

1 Javascript

1.1 Syntaxe

Les espaces ne sont pas pris en compte. Les tabulations permettent d'améliorer la lisibilité, mais ne sont pas importantes. Ces deux lignes sont équivalentes. :

```
var X = 1;
var X = 1;
```

Commentaires avec `//` : *// ce que je veux ici*

1.2 Variables et types de base

Déclaration de la variable "X" : `var X = 1;`
Les variables ont un type :

```
// entier ou nombre à virgule
typeof(2) // number
```

```
// chaîne de caractères
typeof("2") // string
```

Les types sont importants et changent les calculs :

```
var foo = 1;
var bar = '2';
console.log(foo + bar); // résultat : 12 :'
```

```
// voilà comment faire correctement
var foo = 1;
// on change '2' en entier
var bar = Number('2');
console.log(foo + bar); // résultat : 3 (/o/)
```

1.3 Comparaisons et conditions

On peut tester la valeur d'une variable avec `"=="`, `"!="`, `"<"`, `">"` :

```
1 + 1 == 2; // true
2 > 3 ; // false
3.0*2 >= 6; // true
```

Une condition permet d'exécuter du code selon la valeur d'une comparaison :

```
var age = 15
if(age >= 18){
  // s'exécute
  // écrire "majeur !"
}
else if (age < 18 && age > 0){
  // ne s'exécute pas
  // écrire "mineur"
}
else {
  // s'exécute si les autres
  // comparaisons sont fausses

  // écrire "nombre positif sup !"
}
```

1.4 Boucles

Les boucles servent à effectuer une action plusieurs fois. Elles sont gouvernées par un booléen. Utiliser les boucles for quand on sait l'avance le nombre d'itérations. :

```
// on veut calculer la somme de 1 à 11
var res = 0;
for(var i=1; i < 11; i = i + 1){
  res = res + i;
} // res vaut 1 + 2 + ... + 10 = 55
```

Des fois, on ne sait pas combien d'itérations sont nécessaires pour arriver au résultat. On utilise les boucles while :

```
// on cherche ? * 7 = 500;
// sans utiliser la division
var valeur = 500; res = 0;
while(res * 7 < valeur){
  res += 1;
} // res = 72
```

Attention, les boucles peuvent ne jamais rencontrer le critères d'arrêt, on parle alors de boucles infinies...

1.5 Fonctions

Les fonctions servent à isoler des morceaux de code, il y a quelque chose en entrée, on fait quelque chose

avec. Pour appeler une fonction, il suffit d'écrire son nom et de mettre des paramètres entre par
Utilisation de fonction :

```
// écrire dans la console
console.log("quelque chose");
```

```
Math.random(); // tire un nombre au hasard
```

```
Math.floor(2.3); // renvoie la valeur tronquée (2)
```

```
alert("quelque chose"); // ouvre une popup
```

```
prompt("Nom :"); // ouvre une popup qui demande
```

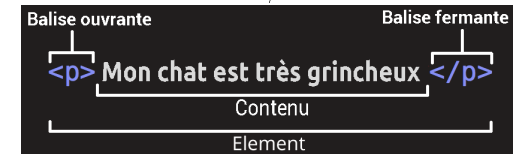
Déclaration de fonction :

```
function le_nom_que_lon_veut(param1, param2) {
  // ce que l'on veut
  return param1 + param2 // par exemple
}
```

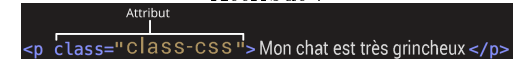
2 HTML

2.1 Vocabulaire

Élément / balise :



Attribut :



2.2 Balises

Les balises permettent de structurer le document HTML, chacune à une particularité décrite dans le standard HTML5.

Détail d'une balise :

- chevron ouvrant `"<"`, un nom, des attributs optionnels, et un chevron fermant `">"`
- balise "normales" :
`<article>...</article>`

- balise "auto fermentes" : `<input ... />`
- ajout d'une classe :
`<input class="ma_classe" />`
- ajout d'un identifiant :
`<input id="mon_id" />`

Dans notre cas, nous n'utiliserons quasiment que des balises normales. Les classes et identifiants permettent de les manipuler plus facilement.

Exemple : `<article>` ou encore :
``

Liste complète des balises HTML :

- https://www.w3schools.com/tags/ref_byfunc.asp
- <http://www.simplehtmlguide.com/cheatsheet.php>

2.3 Structure de base

Page minimale :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>Du css ici</style>
</head>
<body>
  <!-- contenu ici -->

  <script>
    console.log("Du JS ici");
  </script>
</body>
</html>
```

2.4 Séparations de sections

Titres : `<h1></h1>` , `<h2></h2>`

Bloc de contenu : `<div></div>`

Paragraphe : `<p></p>`

2.5 Interaction

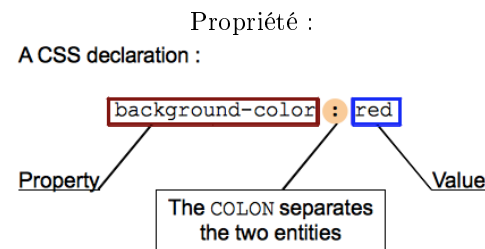
bouton :

```
<button
  onclick="fonction_JS_a_apeller()"
  type="button">
  Text
</button>
```

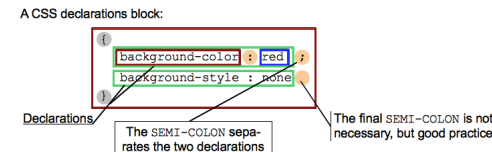
champs de texte : `<input type="text"/>`

3 CSS

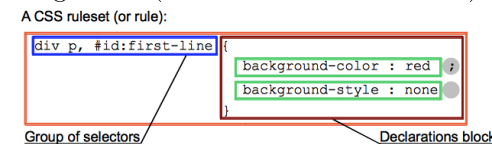
3.1 Syntaxe



Bloc de propriétés (plusieurs propriétés) :



Règle CSS (sélecteur d'élément + bloc) :



Exemple de syntaxe :

```
body {
  background-color: white;
  text-align: center;
  color: rgb(15, 15, 15);
}

h1 {
  color: rgb(55, 55, 55);
}
```

3.2 Sélecteurs

La grande puissance de CSS vient de sa façon de pouvoir sélectionner les éléments. Voilà les techniques les plus classiques.

Tous les élément d'un type :

```
/* tous les textes des divs sont rouge */
div {color: red;}
```

Tous les élément d'un type :

```
/* les textes des "divs" et
des "p" sont rouge */
div, p { color: red; }
```

Tous les élément d'une classe :

```
/* les membres de "beau" */
.beau { font-family: cursive; }
```

Selon un identifiant :

```
/* l'élément : "magnifique" */
#magnifique {
  text-transform: uppercase;
}
```

3.3 Propriétés et valeurs utiles

Pour le TP :

```
/* change la couleur de fond de la page */
background-color: white;

/* centre les textes */
text-align: center;

/* change la couleur d'un texte */
color: rgb(15, 15, 15);

/* affiche le bord de l'élément */
border: 1px solid black;

/* permet d'espacer les éléments */
margin-top: 60px;
```

3.4 Liste de propriétés CSS3

- <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>

Les images sont issues de <https://developer.mozilla.org/fr/Apprendre/>