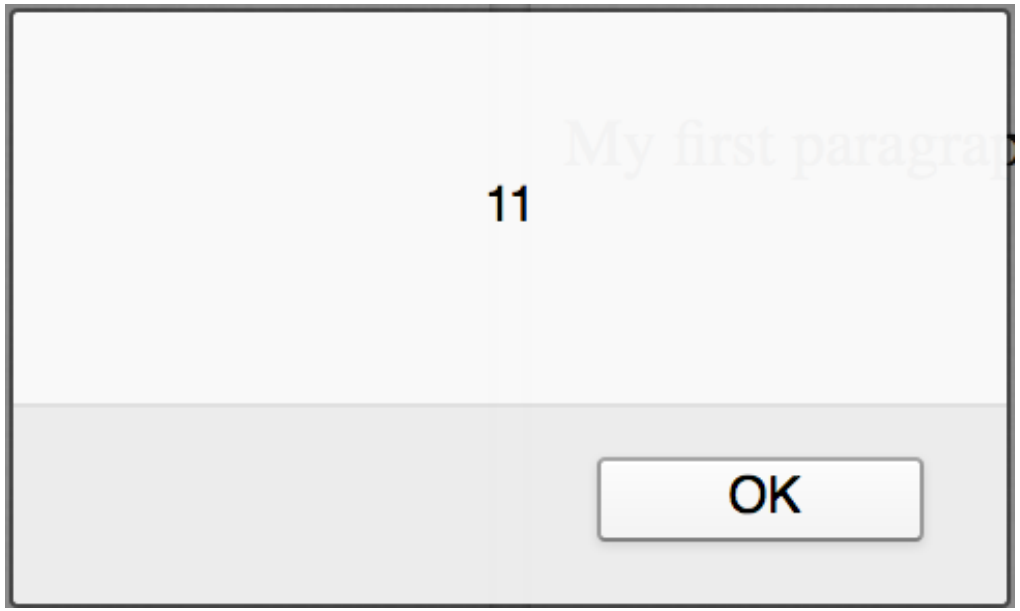
```
document.write(5 + 6);
```

- L'appel à `document.write()` après le chargement d'un document HTML supprime le contenu HTML existant !!

La méthode window.alert()

```
window.alert(5 + 6);
```

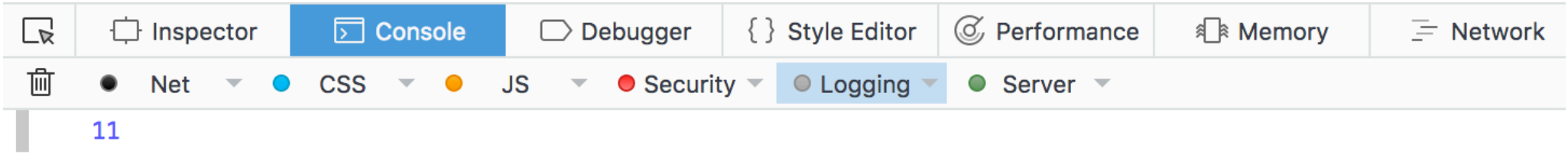
- Permet d'afficher une boîte d'alerte dans le navigateur



La méthode console.log()

- A utiliser pour corriger/débugger le code source

```
console.log(5 + 6);
```



Le concept de variable

- Variable = conteneurs pour stocker des valeurs de données qui pourront être modifiées dans le programme
- Nom de variable
  - Doit commencer par une lettre, ou “_”, ou “$”
  - Peut comporter des lettres, des chiffres et les caractères “_” et “$”
  - Pas d’espace...
  - Ne peut pas correspondre aux divers mots-clés du langage...
  - Sensible à la casse !

```
var lastname, lastName;
```

La déclaration de variables

- Déclaration explicite, en faisant précéder la variable du mot-clé `var` ou du mot-clé `let`

```
var x, y;
let a, b;
```

- Déclaration implicite, sans mot-clé

- Affectation d'une valeur
  - Signe =

```
x = 6;
```

Portée des variables

- Déclarée implicitement, une variable sera accessible de partout dans le script
  - Variable dite globale
- Déclarée avec le mot-clé `var`
  - Globale si elle est déclarée à l'extérieur d'une fonction
  - Locale si elle est déclarée au sein d'une fonction

Portée des variables : var

```
var x = 0; // Déclare x comme variable globale du fichier, on lui affecte 0

console.log(typeof z); // "undefined", car z n'existe pas encore

function a() {
  var y = 2; // Déclare y dans la portée de la fonction a
  // Affecte 2 comme valeur à y

  console.log(x, y); // 0 2

  function b() {
    x = 3; // Affecte 3 à la variable globale x
    // Ne crée pas une nouvelle variable globale
    y = 4; // Affecte 4 à la variable externe y,
    // Ne crée pas une nouvelle variable globale
    z = 5; // Crée une nouvelle variable globale
    // et lui affecte la valeur 5.
  } // (lève une ReferenceError en mode strict.)

  b(); // Crée z en tant que variable globale
  console.log(x, y, z); // 3 4 5
}

a(); // l'appel à a() entraîne un appel à b()
console.log(x, z); // 3 5
console.log(typeof y); // "undefined" car y est local à la fonction a
```

Portée des variables : `let`

- Déclarée avec le mot-clé `let`, la portée de la variable est celle du bloc courant

```
1  let x = 1;
2
3  if (x === 1) {
4    let x = 2;
5
6    console.log(x);
7    // Expected output: 2
8  }
9
10 console.log(x);
11 // Expected output: 1
12
```

Les types de données

- Pas besoin de déclarer le type des variables
- Types simples
  - Des nombres (entiers ou à virgules)
  - Booléens
  - Type *null* pour indiquer qu'il n'y a pas de données
  - Des chaînes de caractères

Les chaînes de caractères

- Suite de caractères encadrée par des quotes simples (‘) ou doubles (“)

```
var answer = "It's alright";
var answer = "He is called 'Johnny'";
var answer = 'He is called "Johnny"';
```

- Longueur d’une chaîne : propriété `length`

```
var text = ‘une chaine quelconque’;
var lg = text.length;
var lg = (‘une chaine quelconque’).length;
```

Les chaînes de caractères

- Caractère de banalisation : `\`
- Caractères spéciaux
  - `\n` : retour à la ligne
  - `\r` : appui sur la touche *ENTREE*
  - `\t` : tabulation horizontale
  - `\v` : tabulation verticale
  - `\b` : backspace

Les chaînes de caractères

- Nombreuses méthodes pour
  - Récupérer la portion d'une chaîne de caractères : `slice(start, end)`, `substring(start, end)`, `substr(start, length)`
  - Modifier une chaîne de caractères : `replace(existing_pattern, new_pattern)`
  - Conversion en majuscules/minuscules : `toUpperCase()`, `toLowerCase()`
  - Concaténation (équivalent à "+") : `concat(str1, str2)`
  - Extraction d'un caractère : `charAt(position)`, `charCodeAt(position)`
- https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/String

L'objet *String*

- Une chaîne de caractères peut être un objet String

```
var x = "John";
var y = new String("John");
// typeof x retournera string
// typeof y retournera object
```

- A ne pas pratiquer !!
- Dans l'exemple ci-dessus
  - `x == y` est `true` car x et y ont les mêmes valeurs
  - `x === y` est `false` car x et y sont de types différents
- Comparaison de deux objets toujours évaluée à `false` !!

Les tableaux

- Utilisés pour stocker plusieurs valeurs
- Accès aux valeurs par référence à l'indice (la numérotation commence à 0)
- Création d'un tableau

```
var tab1 = ["lun", "mar", “mer”]; //A privilégier
var tab2 = new Array (“lun", "mar", “mer”);
```

- Accès à un élément du tableau

```
var jour = tab1[0];
```

- Modification d'un élément du tableau

```
tab1[2] = "mercredi";
```

Les tableaux

- Tableau = objet “spécial”
  - `typeof` retourne un `object`
  - Mais accès aux éléments avec un indice
  - Alors qu’un objet utilise les noms…

```
var person = ["John", "Doe", 46]; //array
var person = {firstName:"John",
lastName:"Doe", age:46}; //object
```

- Possibilité de stocker des variables de types différents

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

Les tableaux

- Propriété `length` pour la taille du tableau

```
var person = ["John", "Doe", 46];
var lg = person.length; // longueur = 3
```

- Parcours d'un tableau

```
var fruits, text, fLen;
fruits = ["Banane", "Orange", "Pomme",
"Mangue"];
fLen = fruits.length;
text = "<ul>";
for (let i = 0; i < fLen; i++) {
    text += "<li>" + fruits[i] + "</li>";
}
text += “</ul>”;
```

Les tableaux associatifs

- JavaScript ne supporte pas les tableaux à indexes nommés
- Si utilisation d'indexes nommés, le tableau devient un objet standard
  - Propriétés et méthodes liées aux tableaux **non utilisables**

```
var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
var x = person.length; //retourne 0
var y = person[0]; //retourne undefined
```

- Pour utiliser des indexes nommés : objets !

```
var person = {"firstName":"John",
"lastName":"Doe", "age":46};
```

Les tableaux : méthodes

- Conversion d'un tableau en chaîne de caractères
  - `toString()` : valeurs séparées par une `,`
  - `join(sep)` : valeurs séparées par `sep`
- Ajout d'un élément
  - `push(elem)` ajoute en fin de tableau, retourne la longueur du tableau
  - `unshift(elem)` ajoute au début du tableau et incrémente l'index des autres éléments
- Suppression d'un élément
  - `pop()` supprime le dernier élément
  - `shift()` supprime le premier élément et décrémente l'index des autres éléments
- Voir la documentation : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array

Les fonctions

- Déclaration

```
function nomMethode(par1, par2, …) {
  //Traitement de la fonction
}

function nomFonction(par1, par2, …) {
  //Traitement de la fonction
  return (resultat);
}
```

- Appel

```
nomMethode(par1, par2, …);
resultat = nomFonction(params);
```

```
function sayHello(prenom) {
  window.alert("Bonjour " + prenom);
}

function additionner (i, j) {
  return(i + j);
}
```

```
//Affiche "Bonjour Dupont"
sayHello("Dupont");

res = additionner (2, 3);
window.alert(res);
```

Table des matières

Plusieurs types d'opérateurs

- Les opérateurs arithmétiques
- Les opérateurs d'assignation
- Les opérateurs d'incrémentation
- Les opérateurs de comparaison
- Les opérateurs logiques

Les opérateurs arithmétiques

a + b	Addition	Somme de a et b
a - b	Soustraction	Reste de la différence de b et a
a * b	Multiplication	Produit de a par b
a / b	Division	Dividende de a par b

Les opérateurs d'assignation

a += b	Additionne les valeurs de a et b et stocke le résultat dans a
a -= b	Soustrait la valeur de b à a et stocke le résultat dans a
a *= b	Multiplie les valeurs de a et b et stocke le résultat dans a
a /= b	Divise a par b et stocke le résultat dans a

Opérateurs d'incrémentation

a ++	Augmente d'une unité la valeur de a
a --	Diminue d'une unité la valeur de a

Les opérateurs de comparaison

a == b	Egal	Vrai si a est égal à b
a != b	Différent	Vrai si a est différent de b
a < b	Inférieur	Vrai si a est strictement inférieur à b
a > b	Supérieur	Vrai si a est strictement supérieur à b
a <= b	Inférieur ou égal	Vrai si a est inférieur ou égal à b
a >= b	Supérieur ou égal	Vrai si a est supérieur ou égal à b

Les opérateurs logiques

a && b	Et	Vrai si a ET b sont vrais
a || b	Ou	Vrai si a OU b sont vrais, ou les deux
!a	Négation	Vrai si a est faux

Les structures conditionnelles

- Code placé entre accolades {}
- Contrôles classiques
  - Test (*if...else if...else*, *switch...case*)
  - Boucles (*while, for*)

Condition *if... else if... else*

```
  if (expr1) {
    liste d'instructions;
  }
  else if (expr2) {
    autre liste d'instructions;
  }
  ...
  else {
    liste d'instructions par défaut;
  }
?>
```

```
var i = 100;
if ((i >= 0) && (i < 200)) {
  window.alert (i + ' est compris entre 0 et 199');
}
else if ((i >= 200) && (i < 500)) {
  window.alert (i + ' est compris entre 200 et 499');
}
else {
  window.alert (i + ' est supérieur à 499');
}
```

Condition *switch*

```
switch (var) {
  case condition1 :
    //Traitement condition1
    break;
  case condition2 :
    //Traitement condition2
    break;
  ...
  default :
    //Traitement par défaut
}
```

```
var i = 1;
switch (i) {
  case 0 :
    window.alert('La variable i vaut 0');
    break;
  case 1 :
    window.alert('La variable i vaut 1');
    break;
  default :
    window.alert('La variable i est différente de 0 et 1 ');
}
```

La boucle *for*

```
for (cond_initiale ; cond_sortie; iteration){
  instructions;
}
```

```
<html>
    <body>
        <script>
            for (let i = 0 ; i < 5; i++){
                window.alert(i);
            }
        </script>
    </body>
</html>
```

La boucle *while*

```
while (condition_vraie){
  instructions;
}
```

```
<html>
    <body>
        <script>
            var i = 0;
            while (i < 5){
                window.alert(i);
                i++;
            }
        </script>
    </body>
</html>
```

Table des matières

Javascript et le web

- JS interprété par tous les navigateurs
- Permet, entre autres, de
  - Modifier dynamiquement la présentation (le CSS) de la page HTML
  - Modifier dynamiquement la structure de la page HTML
  - Réagir aux évènements utilisateurs
  - ...

Code JS et code HTML

- Plusieurs façons d'inclure du code JS dans du code HTML
  - Avec la balise `<script>`
  - Clic sur un lien `<a href="javascript:…">ici</a>`
  - En appelant un fichier (externe) contenant le code JS
  - En réponse à un évènement HTML

Dans la balise <SCRIPT>

- Cette balise peut être insérée n'importe où
- Mais existence de bonnes pratiques
  - Un maximum de code à l'intérieur des balises `<head>` et `</head>` (pour être sûr que tout le code JS soit chargé avant que l'utilisateur n'interagisse avec le site)
  - Évènements placés n'importe où dans le corps de la page entre les balises `<body>` et `</body>`

Dans la balise <SCRIPT>

```
<html>
  <head>
    <script language="javascript">
      function fin(){
        alert('Bye');
      }
    </script>
  </head>
  <body>
    <script language="javascript">
      document.write('Pour se dire aurevoir');
    </script>
    <br /><a href="javascript:fin();">cliquez ici</a>
    <br />ou passez la souris sur
    <a href="" onMouseOver="fin();">ce lien</a>
  </body>
</html>
```

A partir d'un fichier externe

- Code JS dans un fichier annexe

```
<SCRIPT [LANGUAGE="Javascript"]
SRC="url/fichier.js"> </SCRIPT>
```

- Où `url/fichier.js` correspond au chemin d'accès au fichier contenant le code JS

A partir d'un fichier externe

```
<html>
  <head>
    <script language="javascript" src="fin.js"></script>
  </head>
  <body>
      Pour se dire aurevoir
    <br /><a href="javascript:fin();">cliquez ici</a>
    <br />ou passez la souris sur
    <a href="" onMouseOver="fin();">ce lien</a>
  </body>
</html>
```

```
function fin(){ /* fichier fin.js */
alert('Bye');
}
```

Au travers des évènements

- Évènement = action (de l'utilisateur) se produisant au sein d'une page HTML
- Exemples d'évènements
  - Clic de souris
  - Passage de la souris au-dessus d'une zone
  - Changement d'une valeur dans un champ d'un formulaire
  - Chargement de la page
- Syntaxe
  - `onEvenement = "Code_JS";`

Liste des évènements

`onAbort`	**en cas d'interruption**
`onBlur`	**en quittant**
`onChange`	**après modification réussie**
`onClick`	**en cliquant**
`onDblClick`	**en double-cliquant**
`onError`	**en cas d'erreur**
`onFocus`	**en activant**
`onKeydown`	**en appuyant sur une touche**
`onKeypress`	**en maintenant une touche appuyée**
`onKeyup`	**en relâchant la touche**
`onLoad`	**en chargeant le document**

Liste des évènements

`onMousedown`	**en maintenant la touche de souris appuyée**
`onMousemove`	**en bougeant la souris**
`onMouseout`	**en quittant l'élément avec la souris**
`onMouseover`	**en passant sur l'élément avec la souris**
`onMouseUp`	**en relâchant la touche de souris**
`onReset`	**en initialisant le formulaire**
`onSelect`	**en sélectionnant du texte**
`onSubmit`	**en envoyant le formulaire**
`onUnload`	**en quittant le fichier**
`javascript:`	**pour les liens**

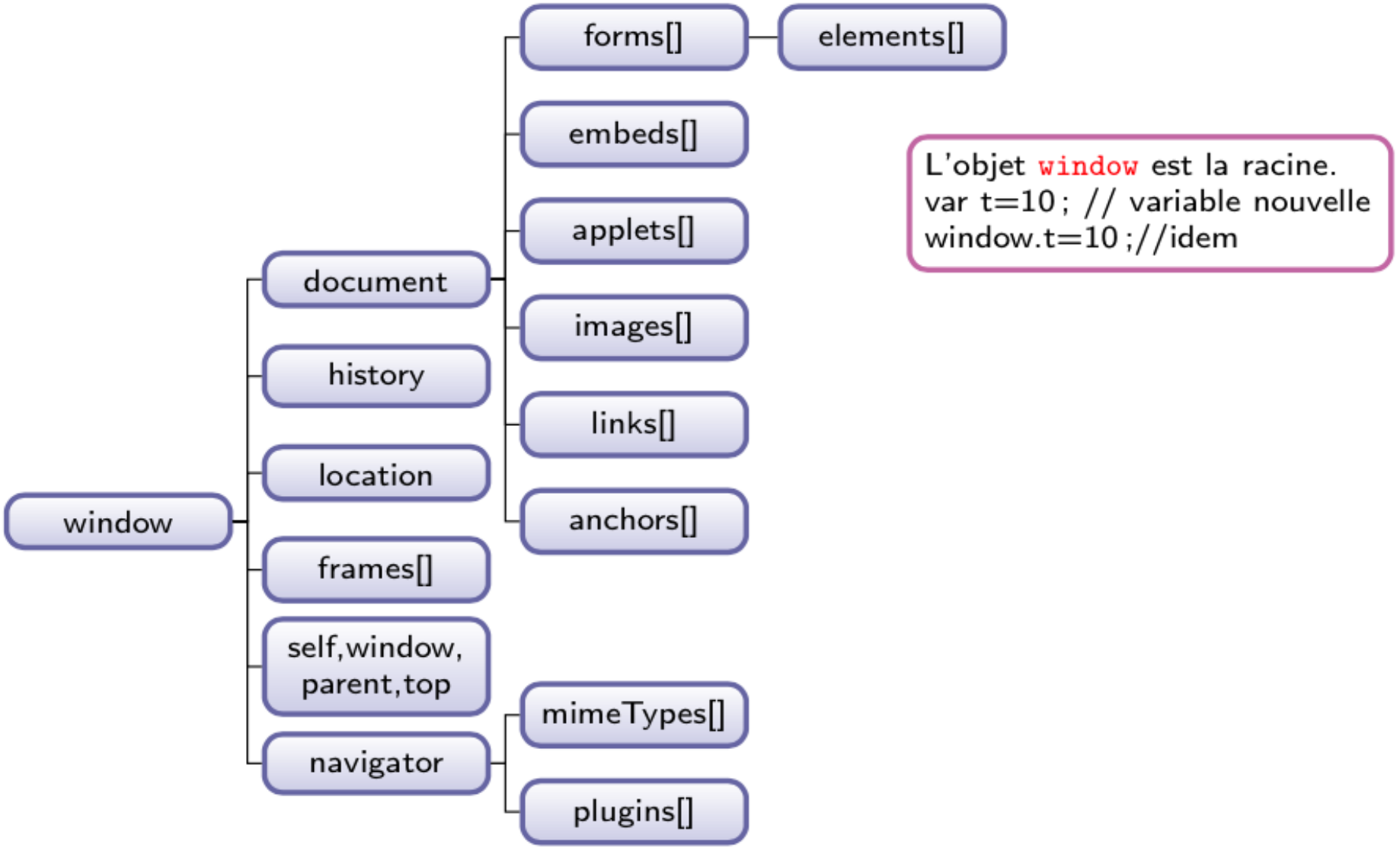
Objets associés aux évènements

- Chaque évènement ne peut pas être associé à n'importe quel objet du navigateur…
  - `onSubmit` ne peut pas s'appliquer à un lien hypertexte
  - …
- Consulter la documentation pour la sensibilité des différentes balises

Les objets du navigateur

- Différents objets prédéfinis permettent d'accéder à des informations concernant le navigateur, les documents HTML, l'écran de la machine, etc.
- Les objets de base
  - *navigator* : contient des informations sur le navigateur (une seule instance)
  - *window* : une instance par fenêtre (et frame du document HTML), accès à tous les objets créés par les balises HTML
  - *location* : informations relatives à l'adresse de la page
  - *history* : liste de liens qui ont été visités précédemment
  - *document* : propriétés sur le contenu du document (couleur de l'arrière-plan, titre, etc.)

Hiérarchie des objets



L'objet `window` est la racine.

```
var t=10; // variable nouvelle
window.t=10;//idem
```

Accès aux objets (DHTML)

- Pour accéder à un objet, il faut parcourir la hiérarchie du navigateur en partant du sommet (*window*)
- Syntaxe : `window.objet1.objet2.objet3`
- Pour modifier une propriété d'un objet
  - `objet.propriete = nouvellevaleur`
  - Modification dynamique d'une propriété donnée
- Certaines propriétés en lecture seule

L'objet *window*

- Les méthodes de *window*
  - `alert` : permet d'afficher dans une boîte de dialogue le texte passé en paramètre (simple bouton "Ok")
  - `confirm` : similaire à *alert*, mais permet un choix entre "Ok" et "Annuler"
  - `prompt` : permet de récupérer une information provenant de l'utilisateur

L'objet *window*

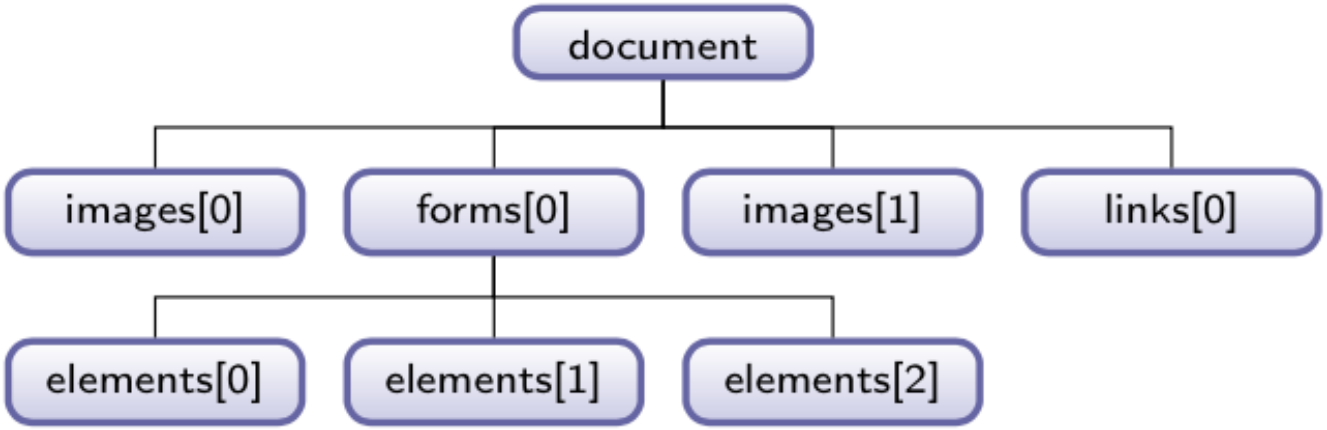
- La méthode `open`
  - Permet l'ouverture d'une nouvelle fenêtre
  - Syntaxe : `window.open("URL", "nom_fenetre", "options_fenetre");`
- La méthode `close`
  - Pour fermer une fenêtre à l'aide d'un bouton, d'un hyperlien, etc.

L'objet *document*

- Accès à tous les éléments affichés sur la page, contrôle des saisies, modification de l'apparence et du contenu
- Les propriétés
  - `location` : URL de la page
  - `title` : titre de la page
  - `fileSize` : taille de la page en octets
  - `domain` : nom de domaine de l'adresse de la page
  - `lastModified` : date de dernière modification de la page

L'objet *document*

- Un champ de saisie est contenu dans un formulaire contenu dans le document




```
<script>
  document.forms.formulaire.
      elements.adresse.value="???";
  document.forms[0].elements[0].value="???";
</script>
<form name="formulaire">
  <input type="texte" name="adresse">
</form>
```

- Accès à un élément à partir de son attribut name

Accès aux objets (DOM)

- Il est possible de récupérer n'importe quel noeud (*node*) du document HTML : représentation DOM
- Différentes méthodes
  - `getElementById()` : retourne un objet HTML à partir de son identifiant
  - `getElementsByName()` : retourne un tableau d'objets à partir de leur nom
  - `getElementsByTagName()` : retourne un tableau d'objets à partir de leur balise
  - `write()` : écrit du texte dans le document HTML

L'objet *forms*

- Permet d'accéder à tous les formulaires du document
  - `window.document.forms["nomform"];`
  - `window.document.forms[numform];`
- Les propriétés
  - `length` : nombre de formulaires dans le document
  - `name` : nom du formulaire
  - `action` : l'attribut *action* du formulaire
  - `method` : méthode de transmission des données du formulaire
- Les méthodes
  - `reset()` et `submit()`

Références

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