Llenguatge de marques i sistemes de gestió de la informació

R.A 3 : Accedeix i manipula documents web utilitzant llenguatges de guions de client

Javascript - Sintaxis bàsica

JavaScript és un llenguatge de programació d'alt nivell i dinàmic, on la sintaxis és similiar a la d'altres llenguatges com C o Java. La sintaxis bàsica de JavaScript, inclou:

- Comentaris

```
// Això és un comentari d'una línia
/* Això és un comentari de múltiples línies */
```

- Instruccions (acabades sempre amb ;)

```
var nom = 'Joan';
console.log(nom);
```

- blocs de codi (delimitats entre { })

```
function saludar() {
  console.log('Hola, món');
}
```

Javascript - Variables i tipus de dades

Les variables en JavaScript es poden declarar utilitzant var, let o const.

var es fa servir per declarar variables amb un àmbit de funció o global, i permet la re-declaració dins del mateix àmbit. Les variables declarades amb **var** es "elevan" al principi del seu àmbit però s'inicialitzen amb undefined. En canvi, **let** proporciona un àmbit de bloc, evitant la re-declaració dins del mateix bloc i donant lloc a un error si es referència abans de la declaració, gràcies a la "zona morta temporal". **const** es declara per a variables que no canvien de valor, també amb un àmbit de bloc, i ha de ser inicialitzada al moment de la seva declaració. A diferència de let i var, les variables const no poden ser reassignades després de la seva inicialització.

Els tipus de dades inclouen:

- Números

```
let numero = 42;
let flotant = 3.14;
```

- Cadenes

```
let saludo = 'Hola';
```

- booleans

```
let esVeritat = true;
```

- Arrays

```
let colors = ['vermell', 'verd', 'blau'];
```

- Objectes

```
let persona = { nom: 'Joan', edat: 30 };
```

Javascript - Operadors

Javascript suporta diferents tipus d'operadors:

- Aritmètics +, -, *, /, %.

```
let suma = 5 + 3;
let producte = 4 * 2;
```

- Assignació =, +=, -=, *=, /=

```
let a = 10;
a += 5; // a es ara 15
```

- Comparació ==, ===, !=, !==, <, <=, >, >=

```
let esIgual = (5 == '5');  // true
let esEstrictamentIgual = (5 === '5');  // false
```

- Lògics &&, ||, !.

```
let resultat = (5 > 3 && 2 < 4);  // true
```

Javascript - Estructures de control

Javascript inclou estructures de control pel fluxe del programa

- Condicionals if, else if, else

```
if (edat > 18) {
  console.log('Ets adulto');
} else {
  console.log('Ets menor d'edat');
}
```

- Bucles for, while, do .. while

```
for (let i = 0; i < 10; i++) {
  console.log(i);
}
```

```
let i = 0;
while (i < 10) {
  console.log(i);
  i++;
}
```

Javascript - Funcions i events

- Les **funcions** són blocs de codi reutilitzables.

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

- Els **events** són accions que ocorren en el navegador, els quals es poden respondre amb JavaScript

```
document.getElementById('Botó').addEventListener('click', function() {
  alert('Botó clickat');
});
```

  - Events més comuns: click, mouseover, mouseout, keydown, load, resize.

Javascript - Manipulació del DOM

El **DOM (Document Object Model)** és la representació estructural del document HTML. Amb Javascript també podem interactuar i modificar el DOM per actualitzar el contingut de forma dinàmica. Amb el DOM podem:

- Seleccionar elements:

```
let element = document.getElementById('ElMeuElement');
let elements = document.getElementsByClassName('LaMevaClasse');
let seleccio = document.querySelector('.LaMevaClasse');
```

- Modificar contingut

```
element.textContent = 'Nou Contingut';
```

- Modificar estil

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

- Crear i afegir elements

```
let nouElement = document.createElement('div');
nouElement.textContent = 'Hola';
document.body.appendChild(nouElement);
```

Javascript - Exemple

Anem a veure-ho amb un exemple:

Typescript

TypeScript va ser desenvolupat per Microsoft i va aparèixer el 2012 com un superconjunt tipat de Javascript. El seu **principal objectiu** és millora la productividad del desenvolupament i la mantenibilitat del codi mitjançant la introducció del **tipus estàtic** y característiques **orientades a objectes**.

Tipatge estàtic: Permet detectar errors en temps de compilació en lloc de en temps d'execució, millorant la qualitat del codi i reduint bugs.

Millores en l'autocompletat i refactorització: Gràcies al tipatge i a les anotacions, els editors de codi poden proporcionar autocompletat més precís i eines de refactorització més robustes.

Interfícies i tipus personalitzats: Faciliten la definició d'estructures de dades complexes i milloren la claredat i la documentació del codi.

Suport per a ES6(última versió JavasCript) i més enllà: TypeScript suporta característiques modernes d'ECMAScript, cosa que permet als desenvolupadors utilitzar les últimes millores del llenguatge.

Integració amb sistemes de construcció: TypeScript s'integra bé amb eines com Webpack, Gulp i Grunt, facilitant la construcció i el desplegament d'aplicacions complexes.

Typescript - Sintaxis bàsica

- Declaració de variables

```
let nom: string = 'Joan';
let edat: number = 30;
let esActiu: boolean = true;
```

- Taules

```
let colors: string[] = ['vermell', 'verd', 'blau'];
let registre: [string, number] = ['Joan', 30];
```

- Enumeracions

```
enum Direccio {
  Nort,
  Sud,
  Est,
  Oest
}
let dir: Direccio = Direccio.Nort;
```

- Registres i tipus

```
interface Persona {
  nom: string;
  edat: number;
}
let persona: Persona = { nom: 'Joan', edad: 30 };
```

- Funcions

```
function sumar(a: number, b: number): number {
  return a + b;
}
```

si fós a?: number voldria dir que aquest valor és opcional

- Classes (orientat a objectes)

```
class Animal {
  nom: string;

  constructor(nom: string) {
    this.nom = nom;
  }
```

Com compilar Typescript per crear JavaScript

Els navegadors només poden executar **JavaScript**, per tant cal compilar el fitxer `.ts` a `.js`.

Instal·la TypeScript: Si encara no tens TypeScript instal·lat, obre un terminal o consola i executa:

```
npm install -g typescript
```

Compila el fitxer `.ts`: Navega fins a la carpeta on tens el fitxer `TypeScript.ts` i executa la comanda:

```
tsc TypeScript.ts
```

Això crearà un fitxer `TypeScript.js` al mateix directori.

Aquest `TypeScript.js` serà el que enllaçarem al nostre codi html per poder funcionar.

Dart

Dart és un llenguatge de programació desenvolupat per Google. És un llenguatge optimitzat per al desenvolupament d'aplicacions web i mòbils. La seva sintaxi és similar a la d'altres llenguatges moderns com JavaScript i TypeScript, però amb algunes diferències notables. Pretén superar algunes de les limitacions de JavaScript oferint una sintaxi neta, un rendiment superior i capacitats tant per a desenvolupament web com mòbil mitjançant **Flutter**.Amb un sol codi escrivim tant per mòbil (android, iOS), com per escriptori (windows, mac o Linux), o per aplicacions web.

Tipus de Dades

Dart és un llenguatge tipat, i suporta els següents tipus de dades bàsics:

- Nombres: `int`, `double`
- Cadenes: `String`
- Booleans: `bool`
- Llistes: `List`
- Mapes: `Map`
- Conjunts: `Set`

Dart - Sintaxis bàsica

- Declaració de variables

```dart
var nom = 'Joan'; // Tipo inferit com String
String cognom = 'Pérez'; // Tipo explícit
int edat = 30;
```

- Funcions

```dart
void saludar() {
  print('Hola, món');
}

int sumar(int a, int b) {
  return a + b;
}
```

- Estructures de control

```dart
if (edat > 18) {
  print('Ets major d'edat');
} else {
  print('Ets menor d'edat');
}

for (var i = 0; i < 5; i++) {
  print(i);
}
```

- Classes i objectes

```dart
class Persona {
  String nom;
  int edat;

  Persona(this.nom, this.edat);

  void mostrar() {
    print('Nom: $nom, Edat: $edat');
  }
}

void main() {
  var persona = Persona('Joan', 30);
  persona.mostrar();
}
```

Dart - Comparacions amb Javascript

Consistència del Llenguatge: Dart és un llenguatge coherent i complet, dissenyat des de zero per ser utilitzat tant en el front-end com en el back-end, mentre que JavaScript ha evolucionat de manera més fragmentada.

Rendiment: Dart es pot compilar a codi natiu, la qual cosa permet un rendiment superior en comparació amb JavaScript, que s'executa en un motor de JavaScript.

Eines i Desenvolupament: Dart ve amb un conjunt complet d'eines, incloent-hi un robust compilador i analitzador de codi, cosa que facilita la detecció primerenca d'errors.

Flutter: Dart és el llenguatge utilitzat per Flutter, el framework de desenvolupament d'interfícies d'usuari de Google. Flutter permet crear aplicacions natives per a iOS, Android, web i escriptori amb una única base de codi.

Suport de Tipatge: Encara que TypeScript afegeix tipatge estàtic a JavaScript, Dart va ser dissenyat amb tipatge estàtic des del començament, cosa que ofereix una experiència de desenvolupament més fluida i menys propensa a errors.

Com compilar Dart per crear JavaScript

Els navegadors només poden executar **JavaScript**, per tant cal compilar el fitxer `.dart` a `.js`.

Heu de seguir els passos d'instal·lació desde:
- Get the Dart SDK | Dart

Un cop teniu instal·lat el Dart SDK podeu compilar i crear el seu javascript seguint la següent comanda:

- dart compile js main.dart -o main.dart.js

Donada la seva complexitat i la poca relevància en el decurs de l'assignatura no el farem servir de forma pràctica.