Introduction to JavaScript

JavaScript defines **6 types**

```
var aNumber = 3.12;
var aBoolean = true;
var aString = "HEIG-VD";
var anObject = {
  aProperty: null
};

// t is true for all of these:
var t;
t = typeof aNumber === "number";
t = typeof aBoolean === "boolean";
t = typeof aString === "string";
t = typeof anObject === "object";
t = typeof anObject.aProperty === "object";
t = typeof anObject.foo === "undefined";
```

The 6 types are:

- number
- boolean
- string
- object
- undefined
- null

null is a type, but **typeof null === "object"**

JavaScript is a **dynamic** language

```
var myVariable = "aString";
typeof myVariable; // "string"

myVariable = 3.12;
typeof myVariable; // "number"

myVariable = true;
typeof myVariable; // "boolean"

myVariable = {
  aProperty: "aValue"
};
typeof myVariable; // "object"
```

- When you declare a **variable**, you don't specify a type.
- The type can **change** over time.

There are **2 scopes** for variables: the (evil) global scope and the function scope

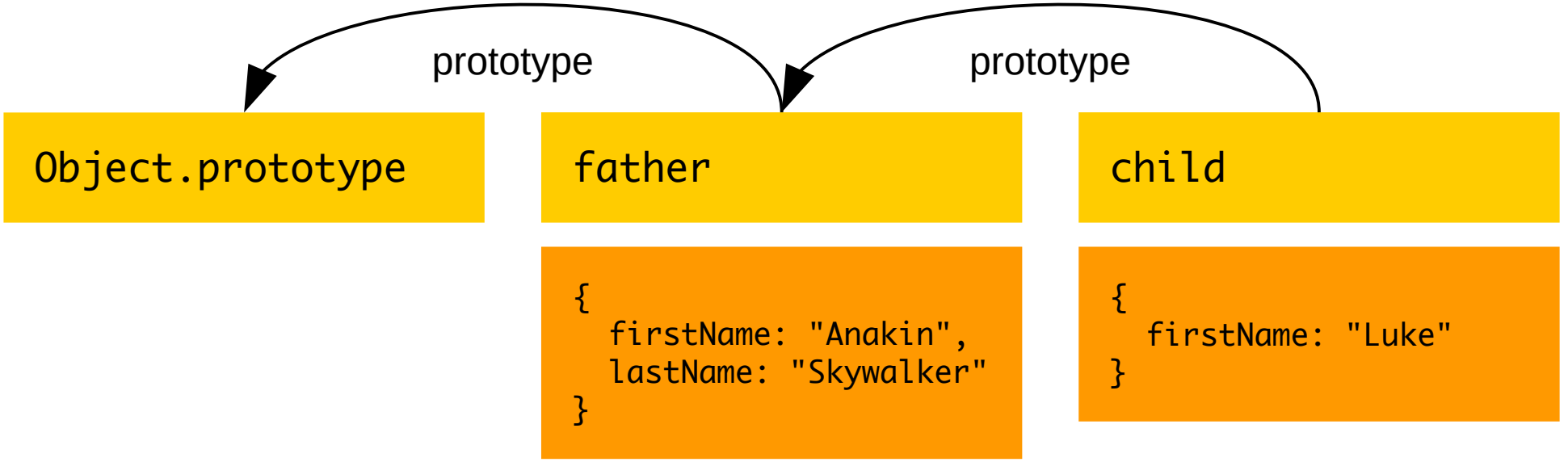
```
var aVariableInGlobalScope;

function myFunction() {
  var  aVariableInFunctionScope;
  anotherVariableInGlobalScope;
}

function myFunction2() {
  for (i = 0; i < 10; i++) {
    // i is in global scope!
  }
  for (var j = 0; j < 10; j++) {
    // j is in function scope!
  }
}
```

- A variable declared within a function is **not accessible** outside this function.
- Unless using **strict mode**, it is not mandatory to declare variables (beware of typos…)
- Two scripts loaded from the same HTML page share the same global scope (beware of **conflicts**…).
- There is no **block scope**.

Every object **inherits** from a prototype object




```
// Prints "Skywalker"
console.log(child.lastName);
```

- Every object inherits from a prototype object. It **inherits and can override** its properties, including its methods.
- Objects created with object literals inherit from **Object.prototype**.
- When you access the property of an object, JavaScript **looks up the prototype chain** until it finds an ancestor that has a value for this property.