

Typer, värden, uttryck och satser

Allt är bara ettor och nollor - data!

- ✓ Allt lagras som en lång ström med bitar; 1 eller 0.

```
0 0 1 0 1 0 1 0 // binärt värde
```

```
128 64 32 16 8 4 2 1 // talbasen 10
```

- ✓ Det binära talet 00101010 är samma som det decimala talet 42, eftersom $32 + 8 + 2 = 42$.
 - Binära tal har talbasen 2; decimala tal har talbasen 10.

- ✓ Strömmen av bitar behöver delas in mindre delar - värden.

- ✓ Värden kan vara av olika typer, som t.ex.:

- tal: 42
- strängar: 'Hej hopp!'

```
01001000 01100101 01101010 01000000 01101000 01101111 01110000 01110000 00100001 // binärt värde
```

```
72 101 106 32 104 111 112 112 33 // ASCII-värde, numeriskt värde som representerar ett tecken
```

```
H e j h o p p ! // tecken
```

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	Space	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	\$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q

Olika typer av data

✓ JavaScript är löst typat språk.

- Sju primitiva typer: Undefined, Null, Boolean, Number, BigInt, Symbol och String.
- En objekttyp, Object, som är en osorterad lista med namn/värde-par.
 - Exempel på globala objekttyper baserade på Object är:
 - Array, objekt av typen Array representerar en samling av värden.
 - Date, objekt av typen Date representerar ett datum och tid.
 - Error, objekt av typen Error representerar ett fel.
- Alla värden måste vara av någon av de åtta typerna ovan.

✓ Operatören typeof ger typen ett värde är av i form av en sträng.

```
typeof 42                // 'number'
typeof 3.14              // 'number'
typeof 'Hej hopp!'       // 'string'
typeof true              // 'boolean'
typeof {age: 42, name: 'Ellen Nu'} // 'object'
typeof function foo() { console.log('bar') } // 'function'
```

(OBS! Rent tekniskt betraktas funktioner att vara av typen Object, även om operatören typeof returnerar strängen function.)

Datatypen Number

- ✓ Alla tal lagras som 64-bitar tal; 18 446 744 073 709 551 616 olika tal. Tal kan uttryckas på flera sätt.

42 // heltal

3.14 // decimaltal (flyttal)

4.712e7 // exponentiella tal ($4,712 \cdot 10^7$)

- ✓ Aritmetiska operatorer fungerar på samma sätt som vanligt.

23 + 19 // addition (→ 42)

115 - 73 // subtraktion (→ 42)

7 * 6 // multiplikation (→ 42)

210 / 5 // division (→ 42)

18 % 5 // rest (→ 3, "remainder/modulo operator")

4 + 2 * 7 // multiplikation har högre prioritet än addition (→ 18)

(4 + 2) * 7 // parenteser har högst prioritet (→ 42)

- ✓ Speciella värden: `infinity`, `-infinity`, `NaN` (*Not-A-Number*)

Datatypen String

- ✓ Strängar används till att representera text.

'Kan skrivas inom apostrofer.'

`Kan skrivas inom grav accent.`

"Kan skrivas inom citationstecken också." // INTE i denna kurs!

- Kursens kodstandard hindrar användning av citattecken!

- ✓ Strängar kan läggas ihop, konkateneras (latin: *catena*, kedja, "ihopkedjning").

'Det går ' + 'bra att lägga ' + 'ihop strängar.'

- ✓ Nyradstecknet `\n`, en escape-sekvens, i en sträng används för att skapa en sträng över flera rader.

'En sträng kan delas\növer flera rader\nutan problem.' // En sträng kan delas

// över flera rader

// utan problem.

Datatypen Boolean

- ✓ Booleska värden används för att representera om något är sant eller falskt.
- ✓ Endast två värden är möjliga, `true` eller `false`.
- ✓ Booleska värden är ofta ett resultat av en jämförelse.

```
3 === 42 // false
```

```
7 === 7 // true
```

```
3 < 42 // true
```

```
3 > 42 // false
```

```
3 !== 42 // true
```

Sanningstabeller för &&, || och !

- ✓ Operatoren && kräver att båda värdena är true för att resultatet ska bli true.
- ✓ Operatoren || kräver att minst ett av värdena är true för att resultatet ska bli true.

&& (AND, OCH)			(OR, ELLER)			! (NOT, ICKE)	
false	false	false	false	false	false	false	true
false	true	false	false	true	true	true	false
true	false	false	true	false	true		
true	true	true	true	true	true		

OCH tänk multiplikation; ELLER tänk addition; false = 0, true = 1.

&& (AND, OCH)					(OR, ELLER)				
0	×	0	=	0	0	+	0	=	0
0	×	1	=	0	0	+	1	=	1
1	×	0	=	0	1	+	0	=	1
1	×	1	=	1	1	+	1	=	1

Det är ett objekt!

- ✓ Är det inte ett primitivt värde är det ett objekt!
 - Värdet är inte till exempel `undefined`, `null`, `true`, `42` eller `'hej hopp!'`.
- ✓ Objekt innehåller en samling av egenskaper (*”properties”*) där varje egenskap har ett namn och ett värde.

```
{ age: 42, name: 'Ellen Nu' } // värde som är ett objekt object
```

```
[ 4, 2, 15, -2, 42, 7 ] // värde som är en array (som är ett objekt!)
```

```
new Date() // ger ett värde som är ett objekt med dagens datum och tid
```


Uttryck, satser (och program)

- ✓ Uttryck ("*expression*") kan liknas med ord i en mening, där meningen i sig är en sats ("*statement*"). Satser skapar ett program.

- ✓ Kod som skapar ett värde kallas ett uttryck.

42

$(4 + 8) * 3 + 6$

- ✓ En sats kan innehålla ett eller flera uttryck.

42

5 > 3 // true

5 < 3 // false

'Mats' < 'Bäst' // false, fungerar med strängar också (M = 77, B = 66)

'Mats' < 'bäst' // true, versaler sorteras före gemener (M = 77, b = 98)

- Det är vanligt att satser avslutas med ; (semikolon), INTE i denna kurs dock!
- Kursens kodstandard kräver att ; ALDRIG används. (Ibland måste ; användas, det är några specialfall!)

