

UBALIA



Formation : Initiation Web – HTML, CSS, Javascript

Sommaire :

- **Introduction :** Définitions et fonctionnement
- **Protocole HTTP :** Requêtes et réponses
- **Langage HTML :** Balises, formulaires, validation de champs
- **Langage CSS :** Feuille de style, sélection d'éléments
- **Langage JavaScript :** ECMAScript, gestion du DOM et des évènements
- **Framework Bootstrap :** Installation et mise en oeuvre

Introduction



Apprendre le développement web

- | | |
|----------------------------|-------------------------------------|
| 1. Choix d'une technologie | Spring – Angular – MySQL |
| 2. Développement Frontend | HTML – CSS – Javascript – Bootstrap |
| 3. Développement Backend | Spring Framework |
| 4. Base de données | MySQL |
| 5. Contrôle de versions | Git |
| 6. Build du projet | Lancement en production |

Frontend

- Interface utilisateur
- Gestion des interactions
- Gestion des entrées utilisateur
- Travail sur le design

Backend

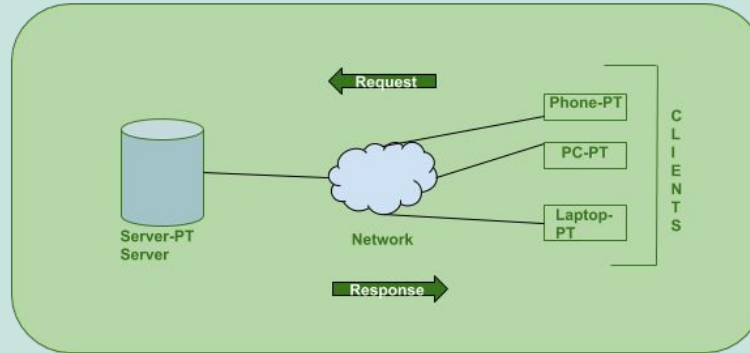
- Invisible à l'utilisateur
- Accès aux données
- Sécurité (chiffrement des données)
- Traitement des entrées utilisateur

Protocole HTTP

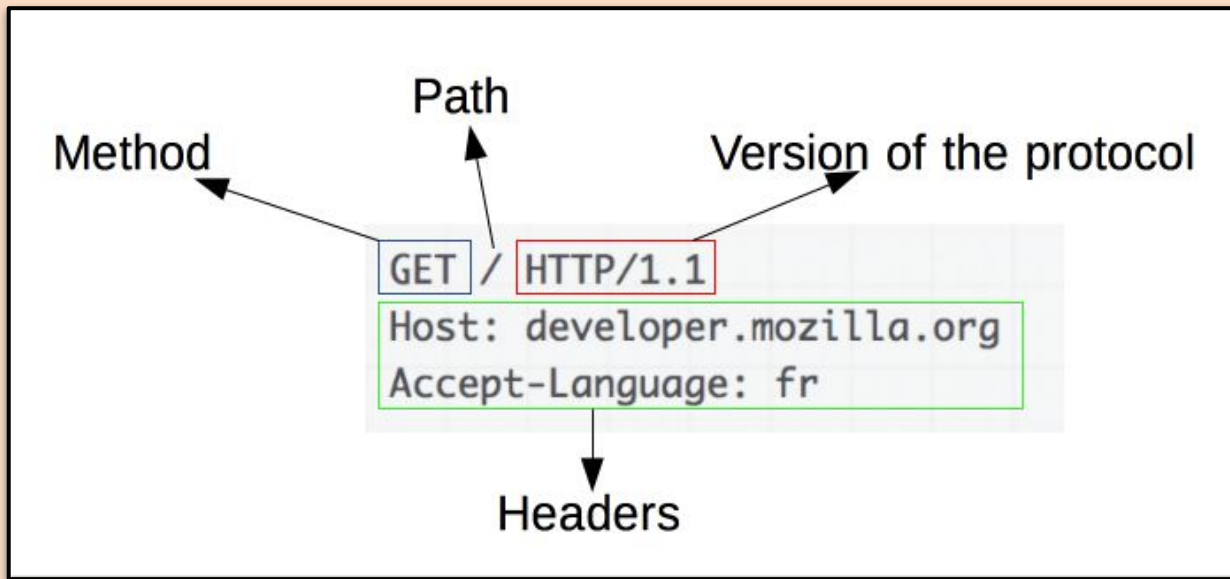


Définition

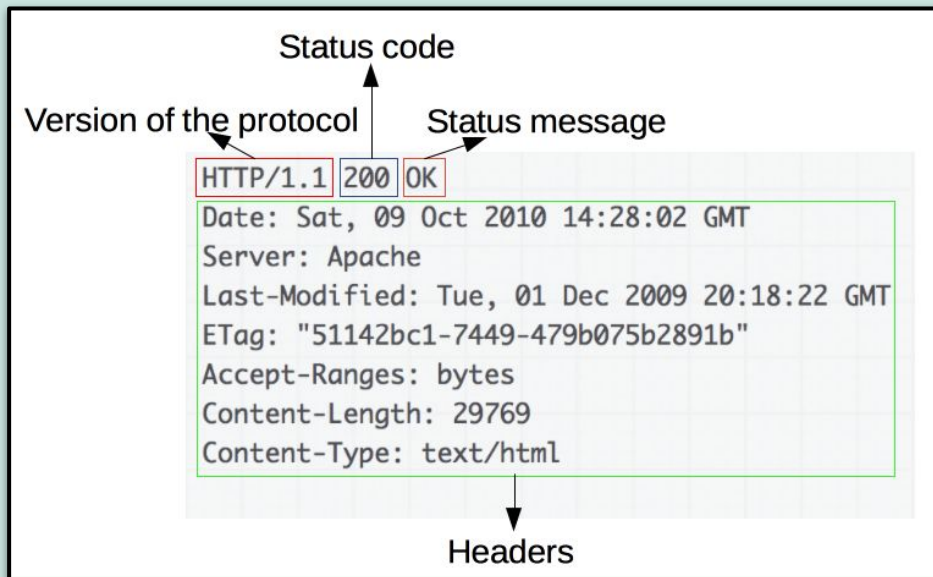
Hypertext Transfer Protocol (HTTP) : Conçu pour la communication entre un navigateur web (client) et un serveur web, ce protocole permet le transfert de documents hypermédias tel que HTML en suivant le modèle client-serveur.



Requête



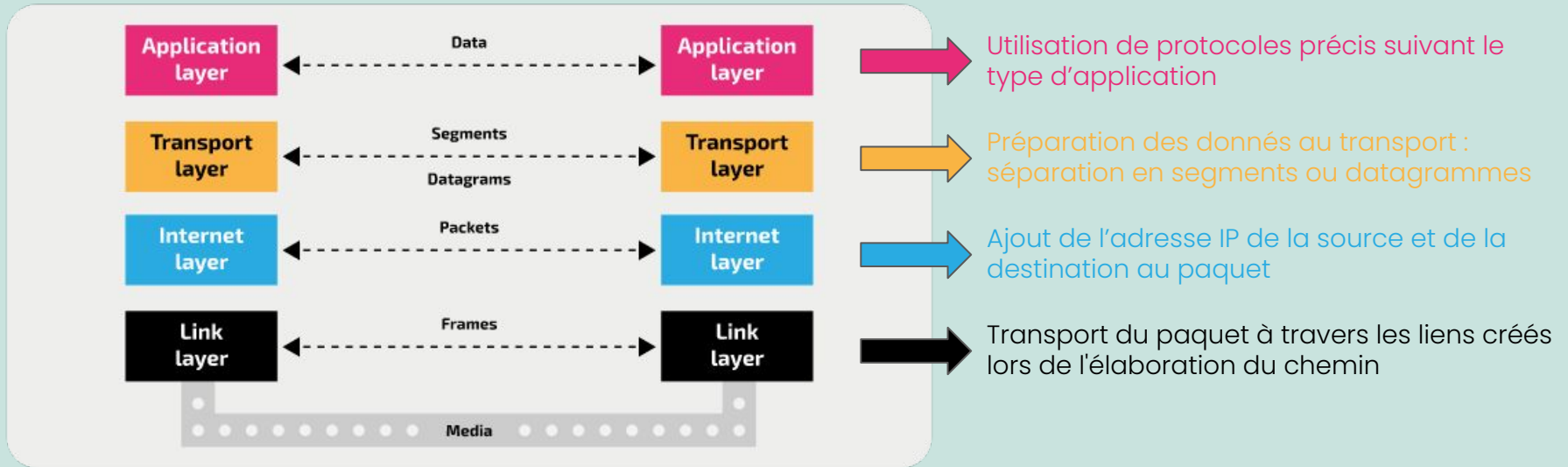
Réponse



Code de status

- Réponse informative 100 – 199
- Succès 200 – 299
- Redirection 300 – 399
- Erreur du côté client 400 – 499
- Erreur du côté serveur 500 – 599

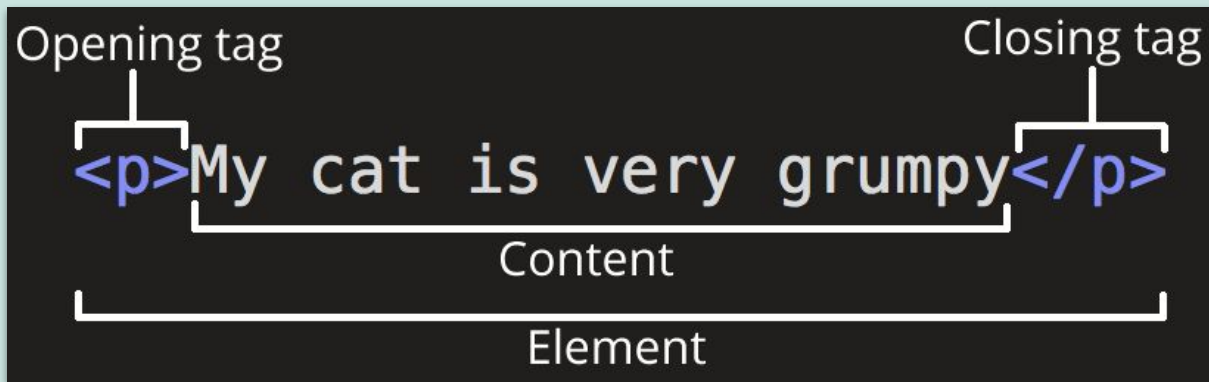
Modèle TCP/IP



Langage HTML



Composition d'un élément HTML



Balises communes

- h1 – h6 titres
- img image
- a lien
- br retour à la ligne
- p paragraphe
- hr ligne de séparation horizontale

```
<h1>titre de ma page</h1>  
<img url="path/to/image" />  
<a href="url/du/liens">lien</a>  
<br />  
<p> paragraphe ici </p>  
<hr />
```

Balises texte

- **b** gras
- **cite** citation
- **em** emphase
- **i** italique
- **s** obsolète
- **strong** gras (important)

```
<b>mis en gras</b>  
<cite>citation</cite>  
<em>emphase</em>  
<i>texte en italique</i>  
<s>texte obsolète</s>  
<strong>texte en gras</strong>
```


Composition d'un document HTML

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>My test page</title>
  </head>
  <body>
    
  </body>
</html>
```

Balises de listes

- `ol` liste ordonnée
- `ul` liste non ordonnée
- `menu` menu
- `li` item d'une liste

```
<ol>
  <li>first item</li>
  <li>second item</li>
  <li>third item</li>
</ol>
```

```
<ul>
  <li>first item</li>
  <li>second item</li>
  <li>third item</li>
</ul>
```

Balises de tableaux

- `table` tableau
- `tr` ligne
- `th` en-tête
- `td` cellule

```
<table>
  <tr>
    <th>First name</th>
    <th>Last name</th>
  </tr>
  <tr>
    <td>John</td>
    <td>Doe</td>
  </tr>
  <tr>
    <td>Jane</td>
    <td>Doe</td>
  </tr>
</table>
```

Balises de formulaire

- **form** formulaire
- **fieldset** groupe de champs
- **legend** titre d'un champs
- **label** label d'un champs
- **input** entrée d'un champs

```
<form method="post">  
  <label>Name:  
    <input name="submitted-name" autocomplete="name" />  
  </label>  
  <button>Save</button>  
</form>
```

Validation de champs

- **min** valeur minimum
- **max** valeur maximum
- **pattern** modèle à suivre
- **size** taille du champs
- **type** type du champs
- ...

```
<label for="name">Name (4 to 8 characters):</label>  
  
<input type="text" id="name" name="name" required  
      minlength="4" maxlength="8" size="10">
```

Balises sectionnantes

- **header** en-tête
- **nav** liens de navigation
- **footer** pied de page
- **section** section de page
- **article** contenu autonome
- **aside** informations complémentaires

```
<header>  
  <h1>Main Page Title</h1>  
</header>
```

```
<nav>  
  <ul>  
    <li><a href="#">Home</a></li>  
    <li><a href="#">About</a></li>  
  </ul>  
</nav>
```

[<header>](#) - [<footer>](#) - [<nav>](#) - [<section>](#) - [<article>](#) - [<aside>](#)

Balises sectionnantes

```
<article>  
  <h2>03 March 2018</h2>  
  <p>Rain.</p>  
</article>
```

```
<section>  
  <h2>Heading</h2>  
  <p>Bunch of awesome content</p>  
</section>
```

```
<aside>  
  <p>The Rough-skinned Newt defends itself with a deadly neurotoxin.</p>  
</aside>
```

```
<footer>  
  <p>© 2018 Gandalf</p>  
</footer>
```

[<header>](#) - [<footer>](#) - [<nav>](#) - [<section>](#) - [<article>](#) - [<aside>](#)

Balises génériques

- `div` block
- `span` inline

```
<p><span>Some text</span></p>
```

```
<div>  
  <p>  
    Any kind of content here. Such as &lt;p&gt;, &lt;table&gt;. You name it!  
  </p>  
</div>
```

[<div>: The Content Division element](#) – [: The Content Span element](#)

Balises audio et vidéo

```
<video src="rabbit320.webm" controls>
  <p>
    Your browser doesn't support HTML video. Here is a
    <a href="rabbit320.webm">link to the video</a> instead.
  </p>
</video>
```

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3" />
  <source src="viper.ogg" type="audio/ogg" />
  <p>
    Your browser doesn't support this audio file. Here is a
    <a href="viper.mp3">link to the audio</a> instead.
  </p>
</audio>
```

Balises *iframe*

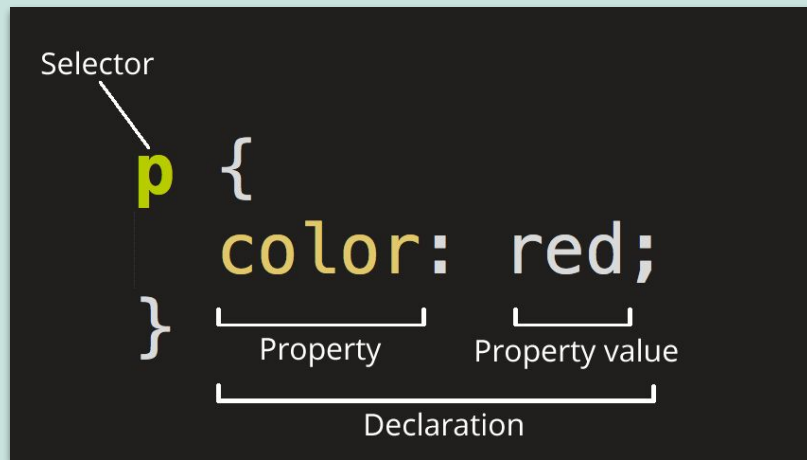
```
<iframe id="inlineFrameExample"  
  title="Inline Frame Example"  
  width="300"  
  height="200"  
  
src="https://www.openstreetmap.org/export/embed.html?bbox=-0.004017949104309083%  
2C51.47612752641776%2C0.00030577182769775396%2C51.478569861898606&layer=mapnik">  
</iframe>
```

[<iframe>: The Inline Frame element](#)

Langage CSS



Composition d'une règle CSS



Inclusion du CSS

- **interne** dans l'en-tête
- **externe** dans un autre fichier
- **inline** dans un tag HTML

Inclusion du CSS externe et interne

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet"
href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Inclusion du CSS inline

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

Sélecteurs

- **type** cible un élément HTML
- **class** cible une classe
- **id** cible un ID

```
h1 {  
}
```

```
.box {  
}
```

```
#unique {  
}
```


Cascade

- l'ordre d'apparition des règles a une importance

```
h1 {  
  color: red;  
}  
h1 {  
  color: blue;  
}
```

```
<h1>This is my heading.</h1>
```

This is my heading.

Spécificité

- la spécificité d'une sélection, mesurée suivant la précision, détermine la règle choisie

```
.main-heading {  
  color: red;  
}  
h1 {  
  color: blue;  
}
```

```
<h1 class="main-heading">This is my  
heading.</h1>
```

This is my heading.

Héritage

- une règle appliquée à un élément parent s'applique également aux éléments enfants

```
body {  
  color: blue;  
}  
span {  
  color: black;  
}
```

```
<p>As the body has been set to have a color of  
blue this is inherited through the  
descendants.</p>  
<p>We can change the color by targeting the  
element with a selector, such as this  
<span>span</span>.</p>
```

As the body has been set to have a color of blue this is inherited through the descendants.

We can change the color by targeting the element with a selector, such as this span.

Dimensionnement

- **height** hauteur d'un élément
- **width** largeur d'un élément

```
.box {  
  border: 5px solid darkblue;  
  height: 150px;  
  width: 200px;  
}
```

```
.box {  
  border: 5px solid darkblue;  
  width: 50%;  
}
```

Positionnement

- **static** placement normal
- **relative** placement normal avec un offset
- **absolute** placement relatif à son parent le plus proche
- **fixed** placement relatif au bloc englobant initial
- **sticky** placement normal avec un offset par rapport au parent le plus proche

```
.positioned {  
  position: static;  
}
```

CSS basic box model

- **Content area** Contenu “réel” de l’élément
- **Padding area** Étend la zone de contenu afin d’inclure le padding
- **Border area** Étend la zone de padding afin d’inclure une bordure
- **Margin area** Étend la zone de bordure afin d’inclure un zone de
séparation avec les éléments voisins

Margin – padding

```
/* Apply to all four sides */  
margin: 1em;  
margin: -3px;  
  
/* top and bottom | left and right */  
margin: 5% auto;  
  
/* top | left and right | bottom */  
margin: 1em auto 2em;  
  
/* top | right | bottom | left */  
margin: 2px 1em 0 auto;
```

```
/* Apply to all four sides */  
padding: 1em;  
  
/* top and bottom | left and right */  
padding: 5% 10%;  
  
/* top | left and right | bottom */  
padding: 1em 2em 2em;  
  
/* top | right | bottom | left */  
padding: 5px 1em 0 2em;
```

Langage Javascript



Historique de JavaScript




VS



<https://youtu.be/cdDPQkF7hRA>

Un langage atypique

- Ouvrez votre navigateur préféré
- Pressez la touche F12 ou +ALT+i
- Essayez les différentes opérations proposées dans la console

```
10 - '1' // 9
10 + '1' // '101'
3 > 2 // true
3 > 2 > 1 // false
[] == [] // false
0.1 + 0.2 // 0.30000000000000004
```

Insertion du script

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Learning JavaScript</title>
    <script src="myscript.js"></script>
  </head>
  <body>
    <script src="otherscript.js"></script>
  </body>
</html>
```

Syntaxe

- Présence d'accolades pour délimiter une fonction, une boucle ou une condition
- Paramètres sertis de parenthèses
- Instructions se terminant par un point-virgule

```
let name = 'MrBeast';

function sayMyName(nameToSay) {
  if(!nameToSay) {
    console.log('Name not valid');
  }
  console.log('Hello', nameToSay);
}
```

Variables

- Variables dynamiques
- Possibilité d'en changer le type
- camelcase (myVariable, morningRoutine)

```
var iAmVariableA = 'Hi';  
let aAmVariableB = 0.1;  
iAmVariableA = 2;  
iAmVariableA = false;  
iAmVariableB = iAmVariableA;
```

Types des données

```
var age = 18;           // number
var name = 'Jane';      // string
var name = {first:'Jane', last:'Doe'}; // object
var truth = false;      // boolean
var sheets = ['HTML', 'CSS', 'JS']; // array
var a; typeof a;         // undefined
var a = null;            // value null
```

Portée d'une variable

```
function areYouReady() {  
  let init = false;  
  
  if (!init) {  
    let localPhrase = "I'm ready!";  
    var globalPhrase = "I'm ready too!";  
    init = true;  
    console.log(localPhrase);  
    console.log(globalPhrase);  
  }  
  
  console.log(globalPhrase); // reachable outside the if statement  
  console.log(localPhrase); // not reachable outside the if statement  
}
```

Structures de contrôle

```
if (!bras) {  
    chocolat = false;  
} else {  
    chocolat = true;  
}
```

```
for (i = 0; i < pool.length ; i++) {  
    message = pool[i];  
}
```

```
while(pool.length) {  
    message = pool.pop();  
}
```

```
switch (choice) {  
    case 0:  
        planet = 'Mercury';  
        break;  
    case 1:  
        planet = 'Venus';  
        break;  
    case 2:  
        planet = 'Earth';  
        break;  
    case 3:  
        planet = 'Mars';  
        break;  
}
```


Déclaration d'une fonction

- Mot clé **function**
- Nom de la fonction
- Paramètres
- Corps de la fonction

```
function nomDeLaFonction(parametres) {  
    // corps de la fonction  
}
```

This

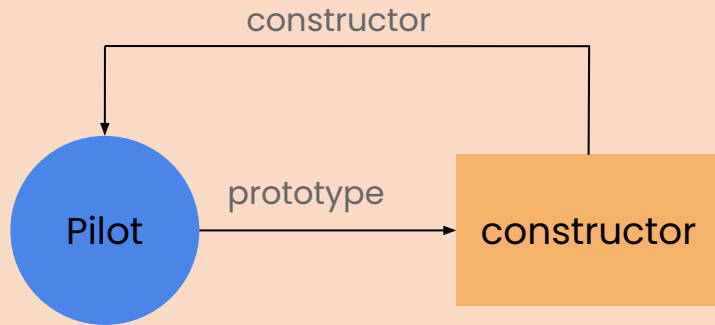
- Fait référence à l'objet appelant la fonction
- La fonction est une propriété de l'objet

```
let register = {  
  id: (new Date()).getTime(),  
  createdOn: function() {  
    return new Date(this.id).toString();  
  }  
}
```

Constructeur et prototype

```
function Pilot(name, points, standing) {  
  this.name = name;  
  this.points = points;  
  this.standing = standing;  
}
```

```
Pilot.prototype.race = function() {  
  console.log(this.name, 'is taking the lead!');  
  this.points += 25;  
}
```



Classes

```
class Pilot {  
  constructor(name, points, standing) {  
    this.name = name;  
    this.points = points;  
    this.standing = standing;  
  }  
  
  race() {  
    console.log(this.name, 'is taking the lead!');  
    this.points += 25;  
  }  
}
```

Fonctions anonymes

- Fonction ne comportant pas de nom
- Peut être affectée à une variable ou directement en paramètre d'une autre fonction

```
(function () {  
    //...  
})();
```

```
let anonymous = function() {  
    console.log('We are legion');  
};
```

```
setTimeout(function() {  
    console.log('Execute after 1 second');  
}, 1000);
```

Fonctions immédiates

- Fonction anonyme
- Exécution dès la définition de la fonction

```
(function() {  
    console.log("Les antibiotiques c'est pas automatique");  
})();
```

JavaScript Object Notation (JSON)



```
var driverLicense = {  
  class: 'S',  
  expiration: '12-14-03',  
  id: 'A1356021',  
  name: { first: 'Spongebob', last: 'Squarepants' },  
  address: {  
    street: '124 Conch street',  
    city: 'Bikini Bottom',  
    country: 'Hawai'  
  },  
  sex: 'M',  
  hair: 'yellow',  
  eyes: 'blue',  
  height: '0-04',  
  weight: '1 oz',  
  birthdate: '07-14-86'  
}
```

JavaScript Object Notation (JSON)

```
{class: 'S', expiration: '12-14-03', id: 'A1356021', name: {...}, address: {...}, ...}
```

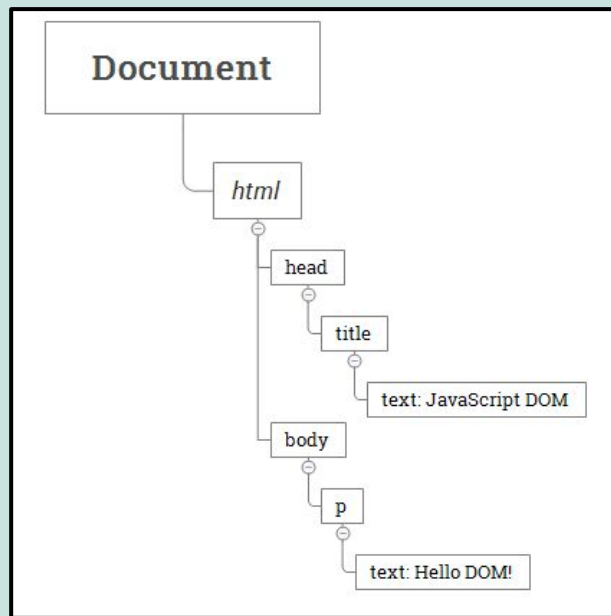
```
driverName = driverLicense.name;           // {first: 'Spongebob', last: 'Squarepants'}  
city = driverLicense.address.city;         // Bikini Bottom  
driverHairColor = driverLicense.hair;      // yellow  
driverBirthdate = driverLicense.birthdate; // 07-14-86
```


Document Object Model (DOM)

- API permettant de modifier des documents HTML
- Document HTML représenté sous forme d'arbre

Structure et exemple

```
<html>
  <head>
    <title>JavaScript DOM</title>
  </head>
  <body>
    <p>Hello DOM!</p>
  </body>
</html>
```



Récupération des éléments

```
<html>
  <head>
    <title>JavaScript DOM</title>
  </head>
  <body>
    <p>Hello DOM!</p>
    <p id="message">Hello DOM!</p>
    <p name="message">Hello DOM!</p>
    <p class="message">Hello DOM!</p>
  </body>
</html>
```

```
document.getElementsByTagName('p');
document.getElementById('message');
document.getElementsByName('message');
document.getElementsByClassName('message');
```

Création d'un élément

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JavaScript DOM</title>
</head>
<body>
  <script>
    let div = document.createElement('div');
    div.id = 'content';
    div.innerHTML = '<p>Hello DOM!</p>';
    document.body.appendChild(div);
  </script>
</body>
</html>
```

Résultat

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JavaScript DOM</title>
</head>
<body>
  <div id="content">
    <p>Hello DOM!</p>
  </div>
</body>
</html>
```

Modification d'un élément

- Modification des objets composant l'élément
- .innerText, innerHTML, textContent
- id, name, class, ...

Modification du style

- Directement depuis le JavaScript

```
element.style.color = 'red';
```

- Récupération de la classe CSS depuis l'élément

```
let classes = element.className;
```

Gestion des évènements

- Via le document HTML

```
<input type="button" value="Save" onclick="alert('Clicked!')">
```

- Via le JavaScript

```
element.addEventListener('click', function () {  
    alert('Clicked!');  
});
```

```
element.onclick = function () {  
    alert('Clicked!');  
};
```


Création d'un formulaire

```
<form action="signup.html" method="post" id="signup">
  <h1>Sign Up</h1>
  <div class="field">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter your fullname" />
    <small></small>
  </div>
  <div class="field">
    <label for="email">Email:</label>
    <input type="text" id="email" name="email" placeholder="Enter your email address" />
    <small></small>
  </div>
  <button type="submit">Subscribe</button>
</form>
```

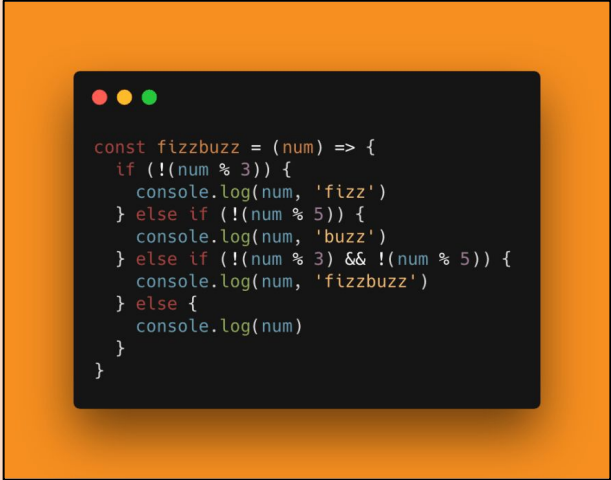
Accès au formulaire

```
const form = document.querySelector("#signup");

const NAME_REQUIRED = "Please enter your name";
const EMAIL_REQUIRED = "Please enter your email";
const EMAIL_INVALID = "Please enter a correct email address format";

form.addEventListener("submit", function(event) {
  // stop form submission
  event.preventDefault();
  // validate the form
  let nameValid = hasValue(form.elements["name"], NAME_REQUIRED);
  let emailValid = validateEmail(form.elements["email"], EMAIL_REQUIRED, EMAIL_INVALID);
  // if valid, submit the form
  if (nameValid && emailValid) {
    alert("Demo only. No form was posted.");
  }
});
```

Structuration du code



```
const fizzbuzz = (num) => {  
  if (!(num % 3)) {  
    console.log(num, 'fizz')  
  } else if (!(num % 5)) {  
    console.log(num, 'buzz')  
  } else if (!(num % 3) && !(num % 5)) {  
    console.log(num, 'fizzbuzz')  
  } else {  
    console.log(num)  
  }  
}
```

Evolution de JavaScript

ECMAScript 20**16**

ECMAScript 20**17**

ECMAScript 20**18**

https://www.w3schools.com/js/js_versions.asp
[Everything new in ECMAScript 2016, 2017, and 2018](#)
[New JavaScript Features ECMAScript 2022](#)

Framework Bootstrap



Définition

- Framework frontend
- Inclus des designs HTML, CSS et des modules JavaScript
- Aide à la réalisation de design responsive

Pourquoi l'utiliser ?

- **Facilité de mise en place**
- **Fonctionnalités responsive**
- **Approche mobile-first**
- **Compatibilité des navigateurs**

Des notions de base en HTML et CSS suffisent

Adaptation du contenu en fonction de la taille d'écran

Focus sur l'intégration mobile

Compatible avec tous les navigateurs

Notion de layout

- Mise en page
- Positionnement des éléments
- Contraintes par rapports aux éléments voisins

Manipulation des layouts

- **Breakpoints** Largeurs customizables déterminant le comportement responsive du layout
- **Containers** Conteneur espacé et aligné suivant la taille choisie
- **Grid system** Mobile-first flexbox grid contenant des classes prédéfinies
- **Columns** Modification de colonnes avec de nombreuses options clé en main
- **Gutters** Espacement entre colonnes d'une grille décidant du comportement responsive
- **Utilities** Classes utiles pour montrer, cacher, aligner et espacer les éléments
- **Z-index** Superposition et interaction entre composants
- **CSS Grid** Personnalisation de la mise en page

[Get started with Bootstrap](#)

Téléchargement

- Direct des fichiers compilés
- Direct des fichiers source
- Via jsDelivr
- Via un gestionnaire de paquets

[Download](#)

Ressources pour s'entraîner

- [Bootstrap 5 Exercises](#)
- [Bootstrap Exercises Tutorial at 4Geeks Academy](#)
- <https://getbootstrap.com/docs/5.0/examples/>
- <https://startbootstrap.com/>

UBALIA



Contact : jacques.bach@ubalia.fr



@UbaliaFR



Ubalia



Ubalia