Berger Levrault

Développement Mobile Native multiplateformes: Flutter

Abderrahmane SERIAI
Docteur en informatique, Architect logiciel & responsable R&D

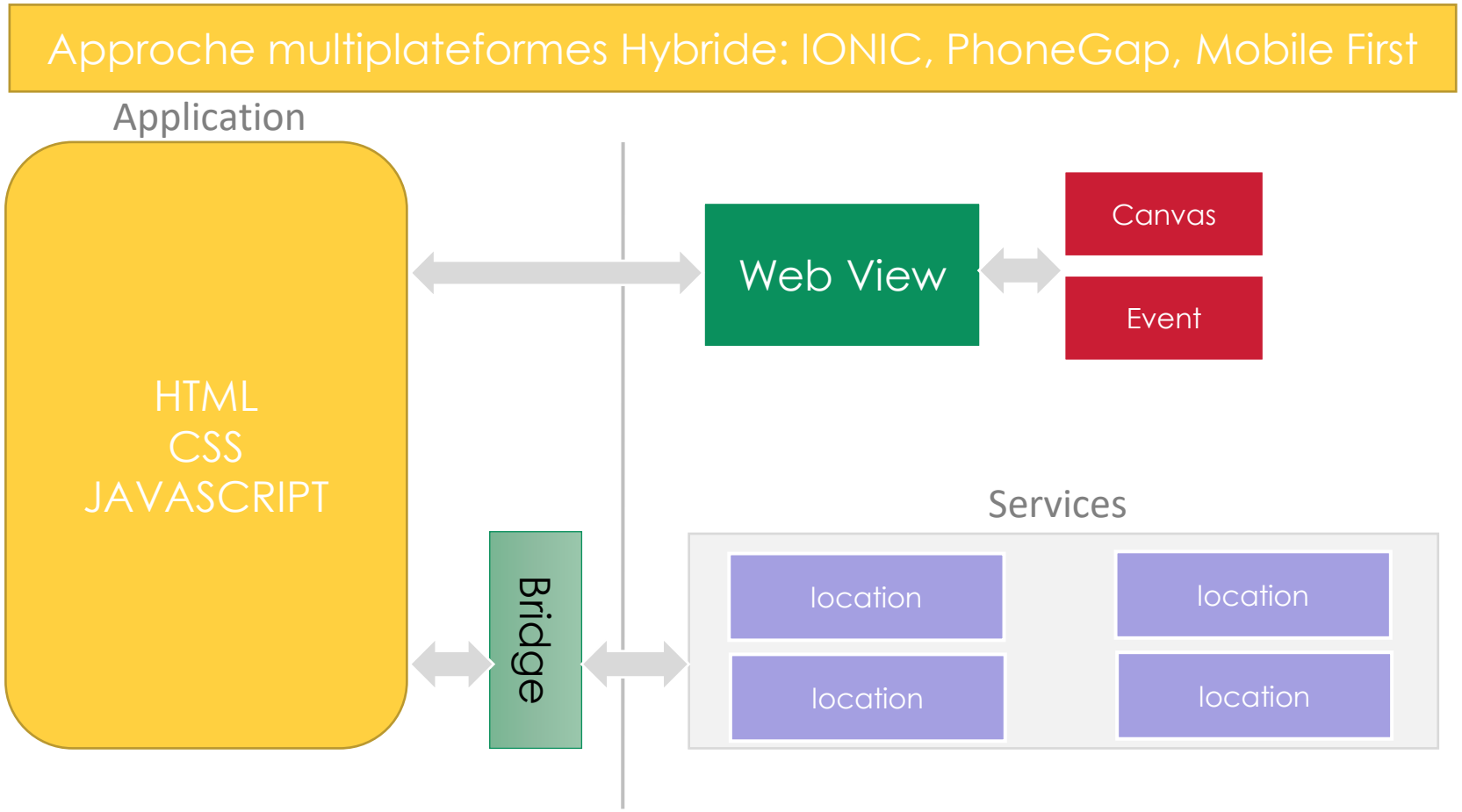
Sommaire

1. Introduction – Développement mobile
    1. Natif vs multiplateformes
    2. Multiplateformes
    3. Pour quoi Flutter
    4. Framework Flutter pour le développement mobile natif multiplateformes
    5. Framework Flutter pour le développement mobile natif multiplateformes: Dart
2. Dart – introduction au langage de programmation
    1. Déclaration de variable & Types de données
    2. Flux de contrôle, instructions If , opérateurs logiques, et boucles For
    3. Instruction switch
    4. Les fonctions
    5. Utilisation de l'opérateur « => » pour retourner des expressions
    6. Programmation orientée objet
3. Flutter – introduction au Framework
    1. Anatomie d'une application
    2. Deux types de Widget : Statless et statfull
    3. quelques Widgets prêts à être utilisés
    4. Architecture d'une application
    5. Gestion du state

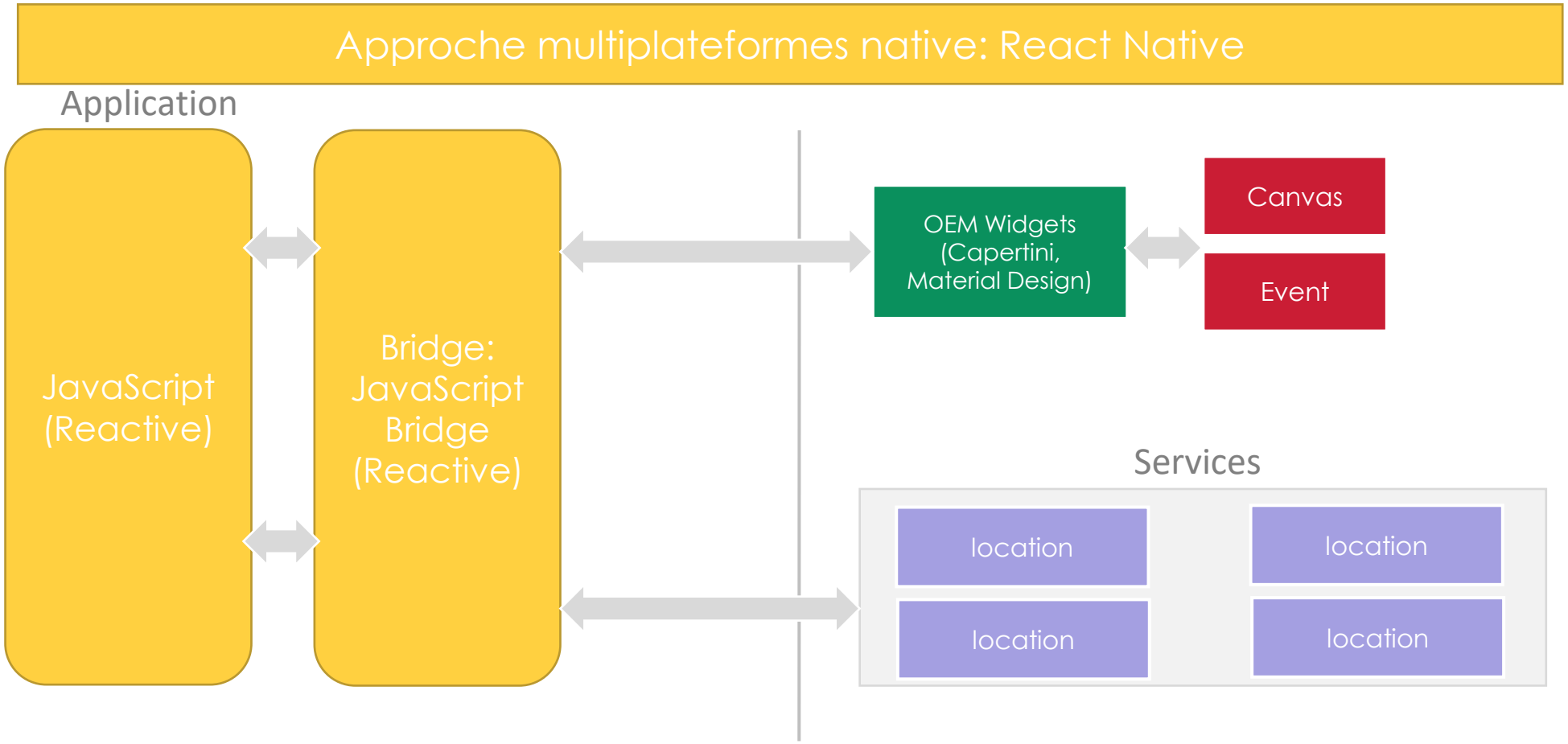
Introduction – Développement mobile : natif vs multiplateformes

1. Natif :
   - Android, java/kotlin
   - Ios, objectifc/swift, xcode
   - Etc.
2. Multiplateformes(Écrivez une fois, exécutez partout)
   - Générer plusieurs version, android, ios, windowsphone, etc.
   - Développement hybrid :
     - *framework comme ionic, cordova : html, css, javascrip, reactnatif*
     - *Flutter :*

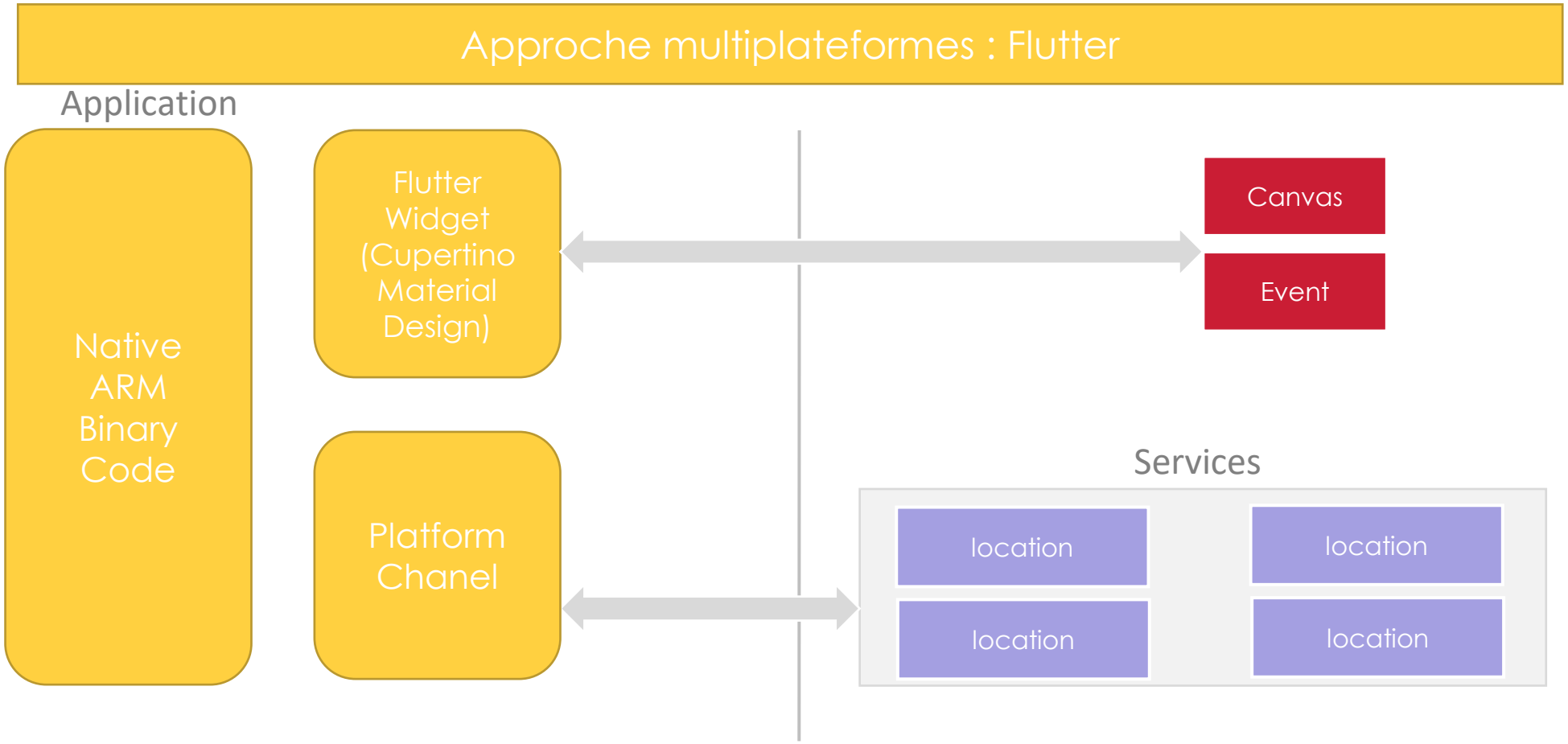
Introduction – Développement mobile : multiplateformes



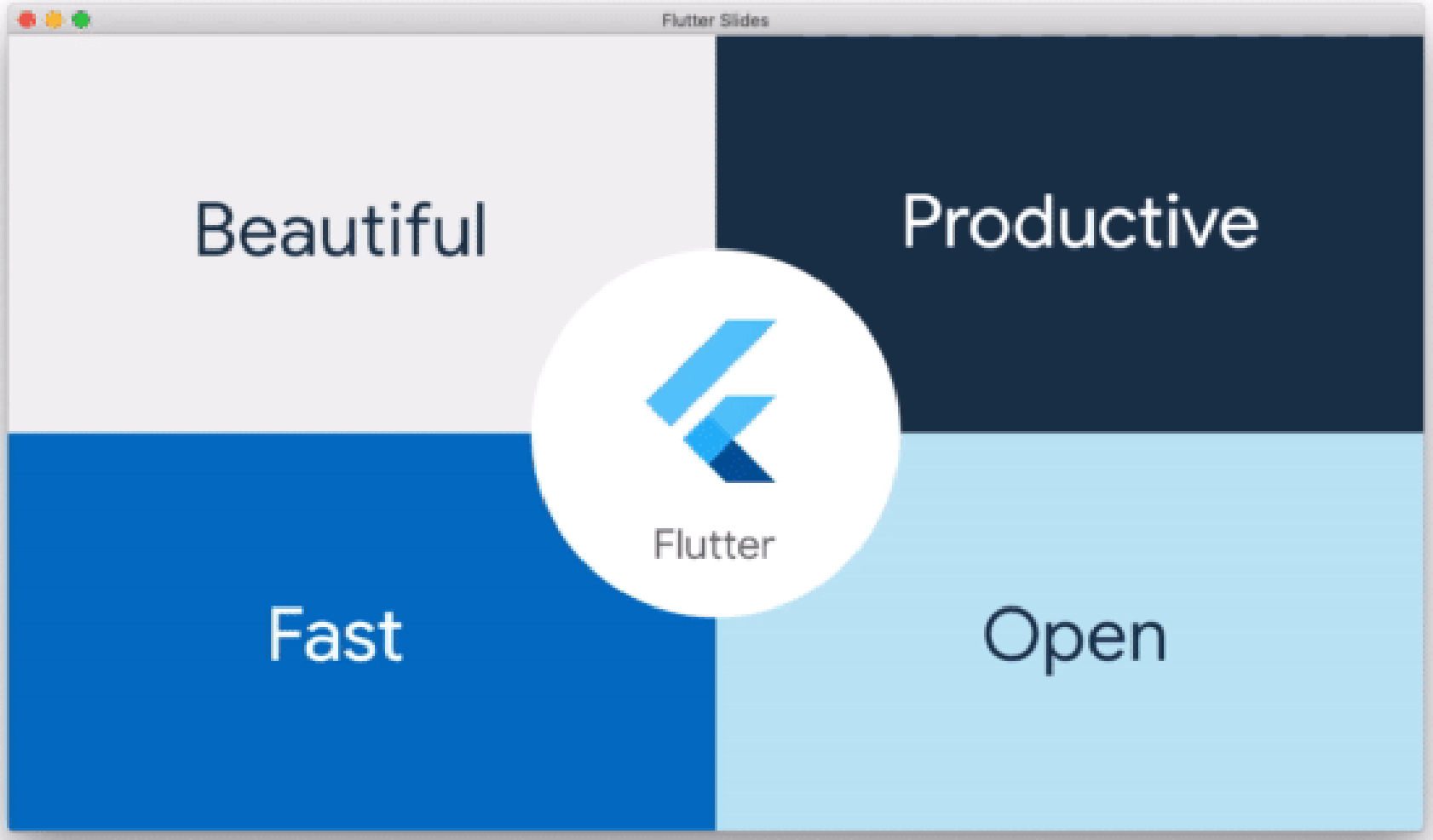
Introduction – Développement mobile : multiplateformes



Introduction – Développement mobile : multiplateformes



Introduction – Pour quoi Flutter

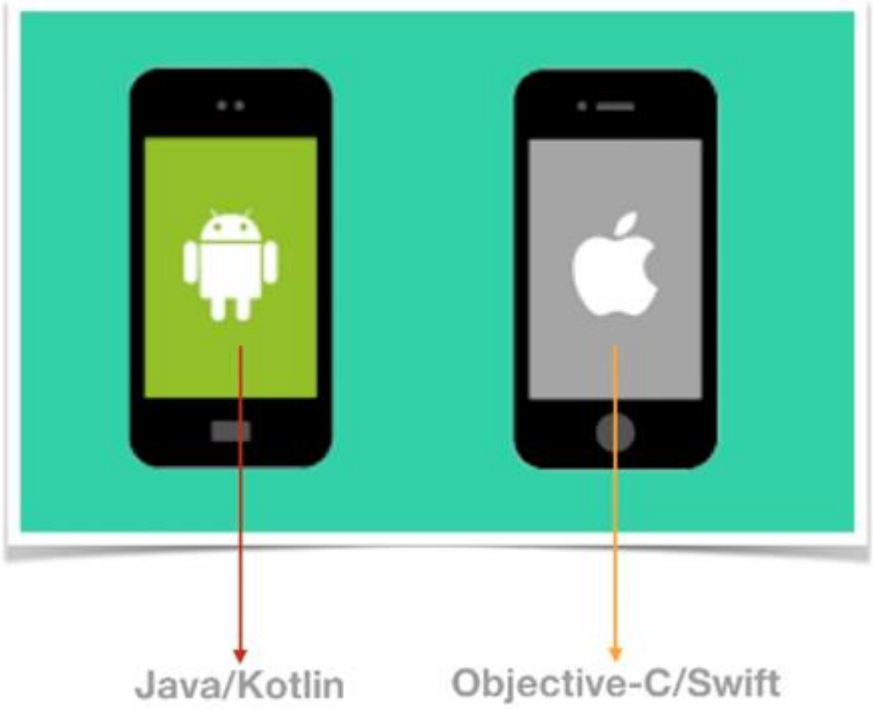


Introduction – Pour quoi Flutter

1. Éviter d'avoir à maintenir deux bases de code, et avoir un développement unique pour tous les supports choisis

2. Possibilité de développer plus rapidement et à moindre coût et faciliter de sortir l'application sur toutes les plateformes en même temps

3. Avoir le meme « look & feel », et les même widgets sur les deux plate-forme Android et IOS



Introduction – Framework Flutter pour le développement mobile natif multiplateformes

1. Framework développé par Google, qui permet de créer des applications natives android/IOS, en utilisant le langage de programmation Dart

2. La version 1.0 de Flutter a été annoncé le 4 décembre 2018.

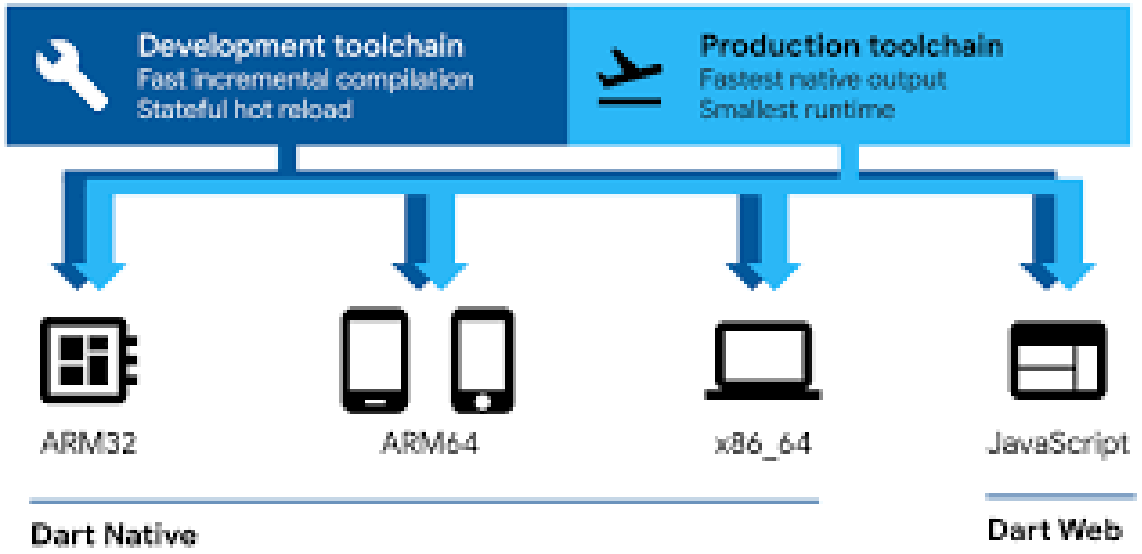
3. Dart est un langage de programmation développer par la communauté google. La première version date de 2011

4. Le but de développement de ce langage est de remplacer javascript afin d'éviter les limites de performance de ce dernier

5. Dart a trouvé sa popularité à travers Flutter qui offre une manière typiquement différente pour développer des applications mobiles multiplateformes

Introduction – Framework Flutter pour le développement mobile natif multiplateformes: Dart

- Utilisé pour écrire:
  - Des scripts simples
  - Ou des application completes
    - *Mobile (Android et IOS)*
    - *Web,*
    - *Applications coté serveur*
    - *Etc.*

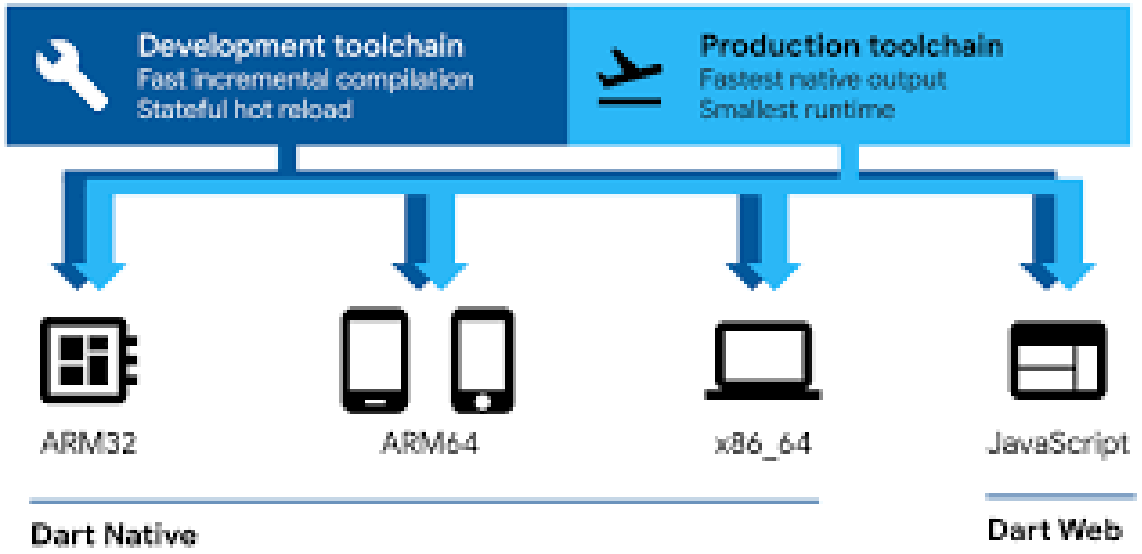


Dart Native :

- Pour les applications ciblant les appareils mobiles et de bureau
- inclut à la fois une machine virtuelle Dart avec compilation
  - **JIT**(with just-in-time)
  - **AOT(ahead-of-time) pour produire du code machine**

Introduction – Framework Flutter pour le développement mobile natif multiplateformes: Dart

- Utilisé pour écrire:
  - Des scripts simples
  - Ou des application completes
    - *Mobile (Android et IOS)*
    - *Web,*
    - *Applications coté serveur*
    - *Etc.*



Dart Web:

- Pour les applications web
- Dart Web inclut à la fois
  - Un compilateur **dartdev** (developpment time compiler) : permet d'exécuter et de débuguer les applications web dart sur le moteur chrome en utilisant l'outil webdev serve
  - Un compilateur **dart2js** (production time compiler)
- Pour faire du Dart web:
  - **Flutter Framework**
  - **AngularDart**

Dart – introduction au langage de programmation

- « dartpad.dev » pour pratiquer le langage Dart
- Dart est Syntaxiquement proche du langage Java : voici quelques notions que nous allons voir
  - Déclaration d'une variable
  - Types de données et affectation de types aux variables
  - Déclaration d'une classe
  - Constructeur d'une classe
  - var variable (variable non typée)
  - List<Person>
  - Map<String, Object>
  - Etc.

Dart – introduction au langage de programmation

Déclaration de variable & Types de données

```dart
var country;
String name;

name = "abderrahmane";

country = "France";
country = 23;
```

```dart
//Type
/*
String
Number (integer, doubles)
Boolean
lists
Maps
....
*/
```

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : numbers, integer, double

```
/*-------------------------------------*/
num age = 23;
num number = 13;
print(number);
/*-------------------------------------*/
```

```
/*-------------------------------------*/
/*
* integers are decimal without decimal point. 1,2,3,4,5, etc
* double do have  decimal point 1.1, 0.2, etc*/
int age = 13;
double number = 23.32;
print(number);
/*-------------------------------------*/
```

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : boolean

```dart
/*-----------------------------------*/
bool isTrue = true;
bool isFalse = false;
print(isTrue);
/*-----------------------------------*/
```

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : const final

```
/*-----------------------------------*/
/*
const and final keywords
const = we use this keyword when we want the value/variable to be a constant at compile-time
final - if we want the value/variable to  always be a  constant = never changes
*/
const pi = 3.14;
pi = 3;
/*-----------------------------------*/
```

```
/*-----------------------------------*/
/*
const and final keywords
const = we use this keyword when we want the value/variable to be a constant at compile-time
final - if we want the value/variable to  always be a  constant = never changes
*/
final pi = 3.14;
pi = 3;
/
```

The final variable 'pi' can only be set once. (Documentation)

Try making 'pi' non-final.

Déclaration de variable & Types de données : concaténation

```
String name = "Abderrahmane";
String lastName = "SERIAI";
int age = 34;
print("$name $lastName is $age");
print("Hello there $name ${lastName.toLowerCase()}");
```

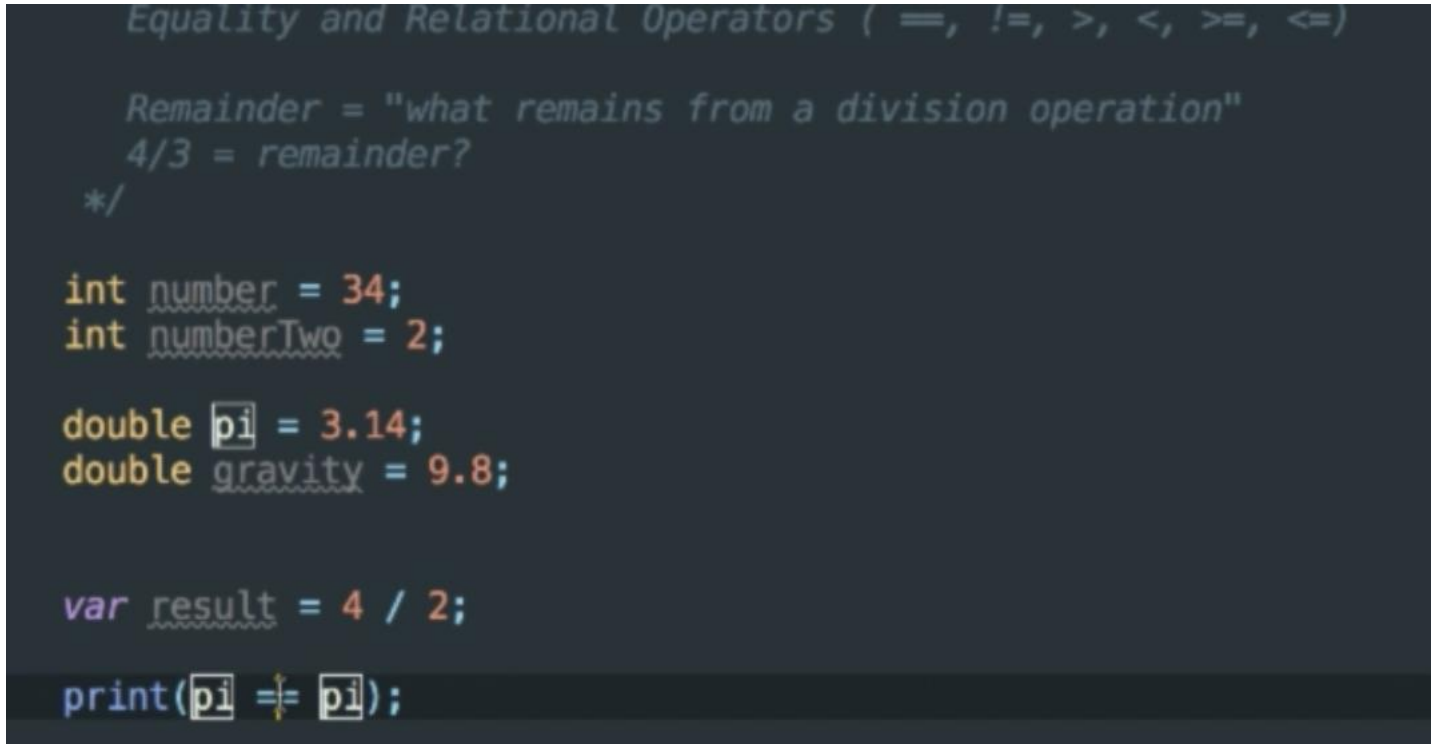
Dart – introduction au langage de programmation

Déclaration de variable & Types de données : opérateurs arithmétiques

```dart
/*-------------------------------------*/
/*
Operator = Arithmetic (-, +, *, /, % (reminder))
*/
int number = 33;
int numberTwo = 2;

double pi = 3.14;
double gravity = 9.8;

var result = number * numberTwo;

print(result);

/*-------------------------------------*/
```

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : opérateurs relationnels, égalité



Dart – introduction au langage de programmation

Déclaration de variable & Types de données : List

```
/*-------------------------------------*/

var list = [10, 2, 13, 23, 14];

print(list.length);
print(list[4]);

/*-------------------------------------*/
```

```
/*-------------------------------------*/

var list = [10, 2, 13, 23, 14];

print(list.length);
print(list[4]);

for(int i=0;i< list.length; i++){
  print("index $i contains ${list[i]}");
}

/*-------------------------------------*/
```

```
void main() {

  List<Person> listPersons = <Person>[];
  listPersons.add(Person(id: 1, name: "Abderrahmane"));
  listPersons.add(Person(id: 2, name: "Pascal"));
  for(Person p in listPersons){
    print("ID=${p.id}, Name=${p.name}");
  }
}
```

Run

Console

```
ID=1, Name=Abderrahmane
ID=2, Name=Pascal
```

```
void main() {

  var listPersons = [Person(id: 1, name: "Abderrahmane"),Person(id: 2, name: "Pascal")];

  (listPersons as List<Person>).forEach((p){
    print("name = ${p.name}");
  });
}
```

Run

Console

```
name = Abderrahmane
name = Pascal
```

```
void main() {

  List<Person> listPersons = <Person>[Person(id: 1, name: "Abderrahmane"),Person(id: 2, name: "Pascal"

  for(Person p in listPersons){
    print("ID=${p.id}, Name=${p.name}");
  }
}
```

Run

Console

```
ID=1, Name=Abderrahmane
ID=2, Name=Pascal
```

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : Map & dynamic

```dart
void main() {
  Map<String, Object> data = new Map();
  data['name'] = "Abderrahmane";
  data['age'] = 34;

  data.keys.forEach((k){
    print("$k => ${data[k]}");
  });

}
```

Run

Console

```
name => Abderrahmane
age => 34
```

```dart
// Map
var winners = {
    //key: value
  "first" : "Bill",
  "second": "George",
  "third" : "Bond"
};


print(winners["second"]);

winners.forEach((k,v) => print(k));
```

```dart
void main(){
  var data=[
    {'title':'Q1','answers':[
      {'answer':'A11','correct':true},
      {'answer':'A12','correct':false},
    ]},
     {'title':'Q2','answers':[
      {'answer':'A21','correct':false},
      {'answer':'A22','correct':true},
    ]},
  ];

  data.forEach((q){
    print(q['title']);
    (q['answers'] as List<dynamic>).forEach((a){
      print('\t ${a['answer']}');
    });
  });
}
```

RUN

Console

```
Q1
    A11
    A12
Q2
    A21
    A22
```

Documentation

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : encode & decode

```dart
import 'dart:convert';

void main() {

  var data = '[{"id":1, "name":"Abderrahmane"},{"id":2, "name":"Pascal"}]';
  print(data);
  var parsedData = json.decode(data);
  print(parsedData);
  for(var p in parsedData){
    print(p['name']);
  }

}
```

Run

Console

```
[{"id":1, "name":"Abderrahmane"},{"id":2, "name":"Pascal"}]
[{id: 1, name: Abderrahmane}, {id: 2, name: Pascal}]
Abderrahmane
Pascal
```

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : encode & decode

```dart
import 'dart:convert';

void main() {

  var data = '[{"id":1, "name":"Abderrahmane"},{"id":2, "name":"Pascal"}]';
  print(data);
  var parsedData = json.decode(data);
  print(parsedData);
  (parsedData as List<dynamic>).forEach((d){
    print(d['id']);
  });

}
```

Run

Console

```
[{"id":1, "name":"Abderrahmane"},{"id":2, "name":"Pascal"}]
[{id: 1, name: Abderrahmane}, {id: 2, name: Pascal}]
1
2
```

Dart – introduction au langage de programmation

Déclaration de variable & Types de données : encode & decode

```dart
import 'dart:convert';

void main() {

  var data = '[{"id":1, "name":"Abderrahmane"},{"id":2, "name":"Pascal"}]';
  print(data);
  var parsedData = json.decode(data);
  print(parsedData);
  (parsedData as List<dynamic>).forEach((d){
    print(d['id']);
  });

  var person = json.encode(parsedData[0]);
  print(person);
}
```

Run

Console

```
[{"id":1, "name":"Abderrahmane"},{"id":2, "name":"Pascal"}]
[{id: 1, name: Abderrahmane}, {id: 2, name: Pascal}]
1
2
{"id":1,"name":"Abderrahmane"}
```

Dart – introduction au langage de programmation

Flux de contrôle, instructions If , opérateurs logiques, et boucles For

As, is, is!

```
//for loop
for(var i = 0; i < 10; i++) {
  print("Hello $i");

}
```

```
while(true) { //infinite loop
  if(number >= 34)  print("Going..."); break;

  print("I am back!");
}
```

Dead code.

```
//If statement
if(number != 34) {
   print("If true, this will run!");

}else {
  print("Else running!");
}
```

```
do {
  print("Hello World");

}while( number < 34);
```

```
Logical Operators ( !, ||, &&)
```

```
if( !(number != 100) && number <= 100 ) {

  print("Not a hundred!");
}else {
  print("Yess!");

}
```

```
if( !(number != 100) || number <= 100 ) {

  print("Not a hundred!");
}else {
  print("Yess!");

}
```

Dart – introduction au langage de programmation

Instructions switch

Dart – introduction au langage de programmation

Les fonctions

```
doSomething() {
  print("Hello Functions!");
}
```

```
void doSomething() {

  print("Hello Functions!");
  sayMyName();
}
```

```
main(List<String> arguments) {
  print('Hello world: ${IntroToFunction.calculate()}!');

}
```

```
String showName() {

  return "Hello From ShowName!";

}
```

```
String sayHello(String name) {

  return "Hello $name";

}
```

```
String sayHello(String name, String lastName, [int age]) => "Hello $name "
    "$lastName $age years old.";
```

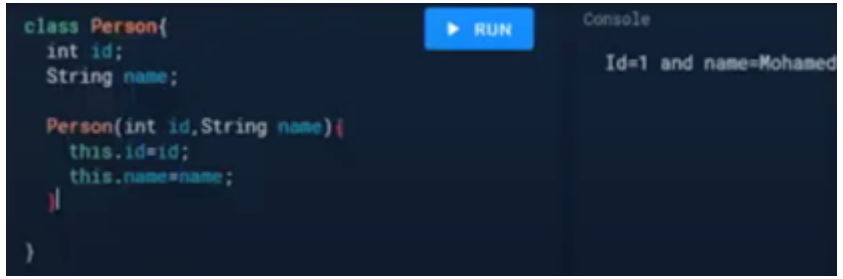
```
String yourAge(String name, String lastName, [int age]) {

  var finalResult = "$name $lastName";

  if (age == null){
     finalResult = "$finalResult doens't want to tell their age!";
  }
  return finalResult;
}
```

Dart – introduction au langage de programmation

Utilisation de l'opérateur « => »pour retourner des expressions

```
String getName() => "James Bond";
```

Constructeur d'une classe



Les setters et getters

```dart
class Microphone{

  //Instance variables, member variables
  //this = this object/class
   String name;
   String color;
   int model = 4536;


   //Syntactic sugar constructor
   Microphone(this.name, this.color, this.model);

// Named Constructor
   Microphone.initialize() {
      name = "Blue Yeti 2nd Edition";
      model = 67;



   }


   String get getName => name; // getter
   set setName(String value) => name = value;
```

Concepts d'héritage

```dart
class Person{
  String name, lastName, nationality;
  int age;

  void showName() {
    print(this.name);

  }


}


class Bonni extends Person {

}




main(List<String> arguments) {

  var bonni = new Bonni();
  bonni.name = "Bonni";
  bonni.showName();

}
```

```dart
class Location {
  num lat, lng; // instance variables or member fields

  Location(this.lat, this.lng);
}

class ElevatedLocation extends Location {
  ElevatedLocation(num lat, num lng) : super(lat, lng);

}
```

```dart
lass Location {
 num lat, lng; // instance variables or member fields

 Location(this.lat, this.lng);

 //named constructor
 Location.create(this.lat, this.lng);


lass ElevatedLocation extends Location {
 num elevation;
 ElevatedLocation(num lat, num lng, this.elevation) : super(lat, lng); //
```

```dart
  Location(this.lat, this.lng);

  //named constructor
  Location.create(this.lat, this.lng);
}

class ElevatedLocation extends Location {
  num elevation;
  ElevatedLocation(num lat, num lng, this.elevation) : super.create(lat, lng);

}
```

interface, implémentation

```dart
abstract class Service{
  double add(double a,double b);
  double sub(double a,double b);
}
class ServiceImpl extends Service{
  @override
  double add(double a, double b) {
    // TODO: implement add
    return a+b;
  }

  @override
  double sub(double a, double b) {
    // TODO: implement sub
    return a-b;
  }

}
void main(){
  Service s1=new ServiceImpl();
  print(s1.add(6,8));
  print (s1.sub(9,3));
}
```

```dart
class A{

}
abstract class IService{
  double add(double a,double b);
  double sub(double a,double b);
}

class ServiceImpl extends A implements IService{
  @override
  double add(double a, double b) {
    // TODO: implement add
    return a+b;
  }

  @override
  double sub(double a, double b) {
    // TODO: implement sub
    return a-b;
  }
  @override
  double div(double a,double b){
    return 0;
  }
```