HTML

<!DOCTYPE html> Déclaration doctype

Éléments d'en-tête

Élément	Description
<html>	Élément principale de toute page web
<head>	En-tête de la page
<title>	Titre de la page web
<style>	Définir du code CSS dans la page
<link rel="stylesheet" href="...">	Référencer du code CSS hors de la page
<script>	Définir un script JS dans la page
<script src="..." />	Référencer un script JS hors de la page

Éléments de structure

Élément	Type	Description
<body>	Block	Corps de la page
	Block	Élément générique de type block
<h1>	Block	Titre de niveau 1
<h2>	Block	Titre de niveau 2
<h3>	Block	Titre de niveau 3
<p>	Block	Paragraphe
<hr />	Block	Crée une ligne de séparation horizontale
<pre>	Block	Texte brut
	Block	Liste à puces non ordonnée
	Block	Liste à puces ordonnée
	List	Permet de créer un élément de liste
<table>	Block	Tableau.
<caption>	-	Titre du tableau
<tr>	-	Ligne de tableau
<th>	-	Cellule d'en-tête du tableau
<td>	-	Cellule du tableau
	Inline	Insère une image
<a href="">	Inline	Lien hypertexte
 	Inline	Retour à la ligne
<form method="GET/POST" action="...">	Block	Formulaire pour envoyer un fichier
<fieldset>	Block	Groupe d'un formulaire
<legend>	Inline	Titre d'un groupe
<label for="...">	Inline	Etiquette d'un champ de formulaire
<input type="...">	Inline	Champ de formulaire. Valeurs de type: Text, password, file, checkbox, radio, button, submit, hidden
<textarea>	Inline	Zone de saisie multiligne
<select>	Inline	Liste déroulante
<option value="...">		Elément d'une liste déroulante
	Inline	Élément générique de type inline
<b>	Inline	Texte en gras
<canvas>	Block	Element de dessin
<!-- -->	-	Commentaires HTML

CSS

Format des textes

color: couleur du texte
text-align: alignement du texte
font: toutes les propriétés de la police
- ordre: style weight size family
font-style: style de police (normal, italic, oblique)
font-weight: poids de police (normal, bold, lighter)
font-size: taille des caractères (pt, px, em)
font-family: famille de polices (générique/spécifique)

Propriétés d'arrière-plan

background: Toutes les propriétés du fond.
- ordre: color image repeat attachment position
background-color: Couleur du fond
background-image: Image de fond (url('/img.jpg');)
background-repeat: Répétition (no-repeat, repeat-x, repeat-y)

Bordures

border: Bordures d'un élément en une déclaration
- ordre : width style color
border-width: Epaisseur
(thin, medium, thick, px)
border-style: Style des bordures
none = pas de bordure
solid = un trait continu
dashed = traitillé
dotted = pointillé
double = double trait
border-color: Couleur de la bordure
border-top: Toutes les propriétés des bordures du haut
border-bottom: Toutes les propriétés des bordure du bas
border-right: Toutes les propriétés des bordure de droite
border-left: Toutes les propriétés des bordure de gauche

Marges

margin: Marge d'un élément en une déclaration
- ordre : top right bottom left
margin-top: Marge du haut
Margin-right: Marge de droite
Margin-bottom: Marge du bas
Margin-left: Marge de gauche

Remplissage

padding: Espace en une déclaration
- ordre : top right bottom left
padding-top: Espace du haut
padding-right: Espace de droite
padding-bottom: Espace du bas
padding-left: Espace de gauche

Dimensions

height: Hauteur d'un élément
width: Largeur d'un élément

Propriétés de position

float: Rend un objet block flottant (right, left)
flear: Cesse le flottement (right, left, both)
display: Affichage de l'élément (block, inline, none)
position: Type de positionnement
(static, relative, fixed, absolute)

Sélecteurs et combinateurs

elem {}, #id {}, .class {}, a b {}, a > b {}

Pseudo-class

a:link Lien normal, jamais visité
a:visited Lien déjà visité par l'utilisateur
a:hover Lien au moment où la souris passe dessus
a:active Lien au moment où on clique
li:nth-child(2n) Eléments de liste pairs

Spécifications des couleurs

rgb(255,255,255), #FFFFFF, white
rgb(0,0,0), #000000, black

JAVASCRIPT

Valeurs, Variables et constantes

```
3.14; 1n; true; "s"; undefined; null; Symbol(v); {}; []
var a = 1; let b = 2; const c = 3;
```

Opérateurs

```
+, -, /, *, %          Opérateurs arithmétiques
=, +=, -=, *=, /=, %=  Opérateurs d'assignement
!, &&, ||              Opérateurs logiques
>, <, ==, !=, ===      Opérateurs de comparaison
typeof                 Opérateur de type
```

Conditions et Boucles

```
if (cond1) {} else if (cond2) {} else {}
while (cond) {}
do {} while (cond);
for (let i = 0; i < 10; i++) {}
for (var prop in obj) {}
for (var entry in iter) {}
break; continue;
switch (val) { case "val": break; default: break; }
try {} catch (err) {}
throw new Error("error");
```

Chaînes de caractères

```
'a', "b"             // String
`${Math.PI}`         // Template literals
s + s                // Concaténation
s.length             // Nombre de caractères dans la chaîne
s.indexOf(r)         // Position de r dans s
s.lastIndexOf(r)     // Dernière position de r dans s
s.slice(debut)       // Tranche commençant à debut
s.slice(debut, fin)  // Tranche allant de debut à fin
                     // (fin non-compris).
s.toUpperCase()      // Mettre en majuscules
s.toLowerCase()      // Mettre en minuscules
```

Tableaux

```
var cantons = ["Uri", "Schwytz", "Nidwald", "Obwald"];
cantons[k] // Elément à l'indice k
cantons.concat(autreTableau) // Concaténation de tableaux
cantons.length // Nombre d'éléments dans le tableau
cantons.indexOf(element) // Position dans le tableau
cantons.push(element) // Ajoute un élément à la fin du tableau
```

Objets

```
var car = { make: 'Ford', model: 'Mustang', year: 1969,
            toString: function() { return this.make }};
car.make = 'Ford'; car["make"] = 'Ford';
var make = car.make; var make = car["make"]
car.toString();
```

Fonctions

```
function f(x) { return x + 1; }
var f2 = function(x) { return x + 2; }
var f3 = (x) => x + 3;
var f4 = (x) => { return x + 4; }
f.apply(this, args); f.call(this, arg, ...); f.bind(this);
```

Générateurs

```
function* gen1(i) {for (var j = 0; j < i; j++) {yield j;}}
function* gen2(i) {yield* gen1(i)}; [...gen2(10)];
```

Prototype

```
Array.prototype.toString = function() {return this.join(",");}
```

Object-oriented

```
class Parent {
  constructor(name) { this.name = name; }
  getName() { return this.name; }
}
class Child extends Parent {
  constructor(name) { super(name); }
  getName() { return super.getName() + " is a child!"; }
}
var child = new Child("John");
```

Fonctions applicables aux nombres

```
parseInt("2.54") == 2; parseFloat("2.54") == 2.54
parseInt("texte") == NaN; parseFloat("texte") == NaN
isNaN(123) == false // (NaN signifie Not-a-Number)
isNaN("123") == false
isNaN("Hello") == true
```

Fonctions mathématiques

```
Math.abs(x) // Valeur absolue
Math.random() // Nombre aléatoire
Math.pow(x, y) // Calcule x^y
Math.min(x1, x2, x3, ...)
Math.max(x1, x2, x3, ...)
```

Document Object Model and Canvas

```
document.location;
document.getElementById("id"); // Get by id
document.querySelectorAll("ul > li"); // Get by css selector
element.innerHTML // Set HTML content
element.setAttribute("id", "myid") // Set HTML attribute
element.addEventListener("click", callback);
ctx = canvas.getContext('2d');
ctx.strokeStyle = 'blue'; ctx.fillStyle = 'green';
ctx.beginPath();
ctx.lineTo(0, 0); ctx.lineTo(50, 50);
ctx.stroke();
```

Regular Expressions

```
var re = /ab+c/g;
., \s, \d              Character classes
[a-z]                  Character sets
x|y                    Either operator
*, +, ?, {n}, {n,m}    Quantifiers
^, $                   Boundaries
()                     Groups
g, i, m                Flags
```

JSON

```
JSON.parse('[1,2,"a"]'); JSON.stringify({a: 1});
```

Functional

```
var a = [1, 2, 3, 4]; f = e => e;
a.map(f); a.flatMap(f); a.forEach(f); a.join(",");
a.reduce((acc, val) => acc + val, 0);
```

Asynchronous

```
var p = new Promise(function(resolve, reject) { resolve(1) });
p.then(f); p.catch(f); async function f() { return await p; };
```

Fetch

```
fetch("http://www.exemple.com/", {method: "GET", headers: {}});
```