

# Week 10 Question & Answer Key

## FILE SYSTEM NAVIGATION

4. What would you type to create a folder called *my\_awesome\_folder* on the command line?

**ANSWER:** *mkdir my\_awesome\_folder*

5. What would you type to create multiple folders with the names *awesome\_folder\_one*, *awesomeness\_two* and *awesome\_sauce* on the command line?

**ANSWER:** *mkdir awesome\_folder\_one awesomeness\_two awesome\_sauce*

6. What would you type to create a single HTML file called *mainPage*?

**ANSWER:** *touch mainPage.html*

7. What would you type to create a single Javascript file called *myScript*?

**ANSWER:** *touch myScript.js*

8. What would you type to create a single CSS file called *coolStyles*?

**ANSWER:** *touch coolStyles.css*

9. What would you type to create those same files on the command line all at once?

**ANSWER:** *touch mainPage.html coolStyles.css myScript.js*

10. What would you type to print out what directory you're currently working in on the command line?

**ANSWER:** *pwd*

11. What would you type to navigate to your home directory?

**ANSWER:** *cd ~*

12. What would you type to navigate one directory up?

**ANSWER:** *cd ..*

13. What would you type to navigate down into a folder called *my\_first\_app* in your current working directory?

**ANSWER:** *cd my\_first\_app/*

14. What would you type to print a list of the files and folders in your current working directory?

**ANSWER:** *ls*

## GIT COMMANDS

15. What would you type to initialize your current working directory as a git repository on the command line?

**ANSWER: *git init***

16. After initializing your current working directory for **git** and presuming you created a repository on Github, what would you type on the command line to add a remote and link your current working directory to the Github repository you just created? Let the link for the Github repository be: <https://github.com/zero-cool/gnarly-repository.git>

**ANSWER: *git remote add origin https://github.com/zero-cool/gnarly-repository.git***

17. What would you type to get the status of your local repository?

**ANSWER: *git status***

18. After making some changes to files and/or folders in your local repository, what would you type to stage your changes for commits on the command line?

**ANSWER: *git add .***

19. To commit your changes with a message, what would you type on the command line?

**ANSWER: *git commit -m "I made some awesome changes to this file!"***

20. What would you type on the command line to get a log of all your commits for the local repository you're currently working in?

**ANSWER: *git log***

21. What would you type on the command line to *unstage* or *reset* the already *staged* changes for a commit?

**ANSWER: *git reset* | *git reset --HARD* | *git checkout* | *git checkout [branch]***

22. What would you type on the command line to push your commits to your remote *master* branch or *myFeatures* branch?

**ANSWER: *git push origin master* | *git push origin myFeatures***

23. What would you type on the command line to fetch or download changes from a *master* branch without automatic merging?

**ANSWER: *git fetch***

24. What would you type on the command line to merge the fetched/downloaded changes from a *master* branch from previous question?

**ANSWER: *git merge* | *git merge origin/master***

25. What would you type on the command line to combine both ***git fetch*** and ***git merge***?

**ANSWER:** ***git pull***

## HEROKU - APP DEPLOYMENT

26. What would you type to create a new Heroku app for your current local repository on the command line?

**ANSWER:** ***heroku create*** | ***heroku create [name\_of\_app]***

27. What would you type to deploy your github repository code to heroku on the command line?

**ANSWER:** ***git push heroku master***

## HTML BASICS

28. List the four core DOM elements composing a web page. Fill in the blanks:

```
<!DOCTYPE html>
<__ (0) __>
  <__ (1) __>
    <title>My Cool App</title>
  </__ (2) __>
  <__ (3) __>
    <div class="cool-stuff">
      <!-- a bunch of HTML -->
    </div>
  </__ (4) __>
</__ (5) __>
```

**ANSWERS:**

(0) ***html***

(1) ***head***

(2) ***head***

(3) ***body***

(4) ***body***

(5) ***html***

29. Write out the DOM element that allows you to apply inline CSS to your webpage. Fill in the blanks:

```
<!-- Inline Styling -->
<__(0)____>
  body {
    background-color: lightgrey;
  }

  h1 {
    color: blue;
  }

  p {
    color: green;
  }
</__(1)____>
```

**ANSWERS:**

(0) *style*

(1) *style*

30. Write out the DOM element and its associated attributes that allow you to load an external stylesheet to your webpage. Fill in the blanks:

```
<!-- Load external stylesheet -->
<__(0)___ rel="__(1)____" href="styles.css" type="__(2)____">
```

**ANSWERS:**

(0) *link*

(1) *stylesheet*

(2) *text/css*

31. Write out the DOM element that allows you to execute inline Javascript on your webpage. Fill in the blanks:

```
<!-- Inline script -->
<__(0)__ type="text/javascript">

    // A bunch of Javascript....

</__(1)__>
```

**ANSWERS:**

(0) *script*

(1) *script*

32. Write out the DOM element and associated attributes that allows you to load an external Javascript file or source on your webpage. Fill in the blanks:

```
<!-- Load external Javascript -->
<__(0)__ type="__(1)__/__(2)_" src="_example.js_">

    // A bunch of Javascript...

</__(3)__>
```

**ANSWERS:**

(0) *script*

(1) *text*

(2) *javascript*

(3) *script*

## JAVASCRIPT BASICS

33. Declare a variable named **ninja**.

**ANSWER:** `var ninja;`

34. Using proper syntax, declare an empty anonymous function (no code to execute in the block) that takes in one argument named **coolness**.

**ANSWER:** `function (coolness) { //....code };`

35. Using proper syntax, declare an empty named function (no code to execute in the block) with the name **beAwesome** that takes in one argument named **awesomeSauce**.

**ANSWER:** `function beAwesome(awesomeSauce) { //...code };`

36. Using proper syntax, declare an empty anonymous function expression with a variable named **beAwesomer** that takes in one argument named **awesomerSauce**.

**ANSWER:** `var beAwesomer = function(awesomerSauce) { //...code };`

37. Using proper VanillaJS syntax, write the most widely supported function that loads the script(s) after all DOM elements and media content (pictures, videos, etc.) have been loaded:

**ANSWER:** `window.onload = function() { //...code };`

38. Using proper jQuery syntax, write the function that you would put at the beginning of a Javascript file to load the script after just the DOM elements have loaded on the web page:

**ANSWER:** `$(document).ready(function() { //...code });`

39. Why is `$(document).ready()` better than `window.onload`?

**ANSWER:** `$(document).ready()` will load the script after just the DOM elements have loaded and won't wait for the media content to complete loading, whereas `window.onload` will wait for all DOM elements and media content to load before loading the script, which may take some time.

40. Create an array called **myCoolArray** with four string elements of your choosing.

**ANSWER:** One solution may look like => `var myCoolArray = ["hello", "yerp", "world", "hacktheplanet"];`

41. Write out how you would access the third element in the array from the previous question you created in the previous question.

**ANSWER:** `myCoolArray[2];`

42. Create an object called **myObject** with four properties:

- **myName** property with a string value of your name,
- **myBirthMonth** property with a string value of your birth month,
- **myAlertFunc** property whose value is a function that takes in one argument (string type) and invokes an alert with the argument text in it
- **myAdditionFunc** property whose value is a function that takes in two arguments (integer type), adds those two arguments, then returns the sum/answer).

**ANSWER:** One solution could look like:

```
var myObject = {  
  myName: "Albert",  
  myBirthMonth: "February",  
  myAlertFunc: function(arg) {  
    alert(arg);  
  },  
  myAdditionFunc: function(arg1, arg2) {  
    var sum = arg1 + arg2;  
    return sum;  
  }  
};
```

43. Using the object that was just created, output the values of the four object properties to the browser console using console.log (for i,ii,iv ONLY) and dot notation.

**ANSWER:**

1. `console.log(myObject.myName);`
2. `console.log(myObject.myBirthMonth);`
3. `myObject.myAlertFunc("hello there");`
4. `console.log(myObject.myAdditionFunc(3,4));`

44. Given the following array of objects:

```
var superheroes = [  
  {  
    name: 'The Hulk',  
    type: 'Human/Mutant',  
    superPower: 'Hulk Smash'  
  },  
  {  
    name: 'Ironman',  
    type: 'Human',  
    superPower: 'One Person Combat Machine'  
  },  
  {  
    name: 'Hawkeye',  
    type: 'Human',  
    superPower: 'Master Marksman'  
  },  
  {  
    name: 'Thor',  
    type: 'Asgardian',  
    superPower: 'God of Thunder'  
  }  
]
```

Write out the syntax using dot notation to access the second element in the array and print out all of its properties:

**ANSWER:**

1. *superheroes[1].name*
2. *superheroes[1].type*
3. *superheroes[1].superPower*



45. Given the following **array** and **if-else** conditional:

```
var superheroes = [
  {
    name: 'The Hulk',
    type: 'Human/Mutant',
    superPower: 'Hulk Smash',
    powerLevel: 1500
  },
  {
    name: 'Ironman',
    type: 'Human',
    superPower: 'One Person Combat Machine',
    powerLevel: 950
  },
  {
    name: 'Hawkeye',
    type: 'Human',
    superPower: 'Master Marksman',
    powerLevel: 600
  },
  {
    name: 'Thor',
    type: 'Asgardian',
    superPower: 'God of Thunder',
    powerLevel: 2000
  }
]

if (superheroes[0].powerLevel > superheroes[3].powerLevel) {
  console.log(superheroes[0].name + " wins!");
} else {
  console.log(superheroes[3].name + " wins!");
}
```

What would be printed out to the console?

**ANSWER:** *"Thor wins!"*

46. Given the following **switch-case** statement:

```
var universe = 'Star Wars';

switch (universe) {
  case 'DC':
    console.log('Spiderman lives here!');
    break;
  case 'Marvel':
    console.log('The X-Men live here!');
    break;
  case 'Star Wars':
    console.log('The Jedi live here!');
    break;
  case 'Star Trek':
    console.log('The Federation lives here!');
    break;
  default:
    console.log('Everyone lives here!');
}
```

What would be printed to the console?

**ANSWER:** *“The Jedi live here!”*

47. What boolean value does **2 === “2”** evaluate to?

**ANSWER:** *false*

48: What boolean value does **3 !== 3** evaluate to?

**ANSWER:** *false*

49. Given the following **array** and **ternary if-else conditional**:

```
var superheroes = [  
  {  
    name: 'The Hulk',  
    type: 'Human/Mutant',  
    superPower: 'Hulk Smash',  
    powerLevel: 1500  
  },  
  {  
    name: 'Ironman',  
    type: 'Human',  
    superPower: 'One Person Combat Machine',  
    powerLevel: 950  
  },  
  {  
    name: 'Hawkeye',  
    type: 'Human',  
    superPower: 'Master Marksman',  
    powerLevel: 600  
  },  
  {  
    name: 'Thor',  
    type: 'Asgardian',  
    superPower: 'God of Thunder',  
    powerLevel: 2000  
  },  
  {  
    name: 'Loki',  
    type: 'Asgardian',  
    superPower: 'Mischief',  
    powerLevel: 2000  
  }  
]
```

```
superheroes[3].powerLevel <= superheroes[4].powerLevel ? console.log('The conflict ensues') : console.log('The battle stops here!');
```

What would be printed to the console?

**ANSWER:** *“The conflict ensues”*

50. What would  $2 + 5$  evaluate to?

**ANSWER:** 7

51. What would  $200 - 100$  evaluate to?

**ANSWER: 100**

52. What would  $10 * 50$  evaluate to?

**ANSWER: 500**

53. What would  $60 / 12$  evaluate to?

**ANSWER: 5**

54. What would  $22 \% 10$  evaluate to?

**ANSWER: 2**

Given the following scenario:

```
var counter = 10;  
  
counter++;  
  
console.log(counter);
```

55. What would be printed to the console?

**ANSWER: 11, data type is Number**

Given the following scenario:

```
var counter = 20;  
  
counter--;  
  
console.log(counter);
```

56. What would be printed to the console?

**ANSWER: 19, data type is Number**

Given the following **for** loop:

```
var jedi = ['Yoda', 'Mace Windu', 'Qui-Gon', 'Obi-Wan', 'Luke'];  
  
for (____;____;____) {  
  console.log(jedi[k]);  
};
```

57. What would be inside the parentheses of the **for** loop?

**ANSWER:** `var k = 0 ; k < jedi.length; k ++`

58. Using VanillaJS, what would you type to add an *Event Listener* to an element with **id="myButton"** on the DOM for a click event and using an anonymous function as a callback, have it output an alert with the text "Hello World!"?

**ANSWER:** `document.getElementById("myButton").addEventListener("click", function() { alert("Hello World!"); }); | .click(function() { alert("Hello World!"); }) | $(document).on("click", "button", function() {alert("Hello World!");`

59. Using jQuery, what would you type to add an *Event Listener* to an element with **id="myButton"** on the DOM for a click event and using an anonymous function as a callback, have it output an alert with the text "Hello World!"?

**ANSWER:** `$("#myButton").on("click", function() { alert("Hello World!"); });`

60. Given elements on the DOM with **class="images"**, write out how you would select those elements with VanillaJS and assign it to a variable named **elementz**.

**ANSWER:** `var elementz = document.getElementsByClassName('images');`

61. Given an element on the DOM with an **id="menu-slider"**, write out how you would select that element with VanillaJS and assign it to a variable named **coolSlider**.

**ANSWER:** `var coolSlider = document.getElementById('menu-slider');`

62. Given an element on the DOM with **class="article-section"**, write out how you would get all the elements contained within it.

**ANSWER:** `$(".article-section").children();`

63. Given an element on the DOM with an **id="win-count"**, write out how you would get the single **p** tag contained within it if you don't know where it's positioned within it.

**ANSWER:** `$("#win-count").find("p"); | $("#win-count").has("p") | $("#win-count p") | $("#win-count > p") | $("#win-count").closest("p") | $("#win-count").children("p")`

```
<a class="link" href="http://cnn.com">CNN</a>
```

64 a). Using VanillaJS and given the element above, write out how you would change the **href** attribute to the value "<http://inc.com>"

**ANSWER:** `document.querySelector(".link").setAttribute("href", "http://inc.com"); |  
element.href | document.getElementsByClassName("link").setAttribute('href',  
"http://inc.com");`

64 b). Using jQuery and given the same **anchor** tag element above, write out how you would change the **href** attribute to the value "<http://inc.com>".

**ANSWER:** `$(".link").attr("href", "http://inc.com");`

```
<div id="like-counter"></div>
```

65. Using VanillaJS and given the element above, write out how you would change the **width** of the **div** to **100px**.

**ANSWER:** One solution could look like =>

`var likeCounter = document.querySelector("#like-counter");  
likeCounter.style.width = "100px"; |  
document.getElementById('like-counter').style.width = "100px"`

66. Using jQuery and given the same element above, write out how you would change the **width** of the **div** to **100px**.

**ANSWER:**

`$("#like-counter").css("width", "100px"); |  
$("#like-counter").css({"width": "100px"}); |  
$("#like-counter").width("100px"); |  
$("#like-counter").width(100)`

67. Fill in the blank line below to complete the callback function using a function expression.

```
var foo = function() {  
  // ...some JS code  
};  
  
$(document).on("click", "button", _____);
```

**ANSWER:** `$(document).on("click", "button", foo);`

68. Given the event listener below, fill in the blank line to complete the callback function using an anonymous function (just make it an empty block).

```
.on("click", "button", _____);
```

**ANSWER:**

```
$(document).on("click", "button", function() {  
  // ...some JS code  
});
```

## APPLICATION PROGRAM INTERFACE - CONSUMING

69. Using jQuery to make an ajax call to the Github API to retrieve data, fill in the blank:

```
$.ajax({  
  method: _____,  
  url: 'http://api.github.com/username/repos',  
})._____ (function(data, status, jqXHR) {  
  // some JS code to access manipulate the data variable  
})._____ (function(jqXHR, status, error) {  
  // some JS code to show the error  
});
```

**ANSWER:**

1. **'GET'**
2. **done**
3. **fail**



# FIREBASE

70. Given the code in the image below, complete the blanks to create a reference to a **Firestore** database so we can store data:

```
<!-- Firebase with saving data -->
<!DOCTYPE html>
<html>
  <head>
    <title>My Ninja App</title>
    <script src="https://cdn.firebase.com/js/client/2.4.1/firebase.js"></script>
  </head>
  <body>
    <!-- A bunch of HTML -->
    <script type="text/javascript">
      var ref = __ (0) __ __ (1) __ ("ninja-catalog.firebaseio.com");

      // Initial Values
      var nickName = "";
      var weapon = "";
      var powerLevel = 0;

      // Capture Button Click
      $("#addUser").on("click", function() {

        // Grabbed values from text boxes
        nickName = $('#nickNameInput').val().trim();
        weapon = $('#weaponInput').val().trim();
        powerLevel = $('#powerLevelInput').val().trim();

        // Code for pushing our data to our Firebase database
        ref.__ (2) __ ({
          nickName: nickName,
          weapon: weapon,
          powerLevel: powerLevel
        })

        // Don't refresh the page!
        return false;
      });
    </script>
  </body>
</html>
```

ANSWER:

(0) new

(1) Firestore

(2) push

71. Given the code in the image below, complete the blanks to listen to your **Firestore** database when your data changes with `.on("value", callback)` and have it reflect on the DOM:

```
// Don't refresh the page!
return false;
});

//Firestore watcher + initial loader
ref._(3)_( "__ (4) __", function(snapshot) {

    // Log everything that's coming out of snapshot
    console.log(snapshot.val());
    console.log(snapshot.val().nickName);
    console.log(snapshot.val().weapon);
    console.log(snapshot.val().powerLevel);

    // Change the HTML to reflect using jQuery
    (5)_( "#nickNameDisplay" ).html(snapshot.val().nickName);
    (6)_( "#weaponDisplay" ).html(snapshot.val().weapon);
    (7)_( "#powerLevelDisplay" ).html(snapshot.val().powerLevel);

    // Error handling
}, function(errorObject) {

    console.log("Errors handled: " + errorObject.code)
});
```

ANSWER:

(3) on

(4) value

(5) \$

(6) \$

(7) \$

72. Given the code in the image below, complete the blanks to listen to your **Firestore** database when a *new child* is added with `.on("child_added", callback)`.

```
//Firestore watcher + initial loader
ref.__(8)__("__(9)___", function(childSnapshot) {
  // Log everything that's coming out of snapshot
  console.log(childSnapshot.val().nickName);
  console.log(childSnapshot.val().weapon);
  console.log(childSnapshot.val().powerLevel);

  // Handle the errors
}, function(errorObject){
  console.log("Errors handled: " + errorObject.code);
});
```

ANSWER:

(8) on

(9) child\_added