ES6

Rappels ES6

var vs let

```
function varTest() {
  var x = 1;
  if (true) {
    var x = 2;  // c'est la même variable !
    console.log(x);  // 2
  }
  console.log(x);  // 2
}

function letTest() {
  let x = 1;
  if (true) {
    let x = 2;  // c'est une variable différente
    console.log(x);  // 2
  }
  console.log(x);  // 1
}

varTest();
letTest();
```

Pour vous exercer:
https://www.programiz.com/javascript/online-compiler/

Rappels ES6

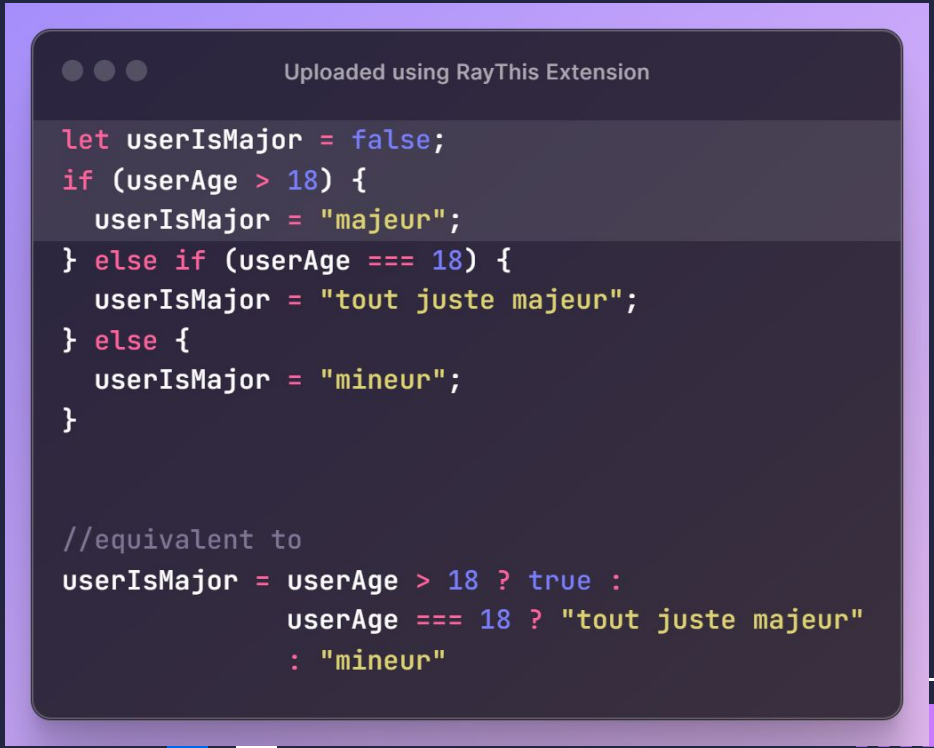
Ternaire

Uploaded using RayThis Extension

```
let userIsMajor = false;
if (userAge >= 18) {
  userIsMajor = "majeur";
} else {
  userIsMajor = "mineur";
}


//equivalent to
let userIsMajor = userAge >= 18 ? "majeur" : "mineur";
```

Rappels ES6

Chaine de ternaire



Cheat sheet fonction fléchées

Uploaded using RayThis Extension

```
const a = (param) => param; // paramètre unique, return implicite

const b = param => param; // paramètre unique (parenthèse non requise), return implicite

const c = (param1, param2) => param1 + param2; // paramètres multiples (parenthèses requises), return implicite
```

Uploaded using RayThis Extension

```
const a = () => {
  return "hello"
} // multi-lignes, return explicite

const b = () => "hello" // une seule ligne, return implicite

const c = () => (
  "hello"
)// multi ligne, return implicite
```

Rappels ES6

Cheat sheet: https://htmlcheatsheet.com/js/

fonctions flêchées

```
//from this
function foo() {
  console.log("bar");
}

//to this
const foo = () => {
  console.log("bar");
}

//or this
const foo = () => console.log("bar);
```

array functions

```
//from this
const array = [1, 2, 3];

for (let i = 0; i < array.length; i++) {
  console.log(array[i]);
}

//to this
const array = [1, 2, 3];

array.map(i => console.log(i));
```

Rappels ES6

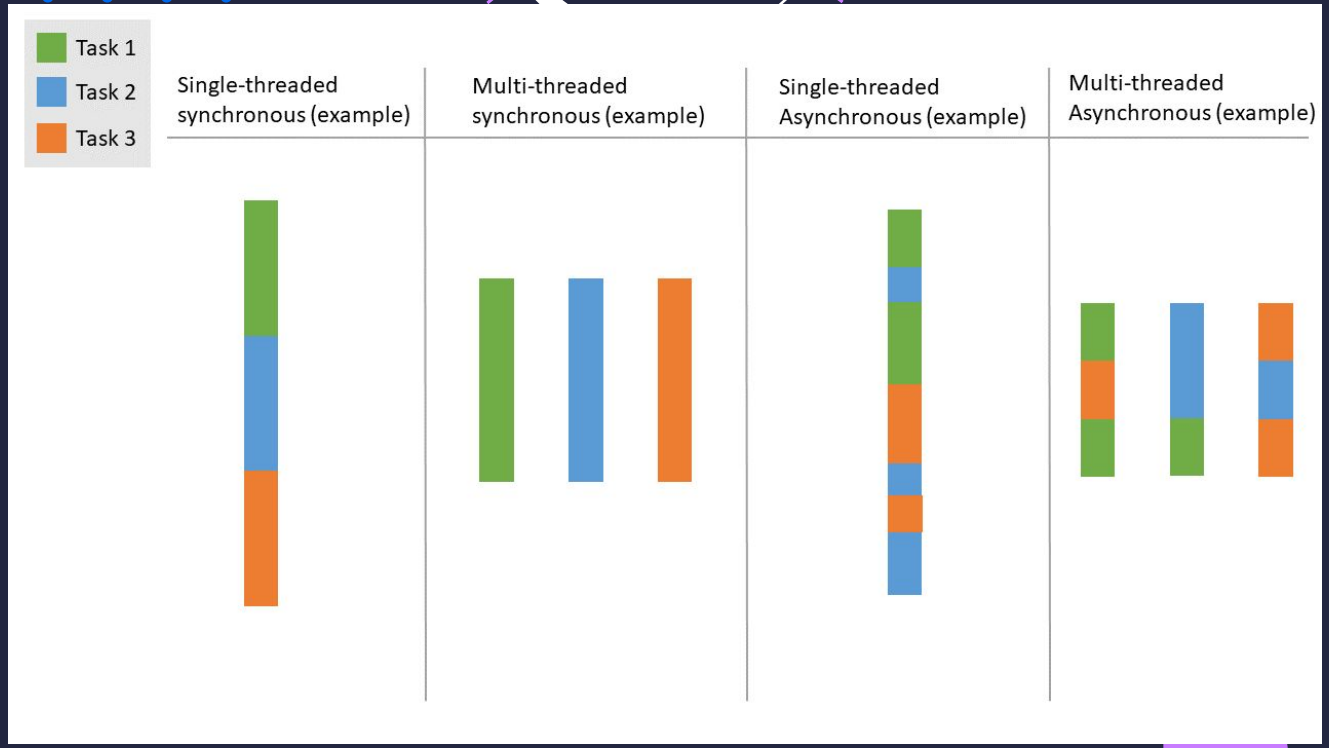
Pour vous exercer:
https://www.programiz.com/javascript/online-compiler/

structuration

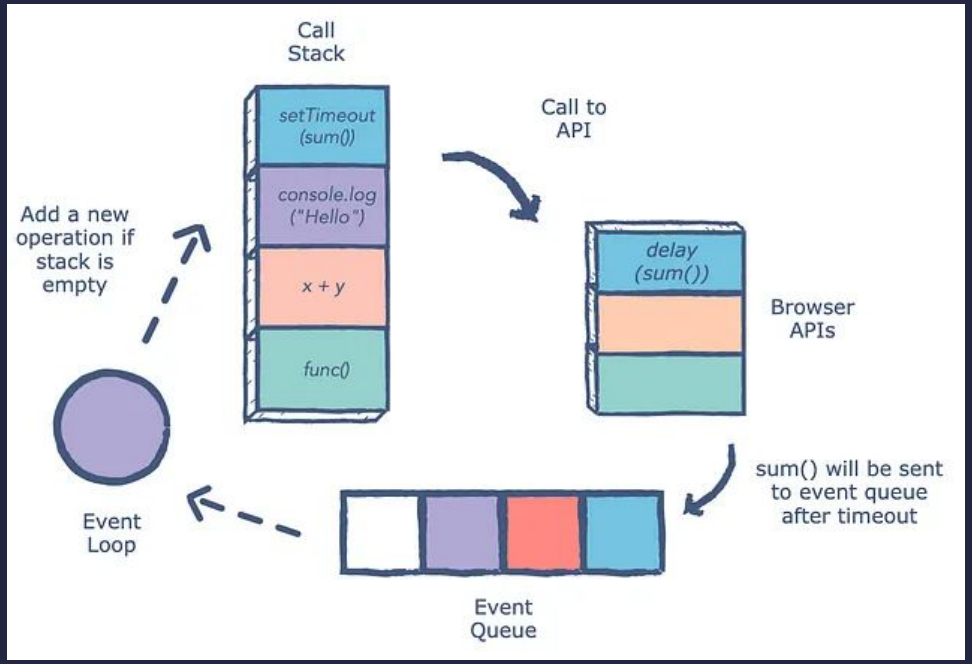
```
//array structuration
let arrayStructuration = [1, 2]
console.log(arrayStructuration); //[1, 2]

//array copy
let arrayCopy = [...arrayStructuration, 3, 4]
console.log(arrayCopy); //[1, 2, 3, 4]

//aray destructuration
let [a, b] = arrayCopy;
console.log(a);//1
console.log(b);//2
```

structuration

```
//object structuration
let obj = {
    message: "hello world"
}
console.log(obj); //{message: "hello world"}

//object add key
obj.type = "success";
console.log(obj); //{message: "hello world", type: "success"}

//object copy
let copy = {...obj}

//object destructuration
const {message} = obj;
console.log(message); //"hello world"
```


Task 1
Task 2
Task 3
Single-threaded synchronous (example)
Multi-threaded synchronous (example)
Single-threaded Asynchronous (example)
Multi-threaded Asynchronous (example)

Fonctionnement asynchrone



Rappels ES6: asynchrone

Uploaded using RayThis Extension

```
//retourne une promesse qui se résoudra après ms millisecondes
const sleep = (ms) =>  new Promise(resolve => setTimeout(resolve, ms));

const synchroneFunction = () => {
  console.log("start");
  sleep(3000);
  console.log("end");// executé immédiatement :-(
}
synchroneFunction();

const asynchroneFunction = async () => {
  console.log("start");
  await sleep(3000);
  console.log("end");// executé après 3 secondes :-D
}
asynchroneFunction();
```

Uploaded using RayThis Extension

```
//retourne une promesse qui se résoudra après ms millisecondes
const sleep = (ms) =>  new Promise(resolve => setTimeout(resolve, ms));

const synchroneFunction = () => {
  console.log("start");
  sleep(3000).then(() => {
    console.log("end");// executé après 3 secondes
  })
}
synchroneFunction();
```

Gestion d'erreur

```
function asyncFunction() {
  return new Promise((resolve, reject) => {
    // Simulation d'une opération asynchrone
    setTimeout(() => {
      const randomNumber = Math.random();

      if (randomNumber < 0.5) {
        // Résoudre la promesse avec le nombre aléatoire
        resolve(randomNumber);
      } else {
        // Rejeter la promesse avec une erreur
        reject(new Error('Une erreur s\'est produite !'));
      }
    }, 1000);
  });
}
```

```
// Utilisation de try/catch pour capturer l'erreur
async function executeAsyncFunction() {
  try {
    const result = await asyncFunction();
    console.log('Résultat :', result);
  } catch (error) {
    console.error('Erreur :', error.message);
  }
}
```

```
// Utilisation de .catch pour capturer l'erreur
asyncFunction()
  .then(result => {
    console.log('Résultat :', result);
  })
  .catch(error => {
    console.error('Erreur :', error.message);
  });
```

Design pattern asynchrone

Waterfall

```
async function asyncOperation1() { /* ... */ }
async function asyncOperation2() { /* ... */ }
async function asyncOperation3() { /* ... */ }

async function executeAsyncOperations() {
  try {
    const result1 = await asyncOperation1();
    const result2 = await asyncOperation2(result1);
    const result3 = await asyncOperation3(result2);
    console.log('Résultat final :', result3);
  } catch (error) {
    console.error('Erreur :', error);
  }
}

executeAsyncOperations()
```

Parallel

```
async function asyncOperation1() { /* ... */ }
async function asyncOperation2() { /* ... */ }
async function asyncOperation3() { /* ... */ }

async function executeParallelOperations() {
  try {
    const [result1, result2, result3] = await Promise.all([
      asyncOperation1(),
      asyncOperation2(),
      asyncOperation3()
    ]);
    console.log('Résultat 1 :', result1);
    console.log('Résultat 2 :', result2);
    console.log('Résultat 3 :', result3);
  } catch (error) {
    console.error('Erreur :', error);
  }
}

executeParallelOperations();
```

Parallel

```
const asyncQueue = [];

function addToQueue(asyncTask) {
  asyncQueue.push(asyncTask);
  if (asyncQueue.length === 1) {
    processQueue();
  }
}

function processQueue() {
  const asyncTask = asyncQueue[0];
  asyncTask()
    .then(result => {
      console.log('Résultat de la tâche :', result);
      asyncQueue.shift();
      if (asyncQueue.length > 0) {
        processQueue();
      }
    })
    .catch(error => {
      console.error('Erreur de la tâche :', error);
      asyncQueue.shift();
      if (asyncQueue.length > 0) {
        processQueue();
      }
    });
}

// Exemple d'utilisation
addToQueue(asyncOperation1);
addToQueue(asyncOperation2);
```