Data And Variables

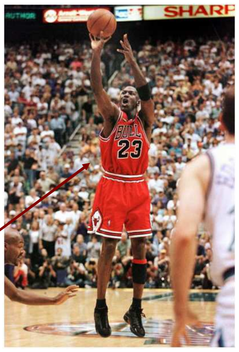
Chapter 18

Names vs. Values

name (the letter sequence used to refer to something)

Michael Jordan

value (the thing itself)

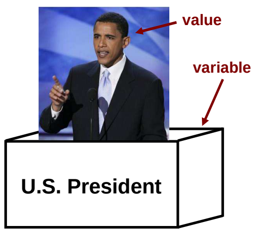


Names Have Changing Values

- We do not tend to distinguish between names and values, because things do not magically transform into other things.
- In programming, names and values are separable.
  - Think of names as offices or titles, like "U.S. President"
  - Example:
    - Previous *values* of the *name* "U.S. President":

Names Have Changing Values

- Another way to think of names is as labeled boxes.
- We refer to these labeled boxes as **variables**.
- **variable**: a piece of your computer's memory that is given a name and can store a value



Identifier

- **identifier**: the letter sequence that makes up a variable's name
  - Must begin with a letter, underscore (_), or dollar sign ($)
  - Following characters can also include digits
  - Cannot contain spaces
- Identifiers are case-sensitive.
  - Example: `Fred` and `fred` and `FrEd` are **NOT** the same.

Examples

- Valid
  - X
  - x
  - height
  - Time_of_day
  - oOoOoOo
  - w1o2o3h4o5o
  - Identifiers_can_be_long_but_typing_them_can_be_a_pain
- Invalid
  - **space not allowed**
  - Michael Jordan
  - 1stValue
  - yay!
  - mininum-number

- Variables should have meaningful identifiers that are descriptive of the value stored in the variable.

Keywords

- The following list are *keywords* that have special meaning in JavaScript and thus may not be used as identifiers.

```
abstract    boolean      break      byte     case
catch       char         class      const    continue
debugger    default      delete     do       double
else        enum         export     extends  false
final       finally      float      for      function
goto        if           implements import   in
instanceof  int          interface  long     native
new         null         package    private  protected
public      return       short      static   super
switch      synchronized this       throw    throws
transient   true         try        typeof   var
void        volatile     while      with
```

Keywords

- As JavaScript is case-sensitive, you could technically use `Class` or `cLaSs` as identifiers, but this is very confusing and thus **strongly discouraged.**

Declaring Variables

- Have to tell computer what variables you want.
- Variable declaration *syntax*:
  `var` ***<identifer>***`;`
- Examples:
  ```
  var x;
  var myGPA;
  ```

Syntax

- **syntax**: set of legal structures and commands that can be used
- Example:
  - Every basic statement ends with a semi-colon.

Declaring Variables

- Declaring a variable sets aside a piece of memory in which you can store a value.

  ```
  var x;
  var y;
  ```

  - Inside the computer:

    `x:` ? `y:` ?

  (The memory has no value yet. It is *undefined*.)

Declaring Multiple Variable At Once

- Can declare multiple variables at once:
  `var` ***<name>***`,` ***<name>***`,` **…**`,` ***<name>***`;`
- Example:
  `var x, y, z;`

  `x:` ? `y:` ? `z:` ?

Exercise

- What is the difference between the following two sets of variable declarations?

```
var Alpha, Beta;

var beta, alpha;
```

- Both will create two variables. However, the two statements will declare different variables, because JavaScript is case-sensitive.
  - *The order of the variables in a declaration is unimportant.*

Expression

- **expression**: data value or a set of operations that produces a value
- Examples:

```
1 + 4 * 3
3
(1 - 2) / 3 * 4
-2382
"yay!"
'hello'
```

$\}$ **see slide #28 on "Strings"**

Operators

- Arithmetic operators we will use:

`+`	addition
`-`	subtraction or negation
`*`	multiplication
`/`	division

Evaluating Expressions

- When the computer *executes* (runs) a program and encounters an expression, the expression is *evaluated* (i.e., computed).
  - Example: `3 * 4` *evaluates* to `12`

Precedence: Remember PEMDAS?

- **precedence**: order in which operations are computed in an expression
  - Operators on the same level are evaluated from left to right.
    Example: `1 - 2 + 3` is `2` (not `-4`)
  - Spacing does not affect order of evaluation.
    Example: `1+`**`3 * 4`**`-2` is `11`

Parentheses	`()`
Multiplication, Division	`* /`
Addition, Subtraction	`+ -`

Assigning Values To Variables

- **assignment statement**: statement that stores a value into a variable
- Assignment statement syntax:
  <variable> = ***<expression>***`;`
- Examples:

```
myGPA = 3.25;      x: 8       myGPA: 3.25
x = 2 * (1 + 3);
```

Assignment vs. Algebra

- The assignment statement is not an algebraic equation!
- ***<variable>*** `=` ***<expression>*****`;`** means:
  - "store the value of ***<expression>*** into ***<variable>***"
- Some people read `x = 3 * 4;` as
  - "x gets the value of `3 * 4`"
- **ERROR**: `3 = 1 + 2;` is an illegal statement, because 3 is not a variable.

19

Assigning Values To Variables

- We often know an initial value for the variables we declare.
- A variable can be declared and assigned an initial value in the same statement.
- Declaration/initialization statement syntax:
  `var` ***<identifier>*** `=` ***<expression>*****`;`**
- Example:
  `var taxRate = 0.088;`

20

Declaring/Initializing Multiple Variables

- Can declare/initialize multiple variables at once:
  `var` ***<name>*** `=` ***<exp>***, ***…***, ***<name>*** `=` ***<exp>***`;`
- Examples:
  `var taxRate = 0.088, balance, years = 15;`

```
taxRate: 0.088   balance: ?   years: 15
```

21

Using Variables

- Once a variable has been assigned a value, it can be used in expressions.

```
var x = 2 * 4;
var y = x * 5 - 1;
```

  - The above statement is equivalent to:

```
var y = 8 * 5 - 1;
```

22

Assigning Variables

- A variable can be assigned a value more than once.
- Example:

```
var x = 1.5;
x = 3;
x = 4 + 7;     // x now has a
               // value of 11
```

23

Assignment Conundrum

- What happens when a variable is used on both sides of an assignment statement?

```
var x = 3;
x = x + 2;   // what happens?
```

  Answer: x now has a value of 5.

24

Exercise

```
var x;
x = 3;
var y;
y = x;
x = 5;
```

- What is in x? What is in y?
  - x has the value of 5
  - y has the value of 3

What Can We Put In Variables?

- For now, we will use three *types* of data:
  - number
  - string
  - Boolean

Writing Numbers

- Do not write units, percent signs, dollar signs, or commas.
- Valid:
  0.088
  -273
  1000000
- Invalid:
  8.8%
  $99
  1,234
  20kg

String

- **string:** A sequence of text characters.
  - Start and end with single or double quotation mark characters
  - Starting and ending quotation marks must match (both single or both double)
- Examples:

```
'hello'
"This is a string"
"This, too, is a string.    It can be very long!"
'Bob said: "You stink!"'
```

More On Strings

- A string may not span across multiple lines.

```
"This is not
a legal string."
```

- The minimum number of characters in a string is zero, which is called the *empty string*.
- Examples:

```
""
''
```

Boolean

- There are only two Boolean values
  - `true`
  - `false`
- Remember, JavaScript is case-sensitive, so `False` is not a Boolean value.
- Boolean values will be used when we discuss conditionals in a future lecture.