PAU CASESNOVES
Centre Integrat de Formació Professional

# Desenvolupament d'aplicacions web

# Desenvolupament web a l'entorn client

## UT 2.2: Mòduls.

Joan Pons Tugores

# Índex de continguts

Els programes JavaScript van començar essent força petits: la majoria del seu ús en els primers dies era per fer tasques de script aïllades, proporcionant una mica d'interactivitat a les vostres pàgines web quan fos necessari, de manera que generalment no es necessitaven scripts grans.

Uns anys avançant ràpidament i ara tenim aplicacions completes que s'executen en navegadors amb molt JavaScript, així com JavaScript que s'utilitza en altres contextos (*Node.js*, per exemple).

Per tant, en els darrers anys ha tingut sentit començar a pensar en proporcionar mecanismes per dividir els programes JavaScript en mòduls separats que es puguin importar quan sigui necessari. *Node.js* té aquesta capacitat des de fa temps i hi ha diverses biblioteques i marcs de JavaScript que permeten l'ús de mòduls.

La bona notícia és que els navegadors moderns han començat a admetre la funcionalitat dels mòduls de manera nativa, i això és el que veurem. Això només pot ser una cosa bona: els navegadors poden optimitzar la càrrega de mòduls, fent-la més eficient que haver d'utilitzar una biblioteca i fer tot aquest processament addicional del costat del client i viatges d'anada i tornada addicionals.

# Mòdul

Què és un mòdul en Javascript? Un mòdul en Javascript no és més que un fitxer js (es poden crear amg l'extensió mjs, però la majoria de servidors web encara no els tracten de la manera adequada) del que feim visibles una sèrie de variables, constants, funcions i classes de manera que la resta de codi queda ocult als altres scripts.

Un mòdul no és més que un fitxer js en el que podem incloure elements que necessitam per realitzar una tasca determinada, i que volem compartir amb altres scripts.

Per defecte el que cream dins un mòdul és privat al mòdul. Per tenir accés a aquests elements els hem d'exportar. Tot el que no exportem no serà visible fora del mòdul.

Quan volem utilitzar aquests elements dins d'un altre script els hem d'importar. No tenim perquè importar tot el que exporta el mòdul, només el que necessitem. Tot el que no importem no serà visible.

## Característiques dels mòduls

- Son fitxers js apart.
- Utilitzen el mode estricte.
- Si un fitxer *js* incorpora x mòduls no cal crear l'element *script* per a cada un d'aquests mòduls, basta crear l'element *script* per aquest fitxer *js*.

  Per exemple, el fitxer *client.js* utilitza el mòdul *funcions.js*. Al document HTML bastarà crear l'element

```
<script src="client.js"></script>
```

- Per defecte, els elements del mòdul són inaccessibles fora del mòdul.
- Per fer accessible un element fora del mòdul s'ha d'exportar.

- Per fer accessible un element exportat a un altre script s'ha d'importar.
- Els elements d'un mòdul només poden ser importats des d'un altre mòdul.
- Podem exportar constants i funcions
- Podem exportar variables, però el seu valor només es pot modificar des de dins del mòdul, no des de l'script que l'importa.
- Només els mòduls poden importar altres mòduls.
- Els navegador donen erros si no es carreguen els mòduls a través d'un servidor.
- Els mòduls només es carreguen una vegada encara que es referenciïn diverses vegades al mateix *script*. Això vol dir, per exemple, que si modificam una variable del mòdul dins d'un script el canvi es mantén als altres scripts.
- Els mòduls només són accessibles des dels scripts on s'han importat, no modifiquen l'espai global.

## Exemple de mòdul

El contingut d'un mòdul podria ser el següent:

```
//funcions.js

let x = 12;

const alfa = 90;

function suma(a, b) {
    return a + b;
}

function resta(a, b) {
    return a - b;
}

function setX(y) {
    x = y;
}

export {suma, x, setX};
```

L'únic que diferencia aquest codi d'un script qualsevol és que al final tenim una instrucció export.

Des d'altres scripts podrem utilitzar les funcions *suma* i *modifica* i podem consultar el valor de la variable *x* però no modificar-lo directament, per això hem definit la funció *setX*.

La funció *resta* i la constant *alfa* no són accessibles des d'altres *scripts*.

# Exportar elements del mòdul

Per poder utilitzar els elements definits dins d'un mòdul des d'altres scripts els hem d'exportar utilitzant la paraula reservada *export*.

La manera més senzilla és posant export abans de la definició de cada element que volguem exportar. També és la que dificulta més saber quins elements exportam, sobre tot si el mòdul és gros.

```
//funcions.js

export let x = 12;

const alfa = 90;

export function suma(a, b) {
    return a + b;
}

function resta(a, b) {
    return a - b;
}

export function modifica(y) {
    x = y;
}
```

La manera més adequada d'exportar els elements és amb una simple instrucció *export* al final del mòdul. Els elements exportats van tancats entre claus { } i separats per comes.

```
//funcions.js

let x = 12;

const alfa = 90;

function suma(a, b) {
    return a + b;
}
```

```
function resta(a, b) {
    return a - b;
}

function setX(y) {
    x = y;
}

export {suma, x, setX};
```

També es pot fer una exportació per defecte(només una per mòdul) per facilitar la tasca d'importar un element per altres scripts. Normalment s'utilitza quan només hi ha un element que es vulgui exportar:

```
export default resta;
```

Quan feim l'exportació per defecte no utilitzam claus per tancar el nom de l'element.

L'exportació per defecte es pot utilitzar amb funcions anònimes:

```
export default function (a, b) {
    return a - b;
}
```

Podem utilitzar els *exports* per reanomenar els elements que exportam amb la clàusula *as*

```
export {suma as addicio, x as y, setX as setY};
```

# Importar elements del mòdul

Per utilitzar elements d'un mòdul dins d'un script abans els hem d'importar, hem de dir al Javascript que els volem importar. Només podem importar els elements exportats i no cal importar-los tots.

## Importar elements del mòdul dins d'un altre fitxer js

En aquest cas importam els elements directament a l'script.. Al document HTML no fa falta incloure l'element *script* per afegir el mòdul.

```
import {suma, x, setX} from './funcions.js';

document.getElementById("mostrador").innerHTML="<p>"+suma(2,3)+"</p>";
// x = x * 3; //Error, no es poden modificar directament les variables del mòdul
setX(x*3); //Així podem modificar el valor de la variable.
document.getElementById("mostrador").innerHTML+="<p>"+x+"</p>";
```

Utilitzam la paraula reservada *import* seguida dels elements del mòdul que volem importar entre claus *{ }*. Llavors hem de dir de quin script els volem importar amb el *from*.

La ruta al fitxer potser absoluta, però millor empleam rutes relatives perquè són més curtes i faciliten la portabilitat.

A partir d'aquí utilitzam aquests elements ben igual que si estassin definits al mateix *script*.

La única diferència és que les variables no es poden modificar directament. Si volem canviar el valor d'una variable definida dins del mòdul ho hem de fer des d'una funció definida al mòdul.

## Importar elements del mòdul dins d'un script a l'HTML

Per incloure un mòdul dins d'un document HTML ho feim utilitzant l'element <*script*> només que amb un petit afegit

```
<script type="module" src="js/funcions.js"></script>
```

en aquest cas hem d'afegir l'atribut *type="module"*.

La paraula clau *import* només pot ser utilitzada dins mòduls, per tant si volem importar els elements del mòdul dins d'un script de la pàgina, aquest script també haurà de tenir l'atribut *type="module"*.

```
<script type="module">
    import {suma, x, setX} from './js/funcions.js';
   document.getElementById("most").innerHTML="<p>"+suma(2,3)+"</p>";
    setX(x*2);
    document.getElementById("most).innerHTML+="<p>"+x+"</p>";
</script>
```

## Importar elements exportats per defecte

Per importar un element exportat per defecte no fan falta les claus *{ }*.

```
import resta from  './funcions.js';
```

Si l'export s'ha fet amb el nom de l'element hem de posar el mateix nom a l'import. Si s'ha fet l'exportació per defecte d'un element anònim l'hi hem de posar el nom que volguem a l'export

```
import doi from  './funcions.js';
```

# Conflictes de noms

De vegades importam més d'un mòdul i hi pot haver elements exportats amb el mateix nom. O que simplement algun element del mòdul tengui el mateix nom que un element del nostre script. Podem solucionar-ho d'un parell de formes.

## Reanomenar elements a l'import

Per fer-ho utilitzam la paraula reservada *as*.

```
import {suma as addicio, x as y, setX as setY} from './funcions.js';
```

## Crear un objecte mòdul

Una altra forma d'evitar els conflictes de noms és importar tots els elements d'un mòdul dins d'un objecte

```
import * as Funcions from  './funcions.js';
```

Tots els elements exportats del mòdul passen a ser membres de l'objecte. Per accedir-hi:

```
Funcions.setX(x * 2);
```