



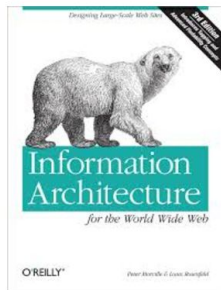
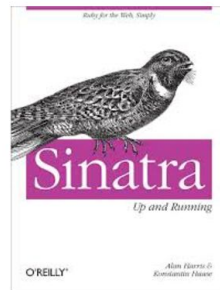
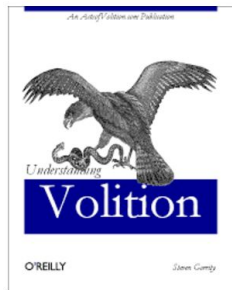
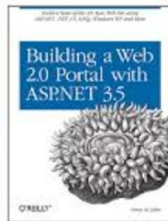
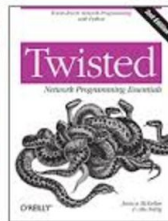
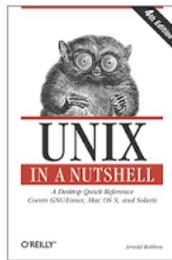
WELCOME TO FRONT-END WEB DEVELOPMENT

Please sit next to a different classmate
and write your name on your name tag.

Wi-fi: GA-Guest
pw: yellowpencil



O'REILLY®



The internet will make those bad words go away



Essential

Googling the Error Message

O RLY?

The Practical Developer
@ThePracticalDev

Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting from Stack Overflow

O RLY?

The Practical Developer
@ThePracticalDev

Software can be chaotic, but we make it work



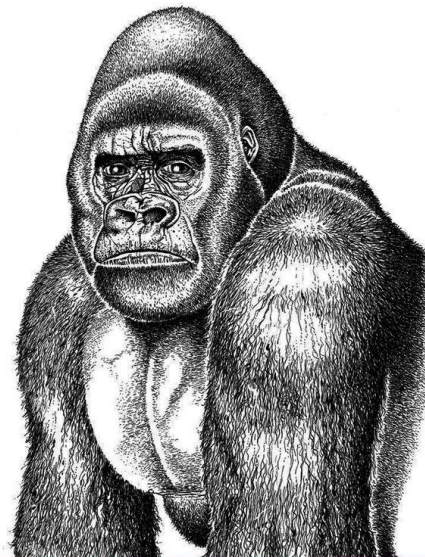
Expert

Trying Stuff
Until it Works

○ RLY?

The Practical Developer
@ThePracticalDev

Who are you kidding?



“Temporary”
Workarounds

○ RLY?

@ThePracticalDev

Does it run? Just leave it alone.



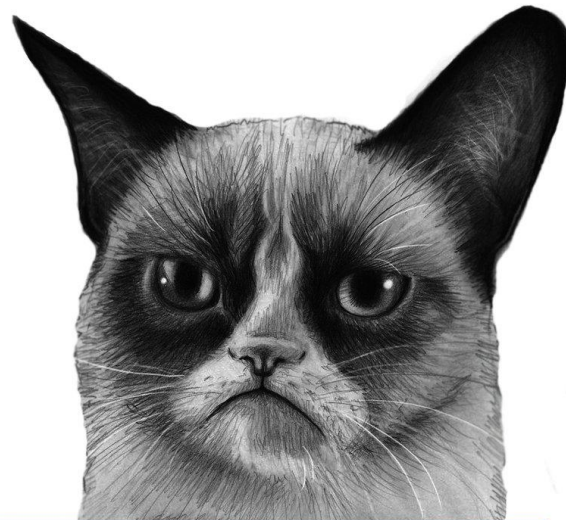
Writing Code that Nobody Else Can Read

The Definitive Guide

O RLY?

@ThePracticalDev

You're a 10x hacker and it must be someone else's fault.

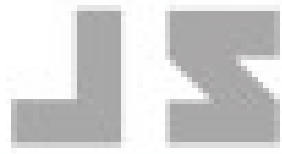


Blaming the User

Pocket Reference

O RLY?

@ThePracticalDev



Lesson 10 & 11

JavaScript

FUNDAMENTALS

RECAP

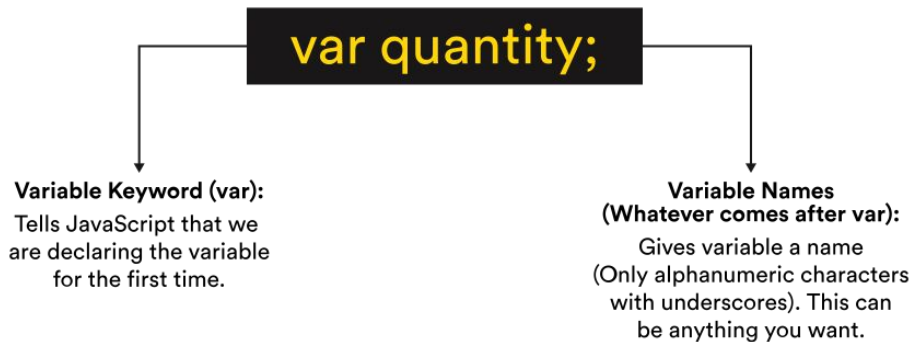


Variables

- declared using `var varName;`
- case-sensitive and written in camelCase (by convention)
- defined by assigning a value: `var varName = "foo";`
- always end with a semicolon
- can be re-assigned:

```
var varName = "foo";  
varName;  
// => "foo"
```

```
varName = "bar";  
varName;  
// => "bar"
```



Data Types

- assigning a value in quotes makes it a string
- numbers are defined without quotes
- numbers can be:
 - integers (whole numbers):
 - ..., -1, 0, 2, 5, ...
 - floats (decimals)
 - 2.71, 3.14, .5, etc

```
// string  
var testString = "foo";
```

```
// integer number  
var currentYear = 2017;
```

```
// float number  
var pi = 3.14159;
```

1. Numeric	2. String	3. Boolean
Handles numbers	Consists of letters and/or other characters	Handles true or false values
Ex: 200.54 Ex: 893	Ex: 'GA@ga.co' Ex: "How are you user?"	Ex: true Ex: false
Used for tasks that involve counting or calculating	Used when working with any kind of text Written with single or double quotes	Used when there are two options for a value (i.e. yes/no, on/off, true/false)

Operators – Arithmetic

	Operator	Example	Result
Addition	+	$2 + 4$	6
Subtraction	-	$8 - 1$	7
Multiplication	*	$2 * 3$	6
Division	/	$4 / 2$	2
Modulus	%	$4 \% 2$	0

Operators – Assignment

	Initial Value	Operator	Example	Result
Assign value to variable	var num = 8	=	num = 6	6
Add value to variable	var num = 8	+=	num += 6	14
Subtract value from variable	var num = 8	-=	num -= 6	2

Operators – Concatenation

```
var firstName = "Han";  
var lastName = "Solo";  
firstName + lastName;  
// => "HanSolo"
```

Operators – Comparison and Equality

Comparison operators compare two values against one another and return a boolean value — either `true` or `false`.

Comparison Operators	
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Equality operators check whether two values are the same as, or equal to, one another.

Equality (`===`) : evaluates `true` if both sides are **completely identical** in data **type and value**.

Example: `(5 === 5)` will evaluate to `true`, while `(5 === '5')` will evaluate to `false` since, while the values are the same, `5` is a number and `'5'` is a string.

Inequality (`!==`) : It is essentially the reverse of the **equality operator** — it compares two values to check that either the data type or value are not the same.

Operators – Logical

Logical operators give us the ability to control the flow of our programs:

- NOT (!): Evaluates whether a value **is not true**
- OR (| |): Evaluates whether at least one value **is true**
- AND (& &): Evaluates whether **both values are true**

&& and

|| or

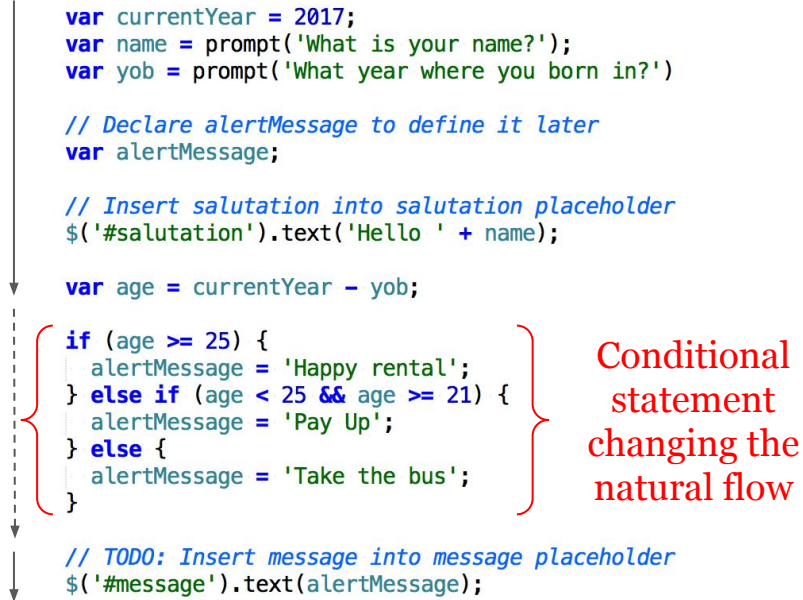
! not

Control Flow

JavaScript runs synchronously and top-down.

Without specifying any conditions, the code is ran line-by-line from top to bottom.

Changing this natural flow via conditional decision making is called **control flow**.



```
var currentYear = 2017;
var name = prompt('What is your name?');
var yob = prompt('What year where you born in?')

// Declare alertMessage to define it later
var alertMessage;

// Insert salutation into salutation placeholder
$('#salutation').text('Hello ' + name);

var age = currentYear - yob;

if (age >= 25) {
  alertMessage = 'Happy rental';
} else if (age < 25 && age >= 21) {
  alertMessage = 'Pay Up';
} else {
  alertMessage = 'Take the bus';
}

// TODO: Insert message into message placeholder
$('#message').text(alertMessage);
```

Conditional statement changing the natural flow

Conditional Statements

Conditions are statements that **make comparisons** that evaluate to `true/false` and control the flow of the program



```
if (grade >= 80) {  
  console.log("You've passed!");  
}
```

Example of other conditions:

```
(agreedToUserTerms === true)
```

```
(age < 25 && age >= 21)
```

```
(name === 'John' || surname === 'Doe')
```

Assignment	Comparison
	
<code>var number = 7;</code>	<pre>if (number === 8) { // Do something }</pre>

NOTE: don't confuse the **assignment operator** (`=`) with the **comparison operator** (`===`). When defining conditions and making comparisons remember to use the *triple equals*.

Conditional Statements

Use the `if` statement to execute a statement if a logical condition is true. Use the **optional** `else` clause to execute a statement if the condition is false.

`if...else` statement

```
if (grade >= 80) {  
    console.log("You've passed!");  
} else if (grade < 80 || grade >= 75) {  
    console.log("Just a bit more.");  
} else {  
    console.log("Sorry!");  
}
```

Conditional Statements

A `switch` statement allows a program to evaluate an expression and attempt to match the expression's value to a `case` label. If a match is found, the program executes the associated statement.

switch statement

```
switch (fruitType) {  
  case "Oranges":  
    console.log("Oranges: 59¢/lb");  
    break;  
  case "Apples":  
    console.log("Apples: 79¢/lb");  
    break;  
  default:  
    console.log("Sorry. We are out of that fruit");  
    break;  
}
```

Loops and Iteration

The `for` statement creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement to be executed in the loop.

`for` statement

```
for (var i = 1; i <= 10; i++) {  
    // this will log 1 through 10  
    console.log(i);  
}
```

The above `for` statement is saying:

- starting with `i` as 1 (`i = 1`)
 - and until `i` is less than or equal to 10 (`i <= 10`)
 - execute the code inside me (`console.log(i);`)
 - and increment `i` by 1 every time you do (`i++`)
-

Loops and Iteration

A `while` statement executes its statements as long as a specified condition evaluates to true.

`while` statement

```
var n = 0;
while (n < 10) {
  i++;
  console.log(n);
}
```

The above `while` statement is saying:

- starting with `n` as 0 (`n = 0`)
 - and as long as `n` is less than 10 (`n < 10`)
 - execute the code inside me (`i++; console.log(n);`)
-

DRY

KISS

Don't Repeat Yourself

Keep It Simple Stupid

Beware of **premature optimization**

1. Implement the basic functionality
2. Look for repetition and patterns
3. Improve and optimize

Functions

In JavaScript, a function can be:

- Made up of either a single reusable statement or a group of reusable statements.
- Called from anywhere in the program, which allows for the statements inside a function to not be written over and over again

```
// Define the function
// NOTE: Always define a function before using it
var errorAlert = function () {
    alert("Please fill out all required fields.");
};

// Call the function
errorAlert();
```

Defining Functions

Functions can be defined:

```
// As function declarations
function showAlert () {
    alert("Please fill out all required fields.");
};
```

```
// As function expressions
var showAlert = function () {
    alert("Please fill out all required fields.");
};
```

Function parameters and arguments

Functions can have **parameters** that are used as placeholders for data that can be passed to them as **arguments** when the function is called.

Parameters	Arguments
The variables that are defined in the function's declaration when the function is defined.	The actual values passed into the function when the function is called.
Ex: <pre>var doSomething = function (parameter) { // does something }</pre>	Ex: <pre>doSomething(argument)</pre>

```
// flavor and numberScoops are parameters  
var makeCone = function (flavor, numberScoops) {  
  console.log("Coming right up!");  
  console.log("Flavor:" + flavor);  
  console.log("Scoops:" + numberScoops);  
};
```

```
// "chocolate" and 3 are arguments passed to the  
function  
makeCone("chocolate", 3);
```
