OBJECTS AND ARRAYS

AGENDA

- Array
- Object
  - Primitives vs Non-primitives
  - Properties
  - Methods
- Mutability
- References

PRIMITIVES

- Vi har lärt oss dom primitiva data typerna

1. **Number:** Floating point numbers 👉 Used for decimals and integers `let age = 23;`
2. **String:** Sequence of characters 👉 Used for text `let firstName = 'Jonas';`
3. **Boolean:** Logical type that can only be `true` or `false` 👉 Used for taking decisions `let fullAge = true;`
4. **Undefined:** Value taken by a variable that is not yet defined ('empty value') `let children;`
5. **Null:** Also means 'empty value'
6. **Symbol (ES2015):** Value that is unique and cannot be changed *[Not useful for now]*
7. **BigInt (ES2020):** Larger integers than the Number type can hold

- Men dessa data typer räcker bara så långt...

ARRAY

- Ibland vill vi kunna hantera flera av något (dataset), för detta har vi array [...]

```
let listOfNumbers = [2, 3, 5, 7, 11];
console.log(listOfNumbers[2]);
// → 5
console.log(listOfNumbers[0]);
// → 2
console.log(listOfNumbers[2 - 1]);
// → 3
```

- Vi har **index** (position) och **value** I en array

ARRAY

- Ta ut från index och ändra i index för samtliga
  - Skapa array ”list” [10, 12, 9, 8, 7]
  - Skapa array ”wins” [true, false, true, true, false, true]
  - Skapa array ”heros” [”hulk”, ”superman”, ”spiderman”, ”ironman”]
  - Kolla längd på alla

ARRAY

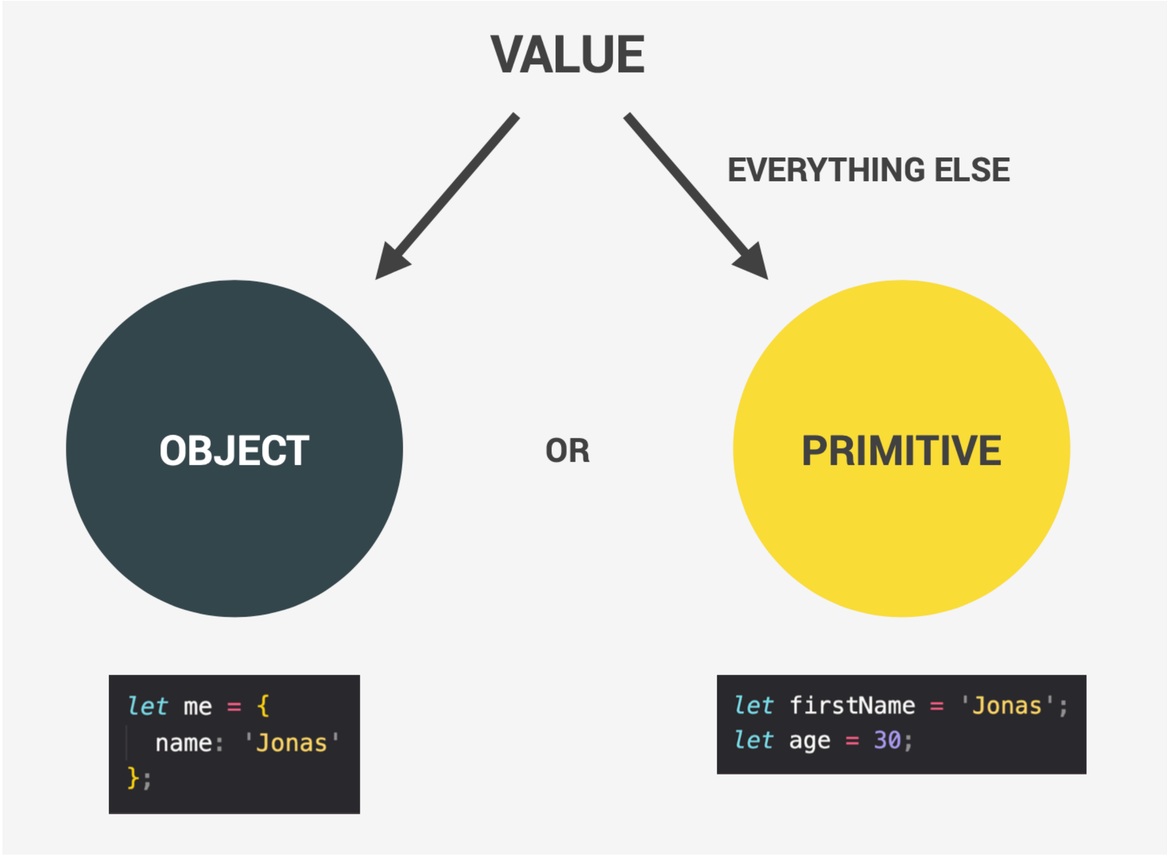
- Strängar påminner lite om en array av bokstäver
  - const course = “Grundläggande Javascript“
  - course[0]
  - course[9] = ’u’ (vi kan inte ändra strängar, dom är immutabla)
  - course.length
- Lägg märke till att const variablers innehåll kan ändras!

ARRAY

- Uppgift
  - Skapa en array ”family” med namn på dina familjemedlemmar
  - Printa ut godtycklig person ur array
  - Printa ut den sista personen ur array
  - Printa ut den första personen ur array
  - Ändra godtycklig persons namn i array
  - Printa ut längden på array

OBJECT

- Dom primitiva datatyperna är legobitar, men ibland vill skapa hus, bilar, personer



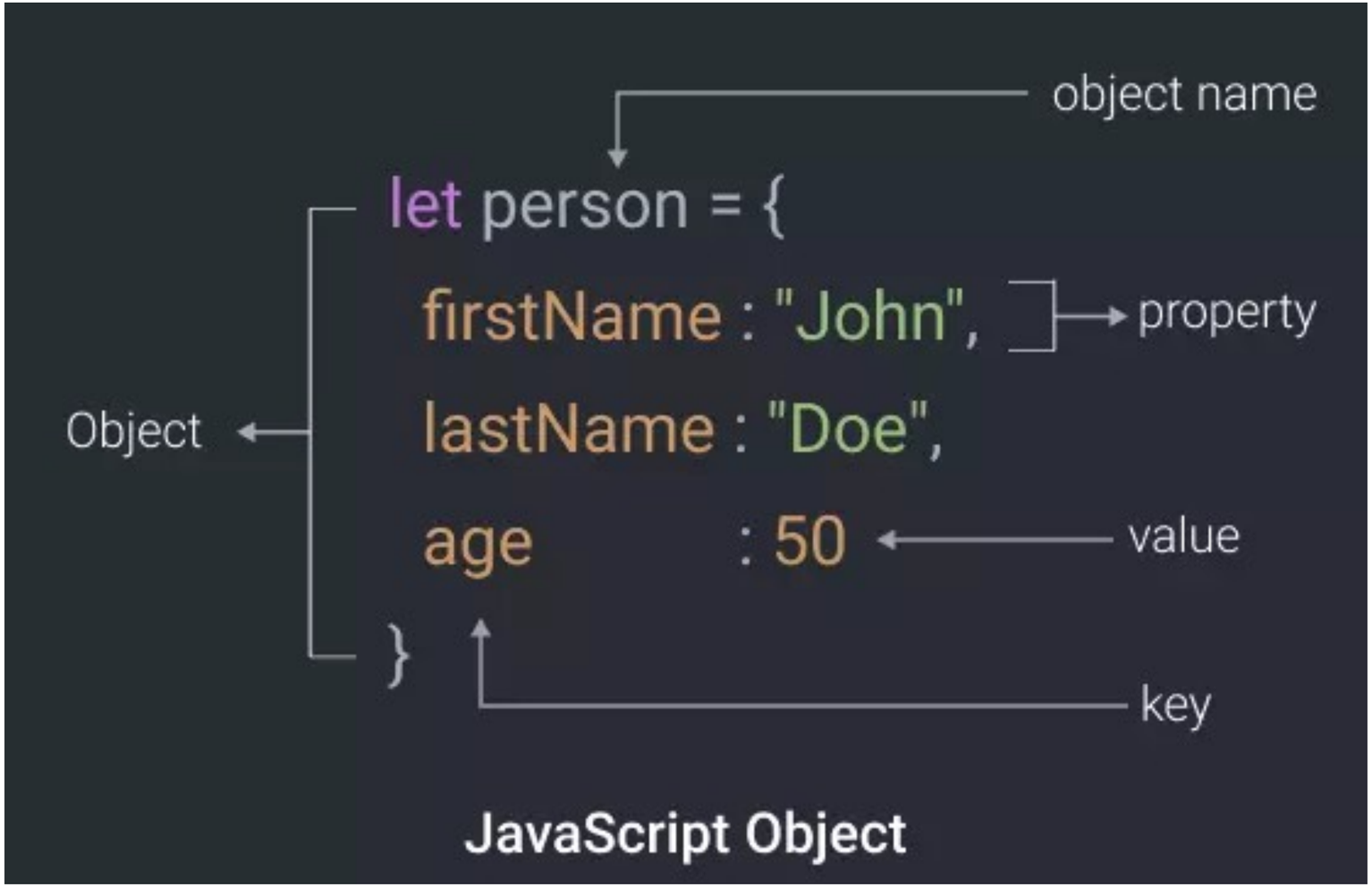
OBJECT

- Skapa ett object, snobben { name: …, age: …, gender: …, alive: … }
- Skapa ett object, mallorcaFlight { departure: …, destination: …, departureDate: …, destinationDate: …. }
- Skapa ett object, gothenburgFlight { departure: …., destination: …, departureDate: …, destinationDate: …. }
- Skapa ett object, vacationInfo { person: snobben, departureFlight: mallorcaFlight, homeFlight: gothenburgFlight, hotel: …., days: … }
- Vad händer: vacationInfo2 { snobben, mallorcaFlight, gothenburgFlight }?

OBJECT

- Uppgift
  - Skapa ett object "**person**" med properties
    - name, age, alive
  - Skapa ett object "**location**" med properties
    - address, city, postalCode
  - Skapa ett object "personInfo"
    - { person: **person**, location: **location**, student: true }
  - Skapa ett nytt object "personInfo2" men med "tricket"...

OBJECT


object name
let person = {
firstName : "John",
property
Object
lastName : "Doe",
age : 50
value
}
key
JavaScript Object

OBJECT

- Frågor
  - const arr = [”hej”, ”du”, ”sköna”, ”nya”, ”värld”];
    - Index?
    - Value?
  - const object = { course: ”Grundläggande JS”, weeks: 7 };
    - Property?
    - Value?
    - Key?

PRIMITIVES VS NON-PRIMITIVES

- Vi har sett nu att vi har vi har 2 typer av värden
  - Primitiva värden: 1, false, null, “hej”, undefined (immutable)
  - Icke-primitiva värden: { isAlien: true }, [1,2,3] (mutable)
- Dessa värden tilldelas ofta till variabler (const let)

PROPERTIES

- Vi har också sett **Math.max**, **Math.min**, **console.log**
- Som vi förstår nu är **Math** och **console** object som har properties (**max, min, log**)
- Primitiva värden har inte properties
  - 1.length (går inte)
  - true.length (går inte)
  - null.length (går inte)
  - ”Hello”.length (Går! Strängar är egentligen objekt!)

PROPERTIES

- Vi kan hämta ett objects properties på 3 olika sätt

PROPERTIES

- Exempel

```
const hero = {
  name: 'Batman'
};
```

- Även array är en specialform av objekt med properties
  - arr[0] där index är en property
    - console.log(hero) vs console.log(arr)

PROPERTIES

- const batman = { name: ”Bruce Wayne”, workDays: [”mån”, ”tis”, ”ons”] }
- Vi kan även skapa properties efter att ha skapat objektet
  - console.log(batman.name)
  - console.log(batman.isHero) // undefined
  - batman.isHero = true
  - console.log(batman.isHero) // true
- Vi kan ta bort properties
  - delete batman.name
  - console.log(batman.name) // undefined
  - console.log(”name” in batman) // false
  - console.log(”workDays” in batman) // true
- Vi kan titta på alla keys
  - console.log(Object.keys(batman))
- Ändra ett object
  - Object.assign(batman, { workDays; [”lör”, ”sön”], capitalist: true })
  - console.log(batman)

PROPERTIES

- Uppgift
  - Skapa ett objekt “wonderWoman”
    - Properties: hero (true), name (“…”), age (34), workDays ([…])
  - Printa ut hero (med object.prop)
  - Printa ut name (med object[“prop”])
  - Printa ut age (med { prop } = object)

PROPERTIES

- Uppgift (forts..)
  - Lägg till ny property ”superPower”
  - Printa ut wonderWoman
  - Ta bort age property (delete)
  - Printa ut wonderWoman
  - Kontrollera att age finns kvar på objektet (in)
  - Printa ut alla keys (Object.keys)
  - Ändra och lägg till properties { name: ”Diana Prince”, origin: ”Themyscira” }
    - (Object.assign)

PROPERTIES

- Frågor
  - Har primitiva värden properties? Finns undantag?
  - Vilka 3 sätt kan vi hämta ut properties från ett objekt?
  - Hur lägger man till en property på ett objekt?
  - Hur tar man bort en property från objekt?
  - Hur kontrollerar man att det finns en property på objekt?
  - Hur tittar man på ett objekts properties (keys)?

METHODS

- Både strängar och arrayer håller fler properties än length
  - [1,2,3].length => 3
  - ”Hello”.length => 5
- Det finns även properties som är funktioner, ex: **toUpperCase** (metod)

```
let doh = "Doh";
console.log(typeof doh.toUpperCase);
// → function
console.log(doh.toUpperCase());
// → DOH
```

METHODS

- toUpperCase, toLowerCase
  - Metoderna har inga argument men kan ändå använda texten? Vi ska titta senare hur det går till…

```
let doh = "Doh";
console.log(typeof doh.toUpperCase);
// → function
console.log(doh.toUpperCase());
// → DOH
```

- arr.push, pop, includes

MUTABILITY

- Vi har förstått att primitiva värden inte går att ändra
  - string[0] = “h“ // går ej! (immuteble)
- Men att vi kan ändra array och object properties
  - listOfNumbers[9] = 109; // OK (mutable)
  - batman.name = “Bruce Wayne“; // OK (mutable)

MUTABILITY

- Muterbarhet rör föränderighet/oföränderlighet hos **värde**
- Muterbarhet hos **variabler** beror på deklaration, vad kommer hända nedan?
  - **let** x = 1;
  - x = 2;
  - x++;
  - **const** y = 1;
  - y = 2;
  - y++;

REFERENSER

- Referenser

```
let object1 = {value: 10};
let object2 = object1;
let object3 = {value: 10};

console.log(object1 == object2);
// → true
console.log(object1 == object3);
// → false

object1.value = 15;
console.log(object2.value);
// → 15
console.log(object3.value);
// → 10
```

REFERENSER

- let och const

```
const score = {visitors: 0, home: 0};
// This is okay
score.visitors = 1;
// This isn't allowed
score = {visitors: 1, home: 1};
```

- Varför?

REFERENSER

- Uppgift
  - Skriv en funktion ”deepEqual” som tar 2 objekt och kontrollerar att båda objekten har samma properties
    - deepEqual({ a: 1, b: 2}, { a: 1, b: 2}) => true
    - deepEqual({ a: 1, b: 2}, { a: 3, b: 2}) => false
    - deepEqual({ a: 1, b: 2}, { a: 1, b: 2, c: 3}) => false
    - Tips: använd Object.keys på båda objekten
      - Kontrollera längden är lika stor, sedan i for-loop att de har samma values

```
const x = { a: 1, b: 2 }
const y = { a: 1, b: 2 }
const arr = Object.keys(x)
x[0] == y[arr[0]] // true
```

FRÅGOR

- Nämn en gemensam property för sträng och array
- Vad kallas properties i objekt som håller funktioner?
- Vilken sträng metod för att göra en sträng till stora/små bokstäver?
- Vilken array metod för att lägga till/ta bort ett värde på en array?
- Vilken array metod för att kolla om en array har ett specifikt värde?
- Hur gör jag om jag vill lägga till en property på ett existerande objekt?

FRÅGOR

- Vad skiljer värden (primitives, non-primitives)?
- Vad skiljer const, let?
- Om jag har en variabel ”batman” och tilldelar variabeln till en ”copyBatman”
  - Vad ger batman == copyBatman?
- Om jag har två likadana variabler ”batman1” och ”batman2”
  - Vad ger batman1 == batman2?

FRÅGOR

- Ok eller inte? Förklara…
- **let** x = { name: ”batman” }, **const** y = { name: ”wonder woman” }
  - x = 1?
  - x[”name”] = ”bruce”?
  - y = true?
  - y.name = ”Diana”?

SUMMERING (VAD HAR VI LÄRT OSS?)

- Array (vad är index och value?)
- Object (vad är en property?)
  - Vilka 3 sätt finns för att hämta en propertyvalue?
  - Primitives vs Non-primitives (ge exempel, vad skiljer dem?)
  - Methods (vad är en metod?)
- Mutability
- References

ÖVNINGAR

1. Create an array containing 4 population values of 4 countries of your choice. You may use the values you have been using previously. Store this array into a variable called `'populations'`
2. Log to the console whether the array has 4 elements or not (`true` or `false`)
3. Create an array called `'percentages'` containing the percentages of the world population for these 4 population values. Use the function `'percentageOfWorld1'` that you created earlier to compute the 4 percentage values

ÖVNINGAR

1. Create an array containing all the neighbouring countries of a country of your choice. Choose a country which has at least 2 or 3 neighbours. Store the array into a variable called `'neighbours'`
2. At some point, a new country called *'Utopia'* is created in the neighbourhood of your selected country. So add it to the end of the `'neighbours'` array
3. Unfortunately, after some time, the new country is dissolved. So remove it from the end of the array
4. If the `'neighbours'` array does not include the country *'Germany'*, log to the console: *'Probably not a central European country :D'*
5. Change the name of one of your neighbouring countries. To do that, find the index of the country in the `'neighbours'` array, and then use that index to change the array at that index position. For example, you can search for *'Sweden'* in the array, and then replace it with *'Republic of Sweden'*.

ÖVNINGAR

1. Create an object called `'myCountry'` for a country of your choice, containing properties `'country'`, `'capital'`, `'language'`, `'population'` and `'neighbours'` (an array like we used in previous assignments)

ÖVNINGAR

1. Using the object from the previous assignment, log a string like this to the console: *'Finland has 6 million finnish-speaking people, 3 neighbouring countries and a capital called Helsinki.'*
2. Increase the country's population by two million using **dot notation**, and then decrease it by two million using **brackets notation**.

- **Dot notation**: obj.prop
- **Bracket notation**: obj[”prop”]

OBJECTS AND ARRAYS

- Läxa
  - Eloquent JavaScript
    - https://eloquentjavascript.net/13_browser.html
    - https://eloquentjavascript.net/14_dom.html
    - Vi kommer att gå igenom DOM imorgon på övning samt börja med inlämning