



**JavaScript 1**  
**alert("Lektion 8");**

Utbildare: Mahmud Al Hakim

**NACKADEMIN**

# Lektionstillfällets mål

- **Mål med lektionen**
  - Versionshantering: Att återställa filer.
  - Mer om objekt och arrayer
  - Nyckelordet this
  - Inbyggda objekt
  - BOM och DOM
- **Arbetsmetod**
  - Teori och praktik varvas under lektionen

# Kort summering av föregående lektion

## Föregående lektion:

- Vi har jobbat med objekt.
- Egenskaper och metoder.
- Primitiva typer och referenstyper

# JavaScript Value vs Reference Types med Mosh

[https://youtu.be/fD0t\\_DKREbE](https://youtu.be/fD0t_DKREbE)

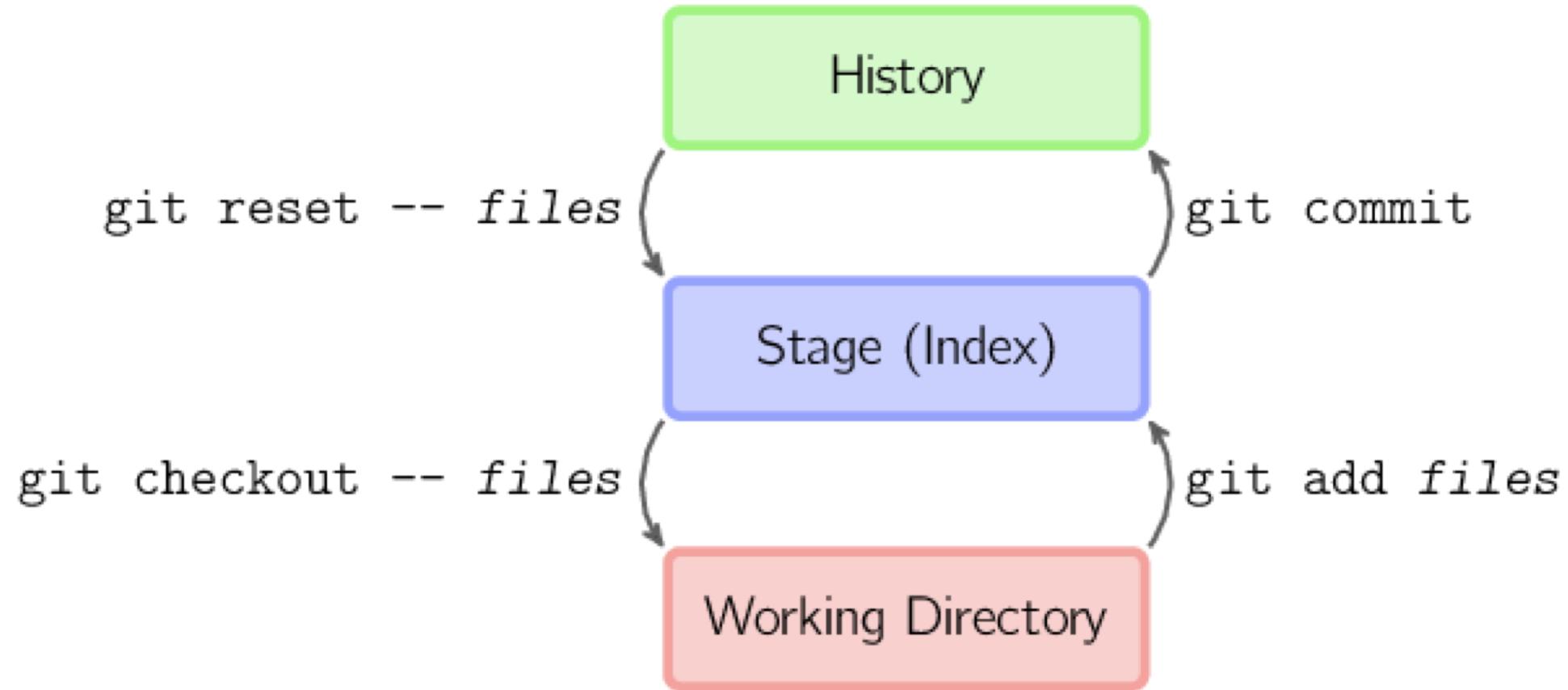
## Value vs Reference Types



**Mosh Hamedani**

[programmingwithmosh.com](http://programmingwithmosh.com)

# Git reset och checkout



<https://github.com/rangle/git-training/blob/master/handouts/getstarted.md>

# Git reset och checkout – Demo

1. Gör några avsiktliga fel i dina filer
2. Lägg dessa filer till din "Stage"
3. Gör en commit
4. Skriv ut en log på en rad

```
git add .  
git commit -m "FEL"  
git log --oneline
```

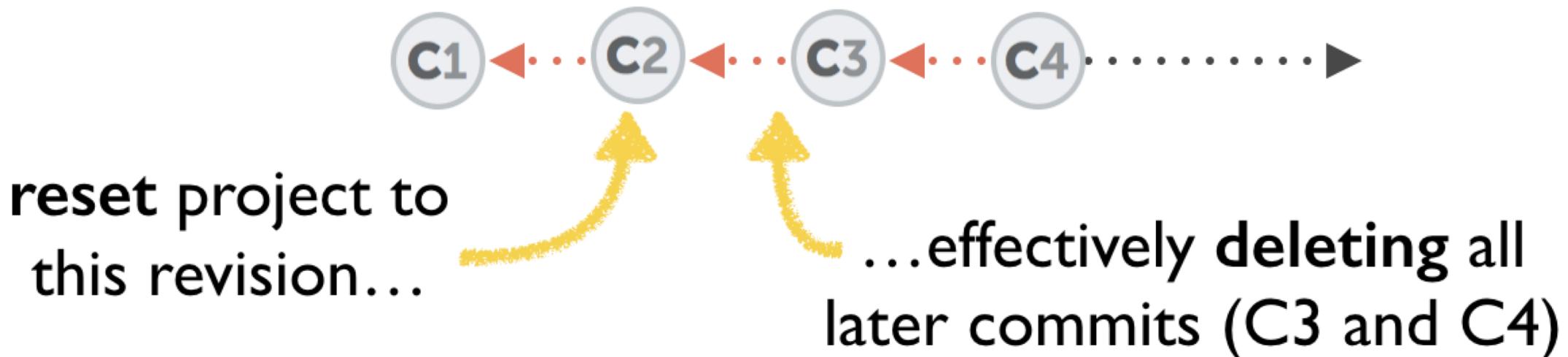


```
Mahmuds-MacBook-Air:JavaScript-1 mahmudalhakim$ git log --oneline  
fc47716 (HEAD) ERRORS  
67be94f (origin/master, master) Några swap-algoritmer
```

5. Leta efter vilken commit du vill gå till t.ex. **67be94f**
6. Gör en reset för att återställa filerna till din stage
7. Gör en checkout på önskade filer för att återställa

```
git reset 67be94f  
git checkout .
```

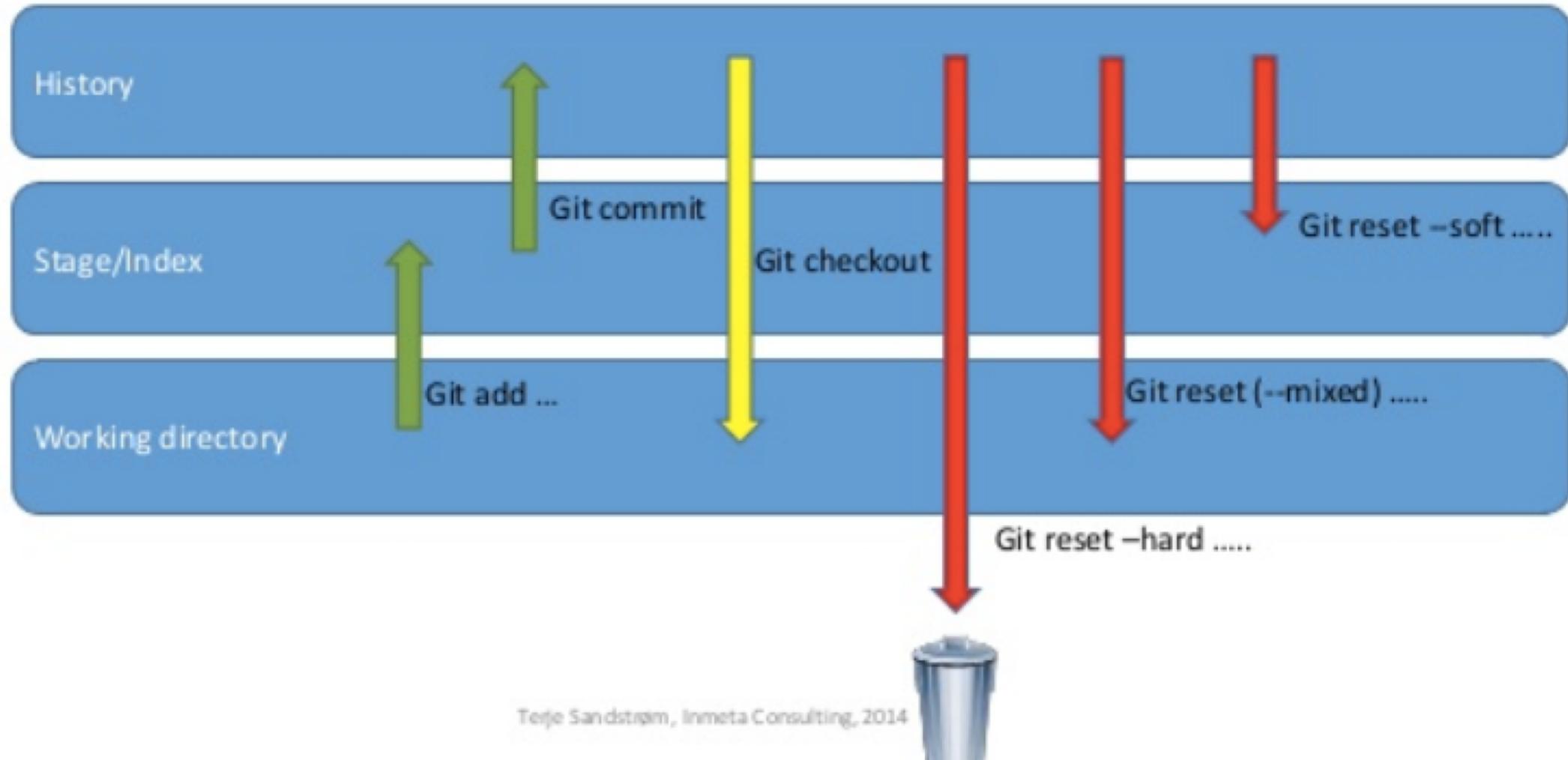
# Att återställa till en tidigare version ”snabbt”



```
$ git reset --hard 0ad5a7a6
```

<https://www.git-tower.com/learn/git/faq/delete-commits>

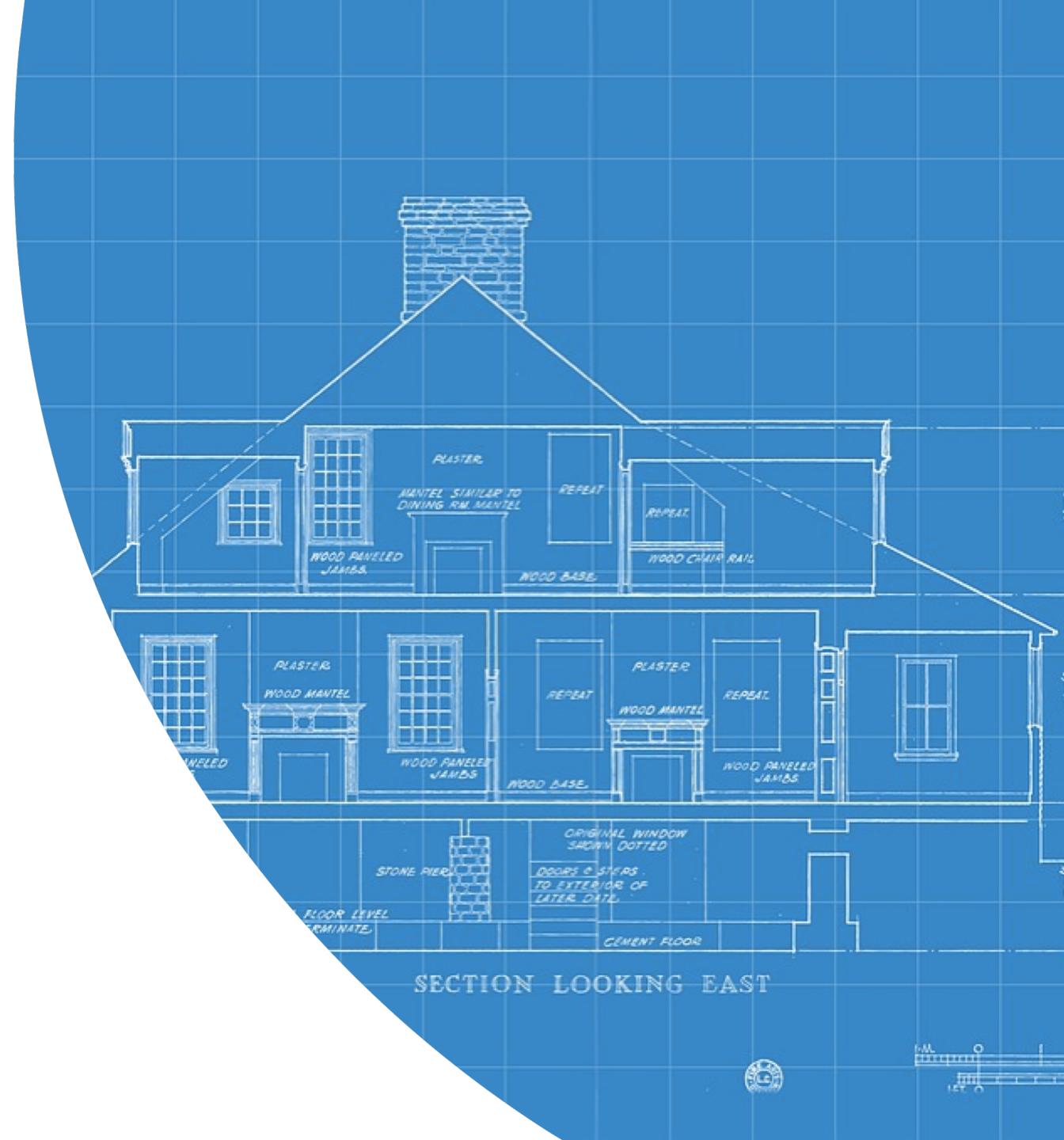
# Git tree movements visualized



# Vad är en konstruktor (constructor)?

---

- En konstruktor är en metod som skapar (initierar) ett objekt.
- Kallas även en *konstruktör* eller *konstruerare*.
- En konstruktor används som en **“blueprint”** för att skapa flera objekt av samma typ.
- Konstruktorer skrivs helst i PascalCase t.ex. Object, Person, Car osv.



# Skapa ett objekt med hjälp av en konstruktor (constructor notation)

```
let mahmud = new Object();
mhmud.name = "Mahmud Al Hakim";
mhmud.age = 45;
mhmud.info = function () {
    console.log(this.name + " är " +
    this.age + " år gammal!");
}
```

Ett nytt blankt  
objekt skapas med  
nyckelordet **new**  
och **Object()**  
som kallas  
en konstruktor  
(constructor)

```
var hotel = new Object();
```

```
hotel.name = 'Quay';
```

```
hotel.rooms = 40;
```

```
hotel.booked = 25;
```

PROPERTIES

```
hotel.checkAvailability = function() {
```

```
    return this.rooms - this.booked;
```

```
};
```

METHOD

# Skapa flera objekt med en egen konstruktor (Constructor Function)

```
function Person(name, age){  
    this.name = name;  
    this.age = age;  
}  
  
let mahmud = new Person("Mahmud Al Hakim" , 45);  
let kalle = new Person("Kalle Anka" , 10);
```

```
function Hotel(name, rooms, booked) {
```

```
    this.name = name;  
    this.rooms = rooms;  
    this.booked = booked;
```

PROPERTIES

```
    this.checkAvailability = function() {  
        return this.rooms - this.booked;  
    };
```

METHOD

```
}
```

OBJECT

CONSTRUCTOR FUNCTION

ASSIGNMENT OPERATOR

NEW KEYWORD

VALUES USED IN PROPERTIES OF THIS OBJECT

```
var quayHotel = new Hotel('Quay', 40, 25);
var parkHotel = new Hotel('Park', 120, 77);
```

# Skapa en array av objekt

```
let mahmud = new Person("Mahmud Al Hakim" , 45);  
let kalle = new Person("Kalle Anka" , 10);  
  
let persons = [ mahmud , kalle ];  
console.log(persons[0]);  
console.log(persons[0].name);
```

# Objekt kan innehålla arrayer

```
function Person(name, age){  
    this.name = name;  
    this.age = age;  
    this.children = [];  
}  
  
let mahmud = new Person("Mahmud Al Hakim" , 45);  
let mehdi = new Person("Mehdi Al Hakim" , 23);  
let hoda = new Person("Hoda Al Haim" , 21);  
  
mahmud.children.push(mehdi);  
mahmud.children.push(hoda);  
  
console.log(mahmud.children);
```

Kom ihåg!

Array är en referenstyp

```
mahmud.children[0].name = "Kalle Anka";
```

```
mahmud.children[0].age++;
```

```
console.log(mahmud.children);
```

# Nyckelordet **this**

- **this** refererar till det aktuella objektet.
- Om du använder **this** på en funktion som har skapats globalt, alltså i "The Global Scope" så pekar **this** på objektet Window i webbläsaren.

```
function demo(){
    console.log(this);
};

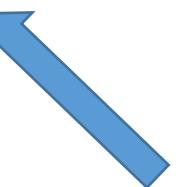
demo();
```

# This är en global funktion – Exempel

```
function windowSize() {  
    let width = this.innerWidth;  
    let height = this.innerHeight;  
    return [height, width];  
}  
alert(windowSize());
```

# Globala variabler deklarerade med **var**

```
var innerWidth = 1000;           // En global variabel
function windowSize() {
    let width= this.innerWidth; // this pekar på den globala variabeln
    let height = this.innerHeight;
    return [height, width];
}
alert(windowSize());
```



# Globala variabler deklarerade med `let`

```
let innerWidth = 1000; // En global variabel

function windowSize() {
    let width = this.innerWidth; // this pekar fortfarande på window
    let height = this.innerHeight;
    return [height, width];
}
alert(windowSize());
console.log(innerWidth); // 1000
```

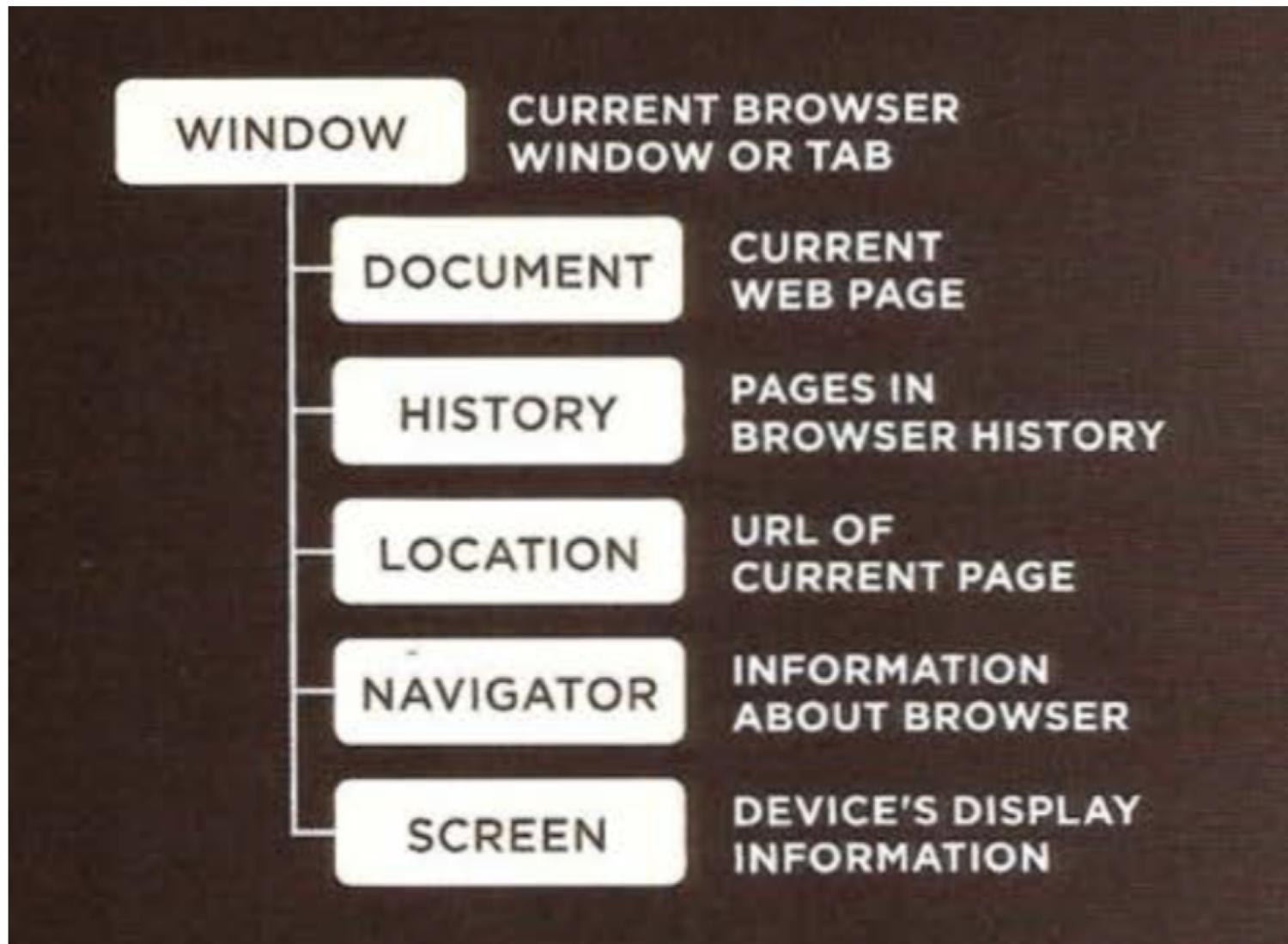
# Inbyggda objekt (built-in objects)

BOM  
Browser  
Object Model

DOM  
Document  
Object Model

Global  
JavaScript  
Objects

# BOM (Browser Object Model)



# Några användbara egenskaper i objektet Window

```
console.log("innerWidth: " + window.innerWidth);
console.log("innerHeight: " + window.innerHeight);
console.log("screen.width: " + window.screen.width);
console.log("screen.height: " + window.screen.height);
console.log("location.href: " + window.location.href);
console.log("navigator.onLine : " + window.navigator.onLine);
```

# Några användbara metoder i objektet Window

```
let person = prompt("Please enter your name");

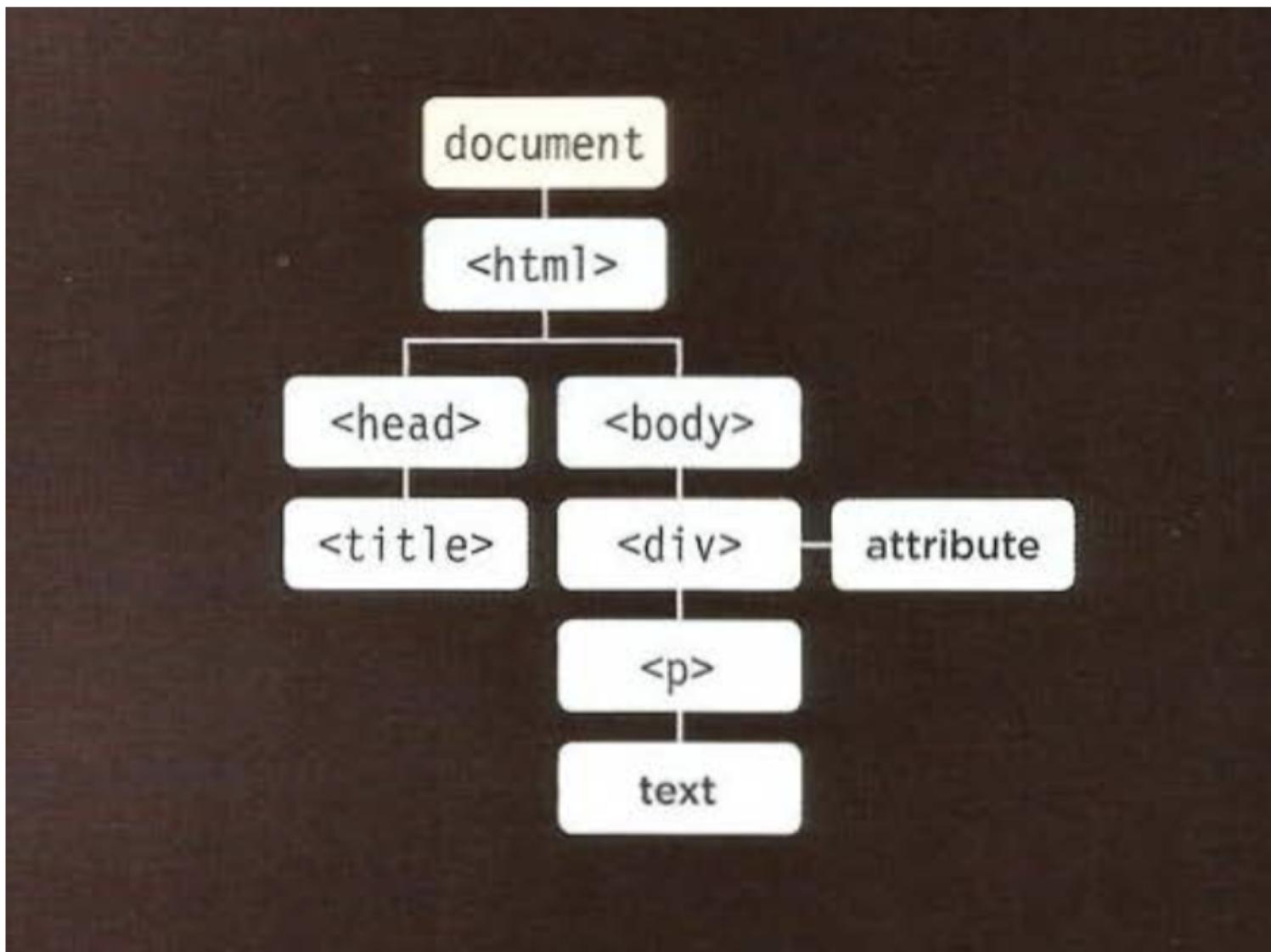
setTimeout(helloPopup, 1000);

function helloPopup(){
    alert("Hello " + person + "!");
}

setInterval(helloConsole, 1000);

function helloConsole(){
    console.log("Hello " + person + "!");
}
```

# DOM (Document Object Model)



## Några användbara egenskaper i objektet Document

```
console.log(document.title);  
console.log(document.lastModified);  
console.log(document.URL);
```

# Övning 1

- Skapa en array som innehåller information om dina kurser.
- Arrayer ska alltså innehålla ett antal objekt.
- Varje objekt presenterar en kurs.
- Varje kurs ska ha följande egenskaper
  - Kursnamn (string)
  - Lärare (string)
  - Antal poäng (number)
  - Betyg (string eller null)
  - Avklarad (boolean)

## Övning 2

- Skapa en array som innehåller info om dina favoritrecept (minst 2 recept)
- Varje recept är ett objekt som har följande egenskaper.
  - Titel (string)
  - Gör så här (string)
  - Ingredienser (array)
- OBS! Egenskapen Ingredienser är alltså en array som innehåller en lista över alla ingredienser.

# Övning 3

Skapa ett skript som körs efter 5 sekunder.

Visa meddelandet  
( i en alert)  
"Anmäl dig till vårt  
nyhetsbrev och få 10%  
rabatt idag".

Bra sammanfattning: Objects & Arrays

<https://youtu.be/FLGzeTHAbqQ>

Objects & Arrays

# Summering av dagens lektion

- Vi har idag jobbat lite mer med objekt och arrayer.
- Vi har jobbat lite med BOM och DOM.
- Reflektioner kring dagens lektion?
  - Vad tar du med dig från dagens lektion?
  - Finns det något som var extra svårt att förstå?
  - Finns det något som vi behöver repetera?
  - Hur upplevde du dagens arbetsmetoder?

# Framåtblick inför nästa lektion

- Läs: Bok 2, sid. 113–127
- Under nästa lektion kommer vi att jobba med globala JS-Objekt.