Why are we here:

* Learn how to code
* Learn how to do anything

Successful student:

* Knows his strength and weakness
* Understand how he or she learns
  + An accountant would re-watch all the lectures
* Keep on trying
* Attend the classes
* Be genuinely curious
  + Hackernews
* Social student
  + Need to be able and willing to help people out
  + Ask questions to fellow students
  + People who asked the least questions got the least out of it
* Know what you understand and don’t understand
* Knows what we are going to learn the first few days
  + Rock solid basis
  + First week a bit dry and theoretical
  + Stick through it

Successful programmer:

* Four main skills:
  + Understand variables very well
  + Understand IF statements
    - Just like propositional logic
  + Understand a loop:
    - Step 1 to 3
    - Back to step 1 and so forth
  + Understand functions
    - Cleaner cleans
    - Accountant accounts
    - Every specific unit has a thing to do
    - Final frontier
* Additional skills:
  + Know how to google
  + Hackernews or reddit
    - Allow you to socialize
  + Dribble.com
  + Find your hook, find your addiction to web development
    - You’ll be a lot more motivated
* Problem based learning style
  + Effective learning & career skills
  + Responses to people only listening to teachers
* The problem of learning
  + Programmers memorize ~~understand~~ code & logic
  + Professionals ~~reproduce knowledge~~ apply skills
* Goals of our time together:
  + Learn how to code
  + Learn how to learn code
* The profit of PBL: You will:
  + Be more valuable to employers/ clients
  + Learn new code faster
  + Be able to explain code better
  + Be more able to work in teams
  + Feel more confident

Structure of PBL:

* In PBL every topic is separated into three phases:
  1. Pre-Discussion
     + 10 minutes: read code & formulate problem statement
     + 40 minutes: brainstorm everything related
       - Put everything on the table
       - What do you recognize?
       - What could this mean?
     + 10 minutes: formulate 3-4 learning goals:
       - Look at the brainstorm
       - What
  2. Research & Development
     + Component 1: the learning goals
       - After the pre discussion you will usually have a lecture or workshop that helps you understand the topic
     + Component 2: exercises:
       - Most discussions have an exercise attached to them. Usually the deadline is the next day after the post-discussion
  3. Post-discussion
     + Component 1: share your findings
       - Discuss what Answers to the learning goals you found
       - Compare the different approaches you took
     + Component 2: summarize your findings
       - Take notes during the discussions
       - Doing this structurally will leave you with your own personal summary of the whole course
     + Optional depending on time: exercises
* Every phase in turn is separated into segments
* Roles in PBL:
  + Discussion Leader: keeps the group focused, keeps time and makes sure everyone gets the room to contribute
  + Note taker: writes down a mindmap/ summary of the discussion on a whiteboard public during the discussions
  + Participant: discusses the code or text to the best of their ability. Explains their knowledge and listens attentively to.
* Open-source: you can read the code behind the program
* Our daily schedule:
  + 9:30 – get settled. Have coffee.
  + 10:00 – PBL Post-discussion (for yesterday’s pre-discussion)
  + 11:00 – opportunity to modify homework
  + 12:00 – PBL pre-discussion (for tomorrow’s post-discussion)
  + 13:00 – Lunch time
  + 14:00 – Lecture/demo time
  + XX:XX - After lecture: time to work on own exercises
  + 18:00- Class ends
* More time you put, more time you have to solidify

Pre-discussion:

Lecture:

* Html: content
* Css: Styling
* JavaScript: interactive part
* Front-end: everything what is processed in your computer
  + Interface
* Back-end: everything you don’t see that happens
* AJAX most difficult in the program
* Day 31 to 46 more miscellaneous stuff
* Final month create own assignment
  + One group assignment
  + One individual assignment
* Emphasis is on back-end
  + Everything learned in the back-end is applicable in the front-end
  + Not the other way around

Lecture 1: Introduction to (JavaScript) programming:

* Console.log
  + Console.log(“hello!”);
  + Printing: something what is shown on the screen
  + Command line is a console
  + Put text and get stuff back
  + “ “: to define what should be printed. String of characters. Meaning the word are normal characters
  + Thermometer
  + ; ends the programming statement
  + Numbers are computational units
* Number operators:
  + + / \* -
  + Mathematical operators
  + Bullion operators
  + =: assigns meaning to variable number
* Everything goes from top to bottom
* String operators
* JavaScript: conditionals and loops
* Anatomy of an if statement:
  + The conditional expression is evaluated.
  + If it is true, the block within the conditional expression {…} is executed
  + If it is false, the interpreter moves to the next statement
    - Interpreter: translate source code into zero’s and one’s
  + The (else) keyword, if present, will be executed if the conditional expression in the if statement is false.
* Chaining if statements:
  + The (else) if keyword can be used to test multiple conditions
* Anatomy of a while loop:
  + A while loop is akin to a repeating if statement.
  + The conditional expression is evaluated, and if the expression is true, the code within the block is executed
  + This repeats until the condition/expression becomes false

JavaScript: Boolean logic

* Values that are true or false
* Boolean:
  + Invented by George Boole
* And (&&)
  + Both conditions must be true for the statement to be true
* Or (||)
  + If either condition is met, the statement is true.
* Not (!)
  + If true, ! will change the statement to false, and vice versa
* Boolean values can be true or false
* Order of operations (Popsicles Meld Badly):
  + Parentheses first
  + Then Math
  + Then Boolean operators

Fizbuzz: exercise:

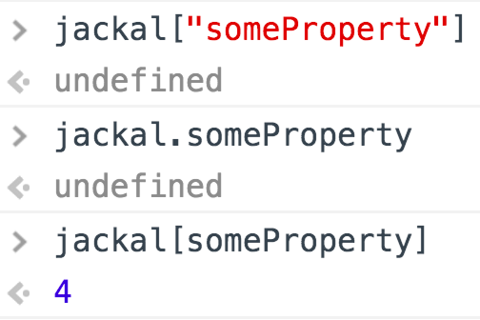
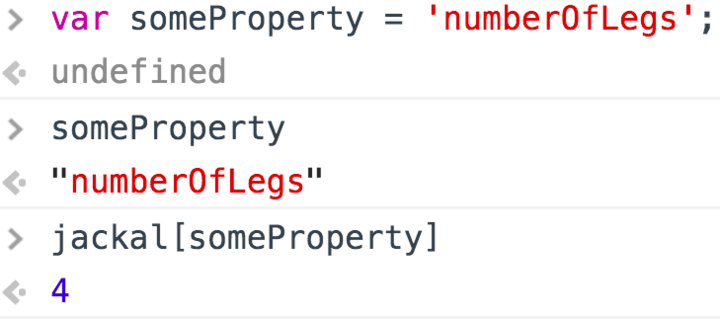
* Module: 10 % 3
  + 1

JavaScript: Arrays

* Anatomy of an array:
  + Create arrays like so: var array = []
    - Data types can be mixed but not advisable
    - Store multiple variables in one container
  + Access elements with ‘Array’ [index]
    - Note that arrays are “0-indexed”
  + Get length of arrays with array.length
* Looping through arrays:
* Looping in reverse
* Nested arrays

Objects:

* Objects have properties
* Brief review of other data types:
  + Numbers: 1,2,3
  + Strings: “Hey”, “Jude”
  + Booleans: True, False
  + Arrays: [], [7,8], [true, true, false]
* Anatomy of an object:
  + Objects have keys which point to values
    - : denotes when you go to the answer
    - Like a variable assignment
    - Difference things is values and keys are related to each other
    - {} denote start and end of the object
    - ; ends the line, one statement
    - , separate different properties and values together
  + To access a value within an object, use its key
  + Object values can also be accessed with array notation, taking in a string
    - Jackal [‘habitat’]; the same as jackal.habitat



* Object properties
  + Object properties can be set at any time
* Changing an object’s properties:
  + An object’s properties can be overwritten at will
* Object property iteration:
  + In: assigns
  + +: adds

JavaScript: Functions

* Little tiny programs in themselves
  + Meant for one specific purpose
  + Mini-programs
  + You can create many mini-programs
* We want to write mini-programs when we are finding out that we are doing the same thing over and over again
* A simple function:
  + It doesn’t do anything yet when we aren’t executing it
  + Difference between:
    - Defining
    - executing
  + Function:
    - Name of the function
    - Two parentheses
    - Code block {} which is called the function body
      * Everything in the function body will get executed when command is given
  + Return:
    - Secretly always there at the bottom
    - Returns back where it came from when function is called
  + Calling this function:
    - Name of function with parentheses and ;
    - If you see function first and afterwards the name of the function it is a definition
* Function Parameters and function arguments
  + Like statistics
  + Parameters are only inside of the function
  + At the moment when you are calling a function, you can say what a parameter is.
  + The first parameter always corresponds with the first argument
  + The second parameter always corresponds with the second argument
  + Parameters are names for function definitions
  + Arguments are real values for those parameters
* Return values:
  + With parentheses, the function will be executed
  + The return will substitute the function with the specific number or string
* Anatomy of a function:
  + This function finds the area of a triangle
  + The function is named findareaoftriangle
  + It takes in two parameters: base and height

Anonymous Functions

* Syntactic sugar: looks better but are the same
* Calling a variable the function program

Prototype:

* The standard way to create an object prototype is to use an object constructor function
* “this” is the placeholder for something to be created in the future
* “new” create a new object

Git:

* Version control system
* Keep track of the changes you put in the code
* Keep track of code during their project
* Github is the website that developers use to upload their code

jQuery: